TWLT18
CEvoLE 2

Learning to Behave
Workshop II: Internalising Knowledge

PROCEEDINGS OF THE EIGHTEENTH
TWENTE WORKSHOP ON LANGUAGE TECHNOLOGY

JOINT WITH

THE SECOND CELE WORKSHOP ON
EVOLUTIONARY LANGUAGE ENGINEERING

NOVEMBER 22-24, 2000
IEPER, BELGIUM

K. Jokinen, D. Heylen,
and A. Nijholt (eds.)

CIP GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Jokinen K., Heylen D., Nijholt A.

Learning to Behave
Proceedings Twente Workshop on Language Technology 18
CELE Workshop on Evolutionary Language Engineering 2
Workshop II: Internalising Knowledge
K. Jokinen, D. Heylen, A. Nijholt (eds.)
Ieper, Centre for Evolutionary Language Engineering

ISSN 0929–0672

trefwoorden: neural networks, brain-like computing, language learning, cognitive models of language processing, evolutionary methods, speech and music perception

Book orders:
Ms. C. Bijron
University of Twente
Dept. of Computer Science
P.O. Box 217
NL 7500 AE Enschede
tel: +31 53 4893680 – fax: +31 53 4893503
Email: bijron@cs.utwente.nl

Druk- en bindwerk: Reprografie U.T. Service Centrum, Enschede

Preface

TWLT is an acronym of Twente Workshop(s) on Language Technology. Over the years, the topics covered in this series of workshops have evolved from natural language theory and technology to other aspects of human-computer interaction. The workshops are organized by the Parlevink Research Project, a language theory and technology project of the Centre of Telematics and Information Technology (CTIT) of the University of Twente, Enschede, The Netherlands. For each workshop proceedings are published containing the papers that were presented. For previous volumes in the series see the final pages of the present volume or consult *http://parlevink.cs.utwente.nl/Conferences/twltseries.html*.

CEvoLE is an acronym for CELE workshop(s) on Evolutionary Language Engineering. These are workshops organised by CELE, or the Centre for Evolutionary Language Engineering. CELE is a research group under S.AI.L Trust vzw, in Ieper, Belgium, focussing on speech and language research using neurocomputational methods and advanced hardware.

The TWLT 18/CEvoLE 2 workshop *Internalising Knowledge*, is organised by the Parlevink Research Project and CELE on November 22-24, 2000. It is the second part of the workshop series *Learning to Behave* which focusses on human-agent interaction and information processing. The first part, TWLT 17/ CEvoLE 1, *Interacting Agents*, was hosted by the Parlevink Group in Enschede (The Netherlands) one month earlier on October 18-20, 2000. The pair of workshops jointly investigate human-agent interaction and knowledge both on the level of agents communicating with the external environment and on the level of the internal agent processes that guide the modelling and understanding of the external sensory input in the brain.

The programme committee of the workshops consisted of three chairs, Anton Nijholt (Twente), Kristiina Jokinen (CELE), Dirk Heylen (Twente) and the following members: Mahoro Anabuki (USA), Elisabeth André (Germany), Norman I. Badler (USA), Justine Cassell (USA), Marc Cavazza (UK), Kristina Höök (Sweden), Lewis Johnson (USA), Marc Leman (Belgium), Nadia Magnenat-Thalmann (Switzerland), Risto Miikkulainen (USA), Klaus Obermayer (Germany), Luc Steels (Belgium). We wish to acknowledge the support of the honorary chairs Dirk Frimout (CELE) and Roel Pieper (Twente).

The CEvoLE 2/TWLT 18 workshop on Internalising Knowledge concentrates on internal aspects of learning via interaction: computation in brain-like systems. The goal is to investigate cognitive models for information processing and coordination, especially how symbolic processing, conceptualising and language learning take place in neural models. This covers investigations on the special characteristics of speech and language perception (including musical patterns), on the architecture and functional diversity of the brain and brain-like computational models, as well as on the methods and representations for efficient information processing and emergence of discreet symbols that are used as a means of thought and language.

Workshops involve the joint action of many persons. We are grateful to the members of the programme committee for their help in the refereeing process. We thank all the authors for their contributions. For all their efforts put in the local organisation of the workshops we thank Charlotte Bijron, Betsy van Dijk, Hendri Hondorp and Alice Vissers-Schotmeijer in Twente; in Ieper: Stefaan Decorte, Sven Degroeve, Liesbeth Houthave, Yvan Saeys, Anne Marie Van Hee and Herwig Van Marck.

Kristiina Jokinen, Dirk Heylen and Anton Nijholt November 2000

Previous TWLT workshops

Previous TWLT workshops were

TWLT1,	*Tomita's Algorithm: Extensions and Applications*. 22 March, 1991.
TWLT2,	*Linguistic Engineering: Tools and Products*. 20 November, 1991.
TWLT3,	*Connectionism and Natural Language Processing*. 12 and 13 May 1992.
TWLT4,	*Pragmatics in Language Technology*. 23 September, 1992.
TWLT5,	*Natural Language Interfaces*. 3 and 4 June, 1993.
TWLT6,	*Natural Language Parsing*. 16 and 17 December, 1993.
TWLT7,	*Computer-Assisted Language Learning*. 16 and 17 June 1994.
TWLT8,	*Speech and Language Engineering*. 1 and 2 December 1994.
TWLT9,	*Corpus-based Approaches to Dialogue Modelling*. 9 June, 1995.
TWLT10,	*Algebraic Methods in Language Processing*. 6-8 December, 1995.
TWLT11,	*Dialogue Management in Natural Language Systems*. 19-21 June, 1996.
TWLT12,	*Automatic Interpretation and Generation of Verbal Humor*. 11-14 September 1996.
TWLT13,	*Formal Semantics and Pragmatics of Dialogue, Twendial'98*. 13-15 May 1998.
TWLT14,	*Language Technology in Multimedia Information Retrieval*. 7-8 December 1998.
TWLT15,	*Interactions in Virtual Environments*. 19-21 May 1999.
TWLT16,	*Algebraic Methods in Language Processing (AMiLP2000)*. 20-22 May 2000.
TWLT17,	*Learning to Behave (CEvoLE1)*. 18-20 Oct 2000.

For the contents of the previous proceedings, please consult the last pages of this volume.

Sponsors

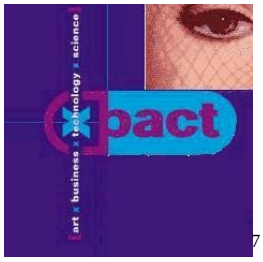
1 2

3

4

5

Fonds voor **W**etenschappelijk **O**nderzoek – Vlaanderen[6]



7

8

[1] http://vr-valley.com
[2] http://www.utwente.nl
[3] http://www.sail.com
[4] http://www.daikin.com
[5] http://www.lhs.com
[6] http://www.nfwo.be
[7] http://www.x-pact.nl
[8] http://www.nwo.nl

Contents

Invited Talks

Regular Talks

Invited Talks without paper

Evolving Populations of Expert Neural Networks

Joseph Bruce and Risto Miikkulainen
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712 USA
{jbruce|risto}@cs.utexas.edu

Abstract

In standard neuroevolution, the goal is to evolve one neural network that would compute the right answer most often. However, it often turns out that the population as a whole could perform even better, if we could only choose the right network for each input. One way to do this is to evolve networks to output not only the answer, but also a confidence value that the answer is correct. Experiments in the handwritten character recognition domain show that such an evolutionary process creates a diverse population of networks that are experts in particular kinds of inputs, and collectively perform better than any individual network.

Keywords: Neuroevolution, Confidence, Niching, Speciation, Expert Networks.

1 Introduction

In a typical approach to problem solving with evolutionary methods, a genetic algorithm is used to evolve a population of individuals each attempting to solve the task. The most fit individual found during the evolution, the *champion*, is designated as the final result. For example, when neural networks are evolved for a decision task, the champion is the neural network that is most likely to produce the correct decision for any given input. The rest of the population, and the knowledge and expertise it encodes, is simply thrown away.

However, if the final population is analyzed in more detail, an interesting and potentially useful observation emerges. There are often other individuals in the population that are able to produce correct decisions for inputs that the champion cannot handle very well. For example figure 1 shows the fitness of the final population in the handwritten character recognition task. Although the champion only identifies 84% of the characters correctly, 98% of the characters are correctly identified by *some* individual in the population.

In principle, we could obtain this level of accuracy by simply choosing the best answer from those produced by the entire population. But how can we determine which individual most likely has the right answer for a given input? One simple solution is to evolve the individuals not only to produce the answer, but also a *confidence* that their answer is correct. The most likely answer can then be obtained from the population as the one associated with the highest confidence.

The viability of this idea is tested in this paper in the handwritten character recognition task. For the method to work, it turns out essential to maintain high diversity in the population. Several speciation methods are tested; the island model and its continuous version, the spatial model, are found to be the most effective. With such diversity, the method leads to populations that collectively perform better than any single individual. Confidence evolution of expert neural networks therefore constitutes a promising approach to utilizing the entire population as the result of the evolutionary algorithm.

2 The Method of Confidence Evolution

The lesson from figure 1 is that in order to make use of the knowledge and expertise in the entire population, the champion should be determined separately for each individual input. Such a

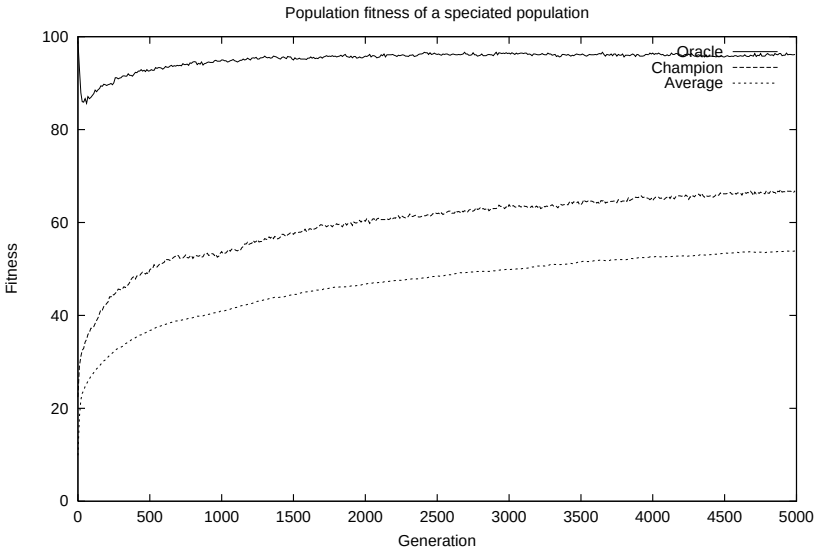


Figure 1: **Percent of correct decisions attainable from a standard neuroevolution population.** Three different measures are used, from top to bottom: the best answer found in the entire population (unreliazable in practice), the answer of the most fit individual, and the population average. Fitness indicates the percentage of correctly-identified test characters in the handwritten character recognition domain (section 4).

selection can be correctly made only based on the correct answer, which is not available during performance. Therefore, although such a high level of performance exists in the population, it is unattainable in practice.

However, it is possible to approximate the selection in various ways. For example, it may be possible to train (or evolve) a meta-level neural network to decide which individual in the population is most likely to produce the right answer for a given input. This is the approach taken for example in the Mixtures of Experts approach Jacobs et al. (1991), which works well in many supervised tasks. An interesting alternative is to require each individual to rate the quality of each answer it produces. If the population learns to do this effectively, one would be able to outperform the champion by choosing the decision of an individual reporting the highest level of confidence for each decision. This is the approach taken in this paper.

Confidence may be represented by an additional output unit on the neural network. Confidence is that unit's activation when the network is presented with an input. The range of confidence, therefore, is between 0 and 1. To encourage the network to output useful estimates through this unit, the fitness evaluation must be altered. Many fitness evaluations in decision tasks are sums of Booleans, counting the number of correct decisions an individual makes in trials on a training set:

$$f = \sum_i s(\vec{v}_i) \tag{1}$$

where $s(\vec{v}_i) = 1$ if the network's answer on trial i was correct, 0 otherwise.

Training with confidence changes this evaluation in a simple way. It treats each of these trials as a bet. Instead of simply winning 1 each time it is correct, it also stands to lose 1 if it is incorrect. Moreover, the size of the bet is determined by its confidence output $c(\vec{v}_i)$:

$$f = \sum_i s(\vec{v}_i)c(\vec{v}_i) \tag{2}$$

where $s(\vec{v}_i) = 1$ if the answer is correct, $s(\vec{v}_i) = -1$ if it is incorrect. In other words, the

network is penalized $s(\vec{v}_i)c(\vec{v}_i)$ for a wrong decision on input $\vec{v}_i$ and is awarded $s(\vec{v}_i)c(\vec{v}_i)$ for a correct decision. This process allows the network to unilaterally set the amount of the bet that its response is correct. It encourages networks that output high confidence only for decisions that are likely to be correct.

The fundamental change to the standard way of evolving neural networks is that in an evolution with confidence, the entire population is considered to be the product of evolution. Answers may be obtained from the population by simply choosing the answer provided by the most confident individual. The specific method for leveraging confidence to extract high-quality decision may be problem-dependent. For example, for some decisions tasks the sum of the population's outputs weighted by confidence might be a useful quantity. But in other domains, such as robotic controllers, it would be imperative to allow a single individual to provide the entire decision: weighted sums of different trajectories could easily lead to a disaster.

However, in order to use confidence, the population must be diverse enough so that significantly different decisions are made by networks inhabiting distinct niches. A strong method of diversity maintenance (also called speciation or niching), is therefore crucial for success. On the other hand, the restrictions imposed by the speciation technique may adversely affect learning performance for all individuals in the population. A proper speciation technique should strike a balance between these two factors.

3 Methods of Speciation

Speciation is an important design principle in genetic algorithms in general. Genetic algorithms lose diversity over the course of evolution, possibly converging to a point at which all of the genomes in the population are essentially the same. At this point, crossover operation between two nearly identical genomes is unlikely to create a more fit offspring, and the progress of the GA from that point will be slow (mostly through mutation). A properly speciated population, containing several "niches" of solutions to the task, can continue improving even after some of the niches reach local maxima from which they are unable to improve. It provides for a more reliable search, with many approaches being explored in parallel throughout the evolution.

Speciation techniques are generally implemented in terms of genomes, rather than the structure implemented by the genomes, or that structure's performance, because a diversity of genomes is needed to search the solution space in parallel. However, this diversity of genomes also results in diversity of behaviors. A speciated population contains a wider range of answers—and is more likely to contain at least one correct response for a particular input—than a homogeneous population. Therefore, speciation can be used to create a population where different individuals are responsible for different inputs.

Methods for promoting diversity may involve changes to different parts of the canonical genetic algorithm. In this paper we will compare speciation techniques that modify the GA's selection scheme, the replacement scheme, and the fitness evaluation function. Also crucial to population diversity is the scaling scheme, i.e. the algorithm that converts the individuals' fitnesses into probabilities of being included in the next generation, either unchanged or combined with another genome by crossover. An aggressive scaling scheme that rewards slightly fitter individuals with much higher probabilities will quickly lead to convergence, as much of the genetic material possessed by moderate individuals will be lost in each generation. More subtle scaling schemes are desirable (and also used in this paper) to delay population convergence.

A very simple approach to speciation is to arbitrarily divide the population into non-interacting subpopulations, or islands. A genome cannot perform crossover with any genome of another island, and a newly created individual may replace only a genome in the island of its parents. In some versions, the island model provides for a small rate of migration between islands. Without migration, this approach is equivalent to running a genetic algorithm independently on each of the islands. Even this trivial approach to speciation can be useful; if the genomes in one island reach a plateau early, others may continue improving. This difference is not directly promoted; it is simply allowed to occur by chance. Under migration, populations on an island are allowed time to make small adjustments before competing with outside genomes Mühlenbein (1991).

A more general, continuous version of the island method is the spatial, or topological, method. Each individual may inhabit a vertex of an undirected graph, and it may only perform crossover with an individual connected to it through an edge. The resulting offspring may only replace the least fit of its parents, only if the offspring is more fit. This setup is more biologically plausible than the usual "panmictic" populations in which any individual may mate with any other. It prevents premature convergence since a particular genome can spread only to immediate neighbors in a single timestep Kephart (1994).

The implementation of a spatial population described above also incorporates the more general speciation strategy called preselection, which stipulates that a newly created individual in a population may only replace one of its parents. This protects against premature convergence because it ensures that at least some of an individual's genetic material will survive into the next generation Goldberg (1989).

Fitness sharing is a technique that penalizes genomes that inhabit neighborhoods of many other genomes. Generally, an individual's fitness evaluation is divided by a sharing factor that measures the genome's proximity to others in the population. Genomes in heavily populated peaks receive a high penalty, which translates into a lower probability of propagating to the next generation. This technique is intended to spread the population across several peaks in the solution space, with larger (wider or higher) peaks able to support more individuals. Implicit sharing is a variation in which, for each input, only the individual with the best response from a randomly-selected subset of the population is awarded fitness for that input Darwen and Yao (1996).

Crowding is a generalization of preselection, where an individual only replaces a genome to which it is similar (but not necessarily the parent). Under crowding, after a new genome is created, a subset of genomes is randomly selected from the population. The genome in the subset which is closest to the new genome is chosen to be replaced by the new genome Goldberg (1989).

The confidence method was tested in conjuction of each of the above speciation methods. Interestingly, their performance was found to differ a lot in the handwritten character recognition domain, which will be described next.

4 Experiments

The method of confidence evolution was applied to the standard benchmark task of recognizing handwritten digits. There are many methods developed specifically for this task. The present goal is not to compete with them, but rather to test the viability of the method and to refine it further. This task is well-suited for such analysis because the correct decisions are readily available. After the method has been tested and refined, it will be possible to apply it to other task where the correct performance is not known.

The data set used was the freely-available subset of 2,992 examples of handwritten digits 0..9 in the NIST database, scaled to an accuracy of 8×8 pixels Choe et al. (1996). The networks had an input layer of 64 units to encode the 8×8 input representing a digit to be classified. The input layer was fully connected to a hidden layer of 20 units, which was fully connected to an output layer of 11 units, representing each of the ten possible digit classifications, and an additional unit to output the confidence in this classification. The output unit with the highest activation was chosen as the classification. The genome represented the real values of the weights and biases of the network. While the canonical GA operates only on binary strings, analogous operations of crossover and mutation were implemented for the real-valued genome: a uniform crossover could take place between weights, and each weight was mutated with a 0.01 probability by adding a normally distributed random value of 0 mean and 1.0 standard deviation to the weight.

The genetic algorithm proceeded on a population of 100 individuals for 5,000 generations. During each generation, the fitness for each network was calculated according to equation 2 (and equation 1 for those experiments where confidence was not used) on a training set of 2,000 patterns, selected randomly among the 2,992 for each experiment. Fitness scaling was done by sigma truncation scaling Goldberg (1989), which tolerates negative fitness values. Selection was fitness-proportionate. Throughout evolution, each population was tested on a randomly-chosen test set of 200 patterns not part of the training set.

Notice that a more accurate fitness evaluation could easily be designed, such as the sum of squared errors between the outputs and the target output. Such an evaluation would capture more information about the differences between individuals; however, Boolean values were chosen in order to emulate less well-understood decision tasks for which such detailed information is not available. Furthermore, the fitness evaluation seeks to reward only the desired outcome (the correct classification in the winner-takes-all sense), not any specific way of attaining the outcome. Having such an open-ended fitness evaluation allows the network to implement its own, possibly unexpected, method for achieving the result Floreano and Urzelai (2000).

The different speciation techniques outlined in section 3 were each tested as part of the confidence evolution method. The island method was implemented by splitting the 100 genomes into 10 noninteracting islands of size 10, with no migration. A spatial population was laid out on a 10×10 grid with edges folded back to create a torus; an individual could mate only with one of its four neighbors, with the new child replacing the parent with lower fitness, only if the child's fitness was higher. Fitness sharing was implemented by penalizing genomes according to their proximity to others in the population in the Euclidean sense. Crowding was implemented such that a new individual was placed in the population in place of the closest Euclidean neighbor in a random subset of 10 from the population. In preselection, an offspring replaced the parent with the lowest fitness if the offspring was more fit than that parent.

5 Results

First, it is interesting to find out how well each speciation method was able to maintain diversity in the evolution. The average Euclidean distance between the 100 genomes in the population throughout evolution (without confidence) is plotted in figure 2 for each method, and also for a control evolution with no speciation. The control evolution quickly lost much genomic diversity, bottoming out at approximately generation 200. This result highlights the need for speciation.

Of the speciation techniques, the island model and the spatial population, i.e. those methods that directly restrict mating to a local neighborhood, were able to maintain diversity most effectively. Evolutions with crowding, sharing, and preselection each slowed convergence compared to the evolutions with no speciation, but were considerably less effective than the best techniques. These techniques do not restrict replacement as strongly as the best techniques do; they involve a high degree of randomization in the choice of which population member a new genome should replace. Since this decision is randomized, sampling error can affect replacement, causing genetic drift. Mahfoud (1992) cites this stochastic error as a difficulty with crowding and preselection.

As expected, those speciation methods that maintained the highest diversity also provided the best advantage for confidence evolution. Populations evolved using the island model and the spatial model were diverse enough so that choosing the answer of the most confident individual resulted in better performance than could be obtained from the population's champion. Figure 3 compares the accuracy in the test set of the (unattainable) best answers, answers selected by confidence, and the most fit individual for the island model.

The accuracy of the different methods in the test set is compared in figure 4. The main result is that confidence evolution with islands resulted in a slightly higher level of performance than the control evolution. The other methods were worse, underlining the importance of an effective speciation method in confidence evolution. Crowding, in particular, did not perform above chance; apparently one or more high-bidding and wrong individuals persisted thoughout the evolution, never being replaced because they were too far from the other genomes. This result demonstrates the difference between random diversity (such as the population before the first generation), and useful diversity (niches of high-performing but different individuals). Only the latter kind of diversity is useful for confidence evolution.

Interestingly, when speciation methods were added to the standard evolution, the performance either was not affected or actually became worse. This result suggests that making full use of speciation requires a technique such as confidence. It also shows that the improvement over standard evolution is indeed due to confidence, and not speciation.

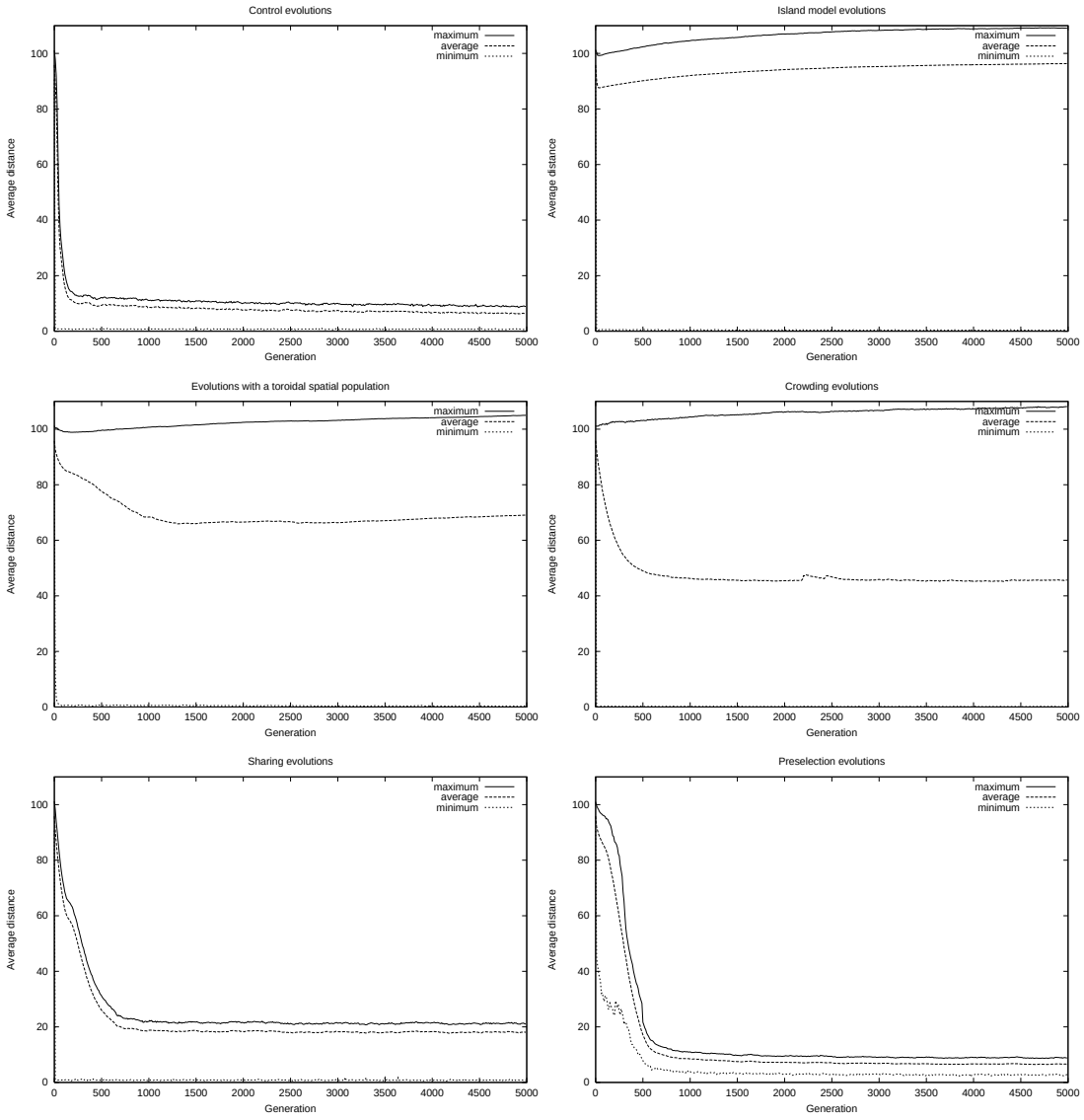


Figure 2: **Population diversity under different speciation methods.** The plots (from top to bottom) show the maximum, average and minimum Euclidean distances between genomes in the population as evolution progresses. Plots are each averaged over 10 independent evolutions.

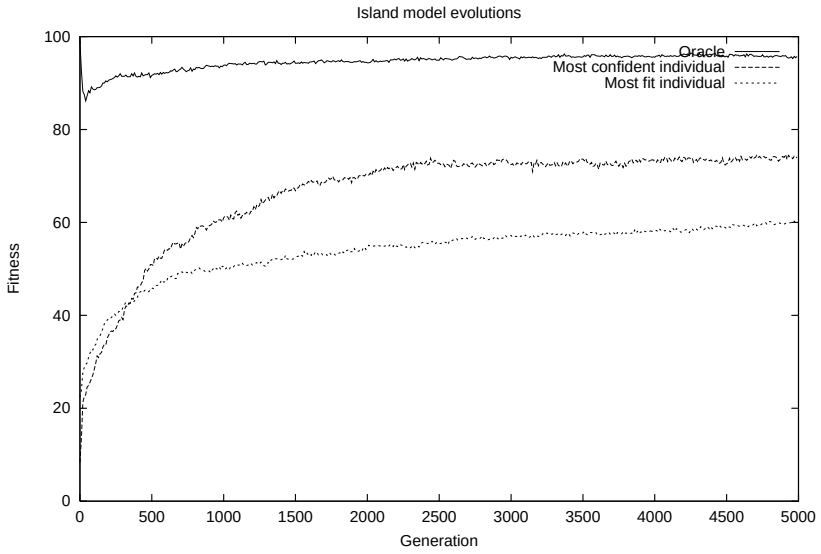


Figure 3: **Accuracy of confidence evolution with islands.** On top, the fitness obtained by choosing (through an unrealizable oracle) the best answer in the entire population is plotted. In the middle, the fitness obtained by choosing the answer of the most confident individual is shown. The fitness of the most fit individual is plotted in the bottom. The plots are averages over 10 runs.

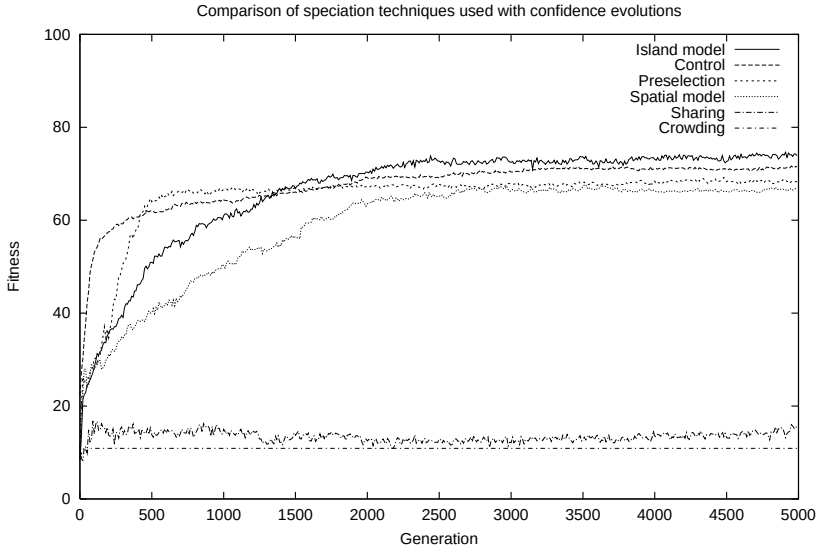


Figure 4: **Accuracy of confidence evolution with different speciation methods vs. standard evolution.** The island method performs the best; the other methods are weaker than the standard evolution, labeled "Control", in the order shown in the legend. The plots are averages over 10 runs.

6 Discussion and Future Work

The experimental results in this paper show that confidence evolution can improve performance of neuroevolution in the handwritten digit recognition task. They also show that effective speciation is crucial for this technique. How general are these results?

Speciation is generally used in a genetic algorithm to increase the overall rate of learning by avoiding, or slowing down, population convergence. However, when speciation is used in conjunction with confidence evolution, the goal is instead to increase the variation in answers made. The existing speciation techniques used in this paper may not be the best for this new, slightly different goal. This issue is underscored by the fact that confidence provided the greatest advantage in evolutions with completely noninteracting subpopulations—a technique that would tend to harm overall fitness since small populations are being evolved in each island, leading to cruder solutions.

In the extreme, a speciation technique that even significantly decreases the overall fitness of the population would work well with confidence evolution if it maintains a large variety of correct answers. This is a tradeoff that is not acceptable in a standard genetic algorithm: if the goal of speciation is to increase the overall rate of learning, a technique that lowered the fitness of all individuals significantly would not be useful. But given this new set of criteria, perhaps such strong speciation techniques could be devised in the future, gaining even more benefit from confidence evolution than is possible with the current techniques.

It is also possible that domains other than character recognition might be more amenable to the current speciation techniques. This domain benefits from a large number of individuals fine-tuning an approximate solution in a small space, which requires exactly the convergence that speciation attempts to avoid. Instead, genetic algorithms in general and speciation methods in particular are strongest at quickly finding approximate solutions. Therefore, problems that involve more global search may be more amenable to the current techniques.

In particular, genetic algorithms are the method of choice for sequential decision tasks where the correct answers are not known and the feedback is highly sporadic Moriarty and Miikkulainen (1997); Moriarty et al. (1999). Given the promising results in the handwritten character recognition domain, confidence evolution should work well in such tasks. For example, imagine applying confidence to the training of a robotic controller. Each neural controller in the population would be presented with an encoding of the robot's sensory input, and it would output a motor action and a confidence level. The action recommended by the most highly confident controller would be selected. After several decisions were made, a fitness evaluation for the whole sequence of decisions would be obtained. This fitness would then be distributed to the controllers according to how confident they were of their outputs and how often they were selected.

Although at first it seems that such fitness information might be too noisy, the situation is very similar to those of SANE and ESP neuroevolution methods Moriarty and Miikkulainen (1997); Moriarty (1997); Gomez and Miikkulainen (1997, 1999), where populations of neurons are evolved to form good neural networks. Each neuron receives a fitness based on how well the whole neural network performed in the task: in effect, the neurons are evolved to speciate into useful subtask that work well together. In reinforcement learning tasks with confidence evolution, similarly each network is rewarded based on how well the whole population did in the sequence of decisions. Given how powerful the SANE and ESP methods are, this same approach should also work well in confidence evolution.

In other learning tasks, it may be useful for an agent to express its confidence in a more direct form, by answering a more specific question about its performance. For example, a neural robotic controller might estimate the amount of time needed to reach a goal state, rather than estimating its probability of success. Such fitness functions might lead to more powerful evolution. Similarly, instead of always selecting the most confident individual's answer, the answer might be constructed by combining answers of the most fit individuals, or by at least not considering the answers of those with the lowest fitness. This method would for example solve the problem that occurred with crowding in the above experiments.

Different techniques of training individuals to output confidence could also be considered.

Evolving networks to provide answers and confidence estimates is clearly a more difficult task than simply evolving networks to provide answers. Might this increased difficulty be offset by using a combination of evolution and learning, or by Lamarkian evolution? Might it be beneficial to evolve the confidence neuron or network in a separate phase of evolution? Or perhaps as a separate network entirely? These are some of the issues that will be explored in future work.

7 Conclusion

This paper shows how the knowledge and expertise encoded by the entire evolved population can be utilized to come up with more accurate performance. High-quality decisions may be extracted from the population if a fitness evaluation rewards individuals that accurately output estimates of the quality of their decisions. To use this technique, a diverse population, capable of producing many different correct answers, is needed. This research motivates the development of techniques to ensure a high level of diversity throughout evolution, possibly even at the expense of overall fitness. The technique of confidence may have broad applicability in the domain of reinforcement learning tasks, which is the main direction of future work.

References

Choe, Y., Sirosh, J., and Miikkulainen, R. (1996). Laterally interconnected self-organizing maps in hand-written digit recognition. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pp. 736–742. Cambridge, MA.

Darwen, P. and Yao, X. (1996). Every niching method has its niche: Fitness sharing and implicit sharing compared. *Parallel Problem Solving from Nature*, pp. 398–407.

Floreano, D. and Urzelai, J. (2000). Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, 13:431–443.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342.

Gomez, F. and Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Denver, CO. Morgan-Kaufmann.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

Kephart, J. O. (1994). How topology affects population dynamics. In Langton, C. G., editor, *Artificial Life III*, volume XVII. Addison-Wesley, Reading, MA.

Mahfoud, S. (1992). Crowding and preselection revisited. *Parallel Problem Solving from Nature*, 2:27–36.

Moriarty, D. E. (1997). *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis, Dept. of Computer Sciences, The University of Texas at Austin. Technical Report UT-AI97-257.

Moriarty, D. E. and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive co-evolution. *Evolutionary Computation*, 5:373–399.

Moriarty, D. E., Schultz, A. C., and Grefenstette, J. J. (1999). Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:199–229.

Mühlenbein, H. (1991). Evolution in time and space - the parallel genetic algorithm. In Rawlins, G., editor, *Foundations of Genetic Algorithms*. Morgan-Kaufmann.

Social Phenomena Emerging by Self-Organisation in a Competitive, Virtual World ('DomWorld').

Charlotte K. Hemelrijk
Department of Information Technology and
Anthropological Institute and Museum,
University of Zürich, Switzerland
hemelrij@ifi.unizh.ch

Abstract

This essay explains how spatial structuring among competitive agents in an agent-based model leads to all kinds of behavioural patterns that are not represented in the behavioural rules of the agents. These phenomena resemble those observed in the real world, and thus, the model provides us with explanations and hypotheses that are more parsimonious than those of the conventional 'top-down', rationalistic view. This I show by means of five examples of a model called 'DomWorld' in which agents inhabit a virtual, homogeneous world and are equipped only with rules for two activities, namely to group and to perform competitive interactions. The first example demonstrates the origination of reciprocation of help in fights (yet the agents lack all moralistic feelings and have no motivation to help others in fights), and the second shows reduction of aggression when agents get familiarised with others (yet agents lack all internal mechanism to do so). The third reveals the phenomenon of spatial centrality of dominant agents (yet agents have no preference for any location within the group). The fourth example demonstrates the increase of dominance reversals between the sexes when grouping is cohesive (yet the agents lack all coalitionary tendency) and the fifth shows increased male 'tolerance' during periods of sexual attraction (in spite of the absence of any motivation to exchange).
In sum, spatial positioning and dominance interactions among nearby, simple agents may lead to interesting social phenomena by self-organisation.

1 Introduction

Patterns of social behaviour may arise by self-organisation through self-reinforcing interactions. Such reinforcing interactions are observed everywhere, even in traffic jams (Resnick 1994).

Nevertheless, only very few studies address the question how social behaviour may arise by self-organisation. The reason is that this approach runs counter to the usual explanation according to which each separate behavioural act is supposed to be genetically inherited or intentional. This disregard of the process of self-organisation is a consequence of the so-called 'top-down' approach, in which investigators first observe behaviour, and then subdivide it into independent traits, for each of which they draw up separate explanations. Because different researchers tend to investigate different features, possible interconnections among traits are overlooked. Thus, the top-down procedure keeps our attention away from emergent patterns that arise from the complicated feedback between interrelated variables. Feedback mechanisms are best understood by using the opposite route, the 'bottom-up' modelling approach. In such models we emphasise processes and dynamics. These models consist, for instance, in an artificial world inhabited by virtual creatures that interact with nearby others and are equipped with a minimum of 'realistic' rules. To detect the regularities that materialise from the model, we need behavioural definitions and categories similar to those used in ethological studies of real animals. By tracing patterns of interactions over time, we discover how social structure unfolds by self-organisation from a complex feedback process

between characteristics of individuals, their interactions and their developing social organisation. Structures that originate in this way are neither intentional nor genetically inherited (e.g. see Hogeweg 1988).

I shall illustrate this with five examples that arise in my model from the feedback between the spatial positioning of group-living, virtual agents and their self-reinforcing competitive interactions. These emergent effects are (1) reciprocation of support in fights, (2) decline of aggression, (3) spatial centrality of dominants, (4) female dominance over males and (5) male 'tolerance' of females. These examples show how the result of the process of self-organisation leads to more parsimonious hypotheses than the 'top-down' procedure. It is, therefore, important to supplement conventional studies on social behaviour with the approach explained in the present paper.

2 The Dominance World (DomWorld)

A summary of the model may suffice; for a more complete description see Hemelrijk (1999b; 2000). The model is inspired by that of Hogeweg (1988). It is based on only a small number of essentials of social life: It consists in a homogeneous virtual world inhabited by agents that are provided with only two tendencies: 1) to group (right half of Figure 1) and 2) to perform dominance interactions (left part of Figure 1). Why agents group (whether this is to avoid predators or because of resources being clumped) is not specified and irrelevant to the model. The same holds for dominance interactions. They reflect competition for resources (such as food and mates), but these resources are not specified. A dominance interaction takes place only if another agent is very near, in its personal space (PersSpace). The likelihood that an agent initiates an aggressive interaction increases with its chance to defeat its opponent (Hemelrijk 2000). The agent's capacity to be victorious (i.e. its dominance) depends, in line with ample empirical evidence, on the self-reinforcing effects of winning (and losing): Winning a fight increases the probability of winning the next time. After victory the dominance value of the victorious agent increases and that of the defeated one is reduced by the same amount. When, unexpectedly, an agent defeats a higher-ranking opponent, the dominance values of both opponents are changed with a greater amount than when the agent, as expected, conquers a lower-ranking opponent (conform to detailed behavioural studies on bumble bees by Honk & Hogeweg 1981). In this way the model allows for rank reversals. After a fight, the winner chases the opponent and the defeated agent flees.

The sexes differ only in fighting capacity: 'Male' agents start with a higher initial dominance value and are characterised by a higher intensity of attack than 'female' agents. To show the effects of winning and losing as clearly as possible, all agents of the same sex are identical at the start. Groups consist usually of 5 males and 5 females. The behaviour of the agents is analysed by means of behavioural units and statistical methods similar to those used for observing real animals.

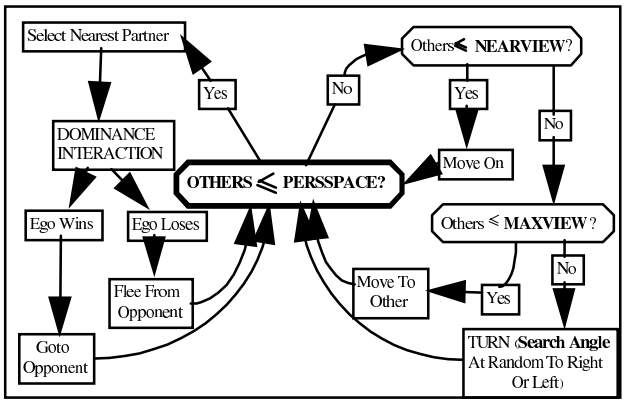


Figure 1: Flow chart for the behavioural rules of agents that are not attracted to another type (sex)

3 The emergence of reciprocation of support in conflicts.

In the real world, individuals sometimes join in fights between two opponents. This is called 'supporting' behaviour (behaviour of individual C in Figure 2) and it seems sometimes to be reciprocated. From observing such 'reciprocation' among primates, investigators conclude that 'moralistic' intentions and record keeping of acts given and received underlie this behaviour (de Waal 1982). In DomWorld, however, reciprocation of support in fights arises in spite of the fact that agents are unable to perceive and return debts and lack all motivation to help others and all ability to keep records of acts (Hemelrijk 1996).

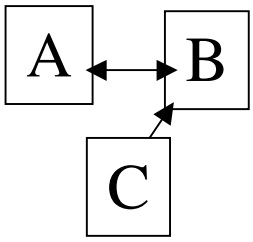


Figure 2: Schematic representation of 'support': 1) fight between A and B, next 2) C attacks B. The behaviour of C is interpreted as 'support' for A.

Instead, the so-called supporting behaviour and its reciprocation arises from the spatial configuration of the agents (Hemelrijk 1997). When one of two nearby agents (say 'A') chases away a third agent ('B'), the third one may by accident come within the range of attack of the nearby agent ('C') and is then attacked by C. In this way agents 'A' and 'C' take turns in chasing away the same victim 'B' and thus 'reciprocate' each other's 'support'. Because co-operating agents are more disturbed and distracted by others in dense than in loose groups, reciprocation occurs less often and bouts of alternating 'support' are shorter in dense than loose groups. That crowding prevents cooperation is not very remarkable, but looseness of groups is itself the result of a parameter setting for a large domain of aggression. Thus, the counter-intuitive, unexpected conclusion from the virtual world - and a readily testable, hypothesis for the real world - is that more aggressive agents are more co-operative.

Such co-operation by turn taking, as found in the model, is, from the point of view of an observer, similar to the game theoretical 'Tit-for-Tat Strategy' (Axelrod & Hamilton 1981). However, this similarity invalidates rather than supports the assumptions behind this theory. In the game theoretical framework co-operation is assumed to evolve on the basis of arbitrary pay-off (in terms of fitness). Cooperation is studied as an isolated feature, unconnected to other behavioural activities, despite the common-sense knowledge that natural selection operates on complete individuals and not on separate traits. The present essay, in contrast, neither deals with evolutionary processes, nor with pay-offs from cooperation. Instead, cooperation is viewed as a direct consequence of the intertwined effects of local competition and the gathering of agents.

4 Decrease of aggression

In many animal species, it is often seen that, when unfamiliar individuals are put together, initially dominance relations are unclear and aggression at first flares up and then, when a dominance hierarchy becomes manifest in the end, it declines. Traditionally, the function of the dominance hierarchy is supposed to be reduction of aggression and this decline has been supposed to be an adaptation based on an internal mechanism meant to reduce aggression as soon as dominance relationships are clear (e.g. see, Pagel & Dawkins 1997). On the other hand, this notion contradicts the view that dominance is beneficial in terms of acquisition of food, mates and safety, and that, therefore, individuals should aggress others wherever possible (especially when risks are low) in order to obtain and maintain high rank. Comparing in DomWorld the effects of rules in the agents based on these functions, the model reveals that aggression declines and dominance hierarchies develop for both rule sets! Thus, aggression declines even without rules that specify its

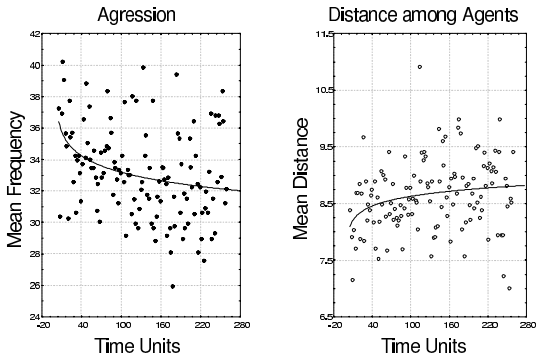


Figure 3: Mean frequency of aggression and mean distance among agents over time. (Logarithmic line fitting.)

diminishment (Hemelrijk 2000). It arises if hierarchy is steep, because low ranking agents become permanent losers. As a consequence, they flee further and further from others and therefore, meet others less often, and this reduces competitive interactions (1,2 in Figure 4).

Note that the presence of rules for risk sensitive aggression not only affects group density, but also spatial structure of individuals with different ranks in the group.

5 Spatial Centrality of Dominants

Usually, in many animal groups, there is a spatial structure in which dominants are in the centre and subordinates at the periphery. This spatial structure is explained by the influential 'selfish herd' theory of Hamilton (1971). The fundamental assumption of this theory is that individuals are better protected against predators in the centre of a group and therefore, have evolved a preference for positions in which conspecifics are between them and the predator, the so-called 'centripetal instinct'.

However, such a spatial structure with dominants in the centre also emerges in DomWorld when agents are attacking others always or especially when risks are low, (but not if they are supplied with rules to reduce aggression once dominance relationships are clear, see Hemelrijk 1998; 2000). The social configuration arises without any 'centripetal instinct' or fitness advantage. It is an automatic result of the mutually reinforcing effects of spatial structure and strong hierarchical differentiation and can be explained as follows. Pronounced rank differentiation causes low-ranking agents to be chased away by many and therefore to end up at the periphery (5 in Figure 4). This automatically leaves dominants in the centre.

Besides, agents of adjacent dominance are treated similarly by others and this causes agents to remain close and interact mainly with partners adjacent in rank; This spatial structure also reinforces the hierarchy, because if a rank reversal between two opponents occurs, it is only a minor one. This stabilises the hierarchy and it maintains the hierarchical differentiation (6 in Figure 4) and consequently also the rank-ordered, spatial structure (4 in Figure 4).

Interestingly, spatial centrality of dominants does not emerge if agents are supplied with rules that diminish their attack rate once dominance relations are clear, because in such a case agents tend to remain non-aggressively close to those that are distant in rank instead!

6 Inter-sexual Dominance

In most mammals, males are larger than females and dominant over them. Of two related chimpanzee species (the common and the pygmy chimpanzee), despite their similar sexual dimorphism in body size, the pygmy chimpanzee is known to show greater female dominance over males. Usually, this is attributed to a stronger coalitionary tendency among pygmy females to oppress males;

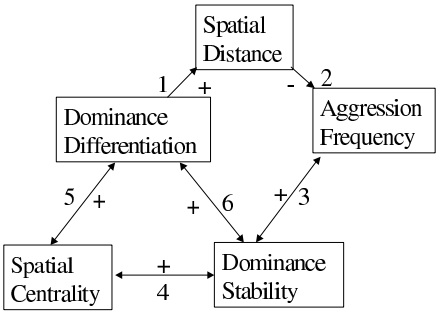


Figure 4: Summary of spatial-social structuring among agents provided with rules of risks-sensitive attack.

yet such an innate difference has so far not been established. Besides, it is clear that both species group very differently: pygmy chimpanzees group more cohesively than common ones. That cohesion may cause female dominance directly is shown in our DomWorld (Hemelrijk 1999a). Here, not only the high frequency of interaction in cohesive groups, but also the marked spatial constraints appear to enhance the mutual reinforcement between the development of the hierarchy and the spatial structure. Thus, a steeper hierarchy and clearer spatial centrality develop in cohesive groups. Consequently, the hierarchy of both males and females differentiates stronger and this automatically implies that although females initially rank lower than males, the differentiation causes certain females to become dominant over some males.

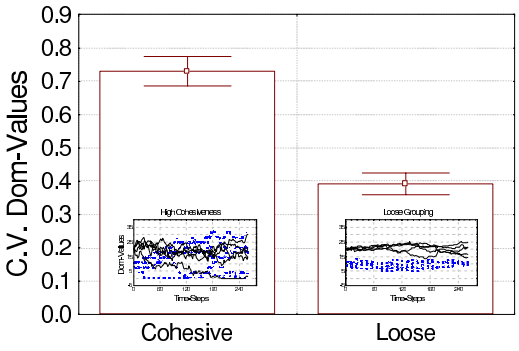


Figure 5: Differentiation of the dominance hierarchy (measured by the coefficient of variation of dominance values) in cohesive and loose groups. Small windows in each bar display the typical differentiation of dominance values for individual males and females. Dotted lines reflect values for females, continuous lines represent those for males.

7 Male 'Tolerance'

In many animal species, males are extremely attracted to females, whereas females are relatively uninterested in males (Trivers 1972). Also in primates, males are the ones who actively maintain proximity to females when females are in their sexually attractive, fertile period (e.g. see Hill 1987). This sexual asymmetry is understandable, because males can fertilise many females, whereas females get fertilised only once per reproductive period. To obtain access to females, males have been supposed to develop many strategies. In primates, females develop a pink swelling during the period in which they can be fertilised, and during this period, but not when females are flat,

males are observed to allow females priority of access to food sources (e.g. see Goodall 1986). This is regarded as an intentional manipulation by the male and as an adaptive exchange of favours, namely priority of access to food for females in exchange for copulation for males (Tutin 1980; Stanford 1996). Evidence for such exchange in terms of the number of offspring is, however, very limited, if existing at all (Hemelrijk et al. 1999). Yet, there is an increase in male tolerance towards females during tumescence and this needs an explanation.

The model provides us again with a new explanation for this 'tolerance'. Upon implementing 'sexual attraction' of males to females as preferential male orientation towards females rather than towards males, it appears that this addition increases female dominance over males! This arises due to the inbuilt mechanism that unexpected victories and defeats cause a greater change in the dominance values of both opponents than expected outcomes do and that the opportunities for these unexpected events increase due to the higher frequency of interaction between the sexes.

As a consequence of their increased dominance, females display more often aggression to males and males display more 'tolerance' of females, that is, they approach females more often non-aggressively. However, instead of male 'tolerance', a better name for it is male 'timidity' and obviously, this is not a manipulative strategy!

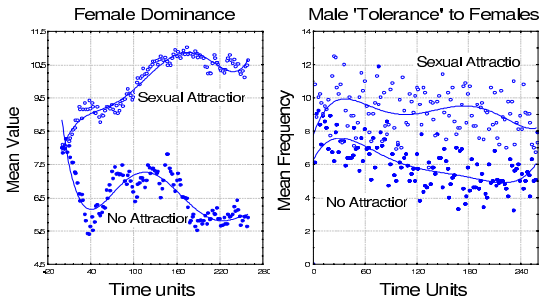


Figure 6: Mean female dominance and mean male tolerance (=non-aggressive proximity) to females over time. (Logarithmic line fitting).

8 Discussion

In summary, following the 'bottom-up' approach in a model with a high potential for self-organization, the integration of effects as regards spatial structure and social behavior leads to alternative (parsimonious) explanations for (1) reciprocation of support, for (2) aggression decline, for (3) spatial centrality of dominants, for (4) differences between species in inter-sexual dominance and for (5) increased male 'tolerance' of females during sexual attraction. Thus, the 'bottom-up' model forces upon us the many side effects that arise from grouping and self-reinforcing dominance interactions. The importance of such alternatives cannot be overestimated.

For instance, despite numerous studies on spatial centrality of dominants, no evidence for the supposed 'centripetal instinct' (Hamilton 1971) has been found so far (Krause 1993; Krause 1994). Further, spatial centrality is also observed among organisms that are not menaced by predators, such as the hammerhead shark (Klimley 1985). All this the model explains and, in addition, it explains the possible occurrence of marked differences in spatial structure between related species with similar predator pressure, but different cohesion or intensity of aggression.

As regards the evolution of societies, a final example may be given. Usually, in studies of animal behaviour, two types of societies are distinguished, egalitarian and despotic ones (Vehrencamp 1983). In certain primate species, such as macaques, a steep hierarchy characterises a despotic, and a weak hierarchy an egalitarian society. Furthermore, both societies differ in a number of other characteristics, such as bidirectionality of aggression, cohesion of grouping. For each of these (and other unmentioned) differences a separate adaptive explanation is usually given, but DomWorld provides us with a more parsimonious explanation for many of them at the same time:

By simply increasing the value of one parameter, viz. intensity of aggression (in which egalitarian and despotic macaques differ Thierry 1990), it appears that our artificial society switches from a typical egalitarian to a despotic society with a cascade of consequences that resemble characteristics observed in macaques (Hemelrijk 1999b), such as more asymmetrical aggression, looser grouping and more rank-correlated social behaviour. In addition, egalitarian and despotic virtual societies differ in spatial centrality of dominants and in female dominance over males. This makes it interesting to study these aspects in both societies of macaques. Such resemblance implies that differences between egalitarian and despotic societies of macaques may be entirely due to one trait only, intensity of aggression. At an evolutionary time-scale, one may imagine that species-specific differences in aggression intensity arose, e. g. because in the distant past in some populations of a common ancestor individuals suffered more food shortage than in others. Consequently, in those populations, intense aggression increased the chance of survival. Automatically, all other differences between both societies emerged from self-organisation and this makes it is unnecessary to look for separate adaptive explanations for each difference between a despotic and egalitarian society.

In summary, our experiments show how the self-organising perspective used in DomWorld leads to more simple explanations. By following the opposite 'top-down' approach, such explanations cannot be reached.

Acknowledgments

I am grateful to Rolf Pfeifer and Bob Martin for continuous support and to Jaap Hemelrijk for correcting the English. This work is supported by a grant from the Marie Heim-Vögtlin Foundation from the Swiss National Science Foundation and one from the A. H. Schultz foundation.

References

Axelrod, R. & Hamilton, W. D. (1981). The evolution of cooperation. Science 211, pp. 1390–1396.

de Waal, F. B. M. (1982). Chimpanzee Politics: sex and power among apes. New York: Harper and Row.

Goodall, J. (1986). The chimpanzees of Gombe: patterns of behaviour. Cambridge MA and London: Belknapp Press of Harvard University Press.

Hamilton, W. D. (1971). Geometry for the selfish herd. In *Journal of theoretical Biology 31*, pp. 295–311.

Hemelrijk, C. (1998). Risk Sensitive and Ambiguity Reducing Dominance Interactions in a Virtual Laboratory. In *Proceedings of the Fourth International Conference on Simulation on Adaptive Behavior. From Animals to Animats 5* (ed. R. Pfeifer, B. Blumberg, J.-A. Meyer & S. W. Wilson), pp. 255–262. Zürich: MIT Press.

Hemelrijk, C. K. (1996). Dominance interactions, spatial dynamics and emergent reciprocity in a virtual world. In *Proceedings of the fourth international conference on simulation of adaptive behavior*, vol. 4 (ed. P. Maes, M. J. Mataric, J-A Meyer, J Pollack & S. W. Wilson), pp. 545–552. Cambridge, MA: The MIT Press.

Hemelrijk, C. K. (1997). Cooperation without genes, games or cognition. In *4th European Conference on Artificial Life* (ed. P. Husbands & I. Harvey), pp. 511–520. Cambridge MA: MIT-Press.

Hemelrijk, C. K. (1999a). Effects of cohesiveness on intersexual dominance relationships and spatial structure among group-living virtual entities. In *Advances in Artificial Life. Fifth European Conference on Artificial Life*, vol. 1674 (ed. D. Floreano, Nicoud, J-D., Mondada, F.), pp. 524-534. Berlin: Springer Verlag.

Hemelrijk, C. K. (1999b). An individual-oriented model on the emergence of despotic and egalitarian societies. newblock In *Proceedings of the Royal Society London B: Biological Sciences. 266*, pp. 361–369.

Hemelrijk, C. K. (2000). Towards the integration of social dominance and spatial structure. Animal Behaviour 59, pp. 1035–1048.

Hemelrijk, C. K., Meier, C. M. & Martin, R. D. (1999). 'Friendship' for fitness in chimpanzees? Animal Behaviour 58, pp 1223–1229.

Hill, D. (1987). Social relationships between adult male and female rhesus macaques: 1. Sexual consortships. Primates 28, 439-456.

Hogeweg, P. (1988). MIRROR beyond MIRROR, Puddles of LIFE. In *Artificial life, SFI studies in the sciences of complexity* (ed. C. Langton), pp. 297–316. Redwood City, California: Adisson-Wesley Publishing Company.

Honk, C. v. & Hogeweg, P. (1981). The ontogeny of the social structure in a captive Bombus terestris colony. Behavioral Ecology and Sociobiology 9, 111–119.

Klimley, A. P. (1985). Schooling in Sphyrna lewini, a species with low risk of predation: a non-egalitarian state. Zeitschrift fur Tierpsychology 70, 279-319.

Krause, J. (1993). The effect of 'Schreckstoff' on the shoaling behaviour of the minnow: a test of Hamilton's selfish herd theory. Animal Behaviour 45, 1019-1024.

Krause, J. (1994). The influence of food competition and predation risk on size-assortative shoaling in juvenile chub (Leuciscus cephalus). Ethology 96, 105-116.

Pagel, M. & Dawkins, M. S. (1997). Peck oders and group size in laying hens: 'future contracts' for non-aggression. Behavioural Processes 40, 13-25.

Resnick, M. (1994). Turtles, termites and traffic jams. Explorations in massively parallel microworlds. Cambridge, MA: MIT Press.

Stanford, C. B. (1996).The hunting ecology of wild chimpanzees: Implications for the evolutionary ecology of Pliocene hominids. American Anthropologist 98, 96-113.

Thierry, B. (1990). Feedback loop between kinship and dominance: the macaque model. Journal of theoretical Biology 145, 511-521.

Trivers, R. L. (1972).Parental investment and sexual selection. In Sexual selection and the descent of man (ed. B. Campbell), pp. 136-179. Chicago: Aldine.

Tutin, C. E. G. (1980). Reproductive behaviour of wild chimpanzees in the Gombe National Park. Journal of Reproduction and Fertility Supplement 28, 43-57.

Vehrencamp, S. L. (1983). A model for the evolution of despotic versus egalitarian societies. Animal Behaviour 31, 667-682.

Spatio-temporal Processing of Musical Texture, Pitch/Tonality and Rhythm*

Marc Leman
IPEM - Dept. of Musicology, Ghent University
Blandijnberg 2, B-9000 Ghent, Belgium
Marc.Leman@rug.ac.be

Abstract

This paper gives an overview of some computer modeling experiments in the areas of texture analysis, pitch perception, tonality induction, and rhythm recognition. Different memories types are explored using brain-like computation in which conceptualization takes place. At the end a general framework is presented for the integration of different memory components.

1 Introduction

Music cognition and, more in general, the mechanisms whereby music is perceived and understood have challenged music theorists for many centuries. From the 1970ies on, music perception and cognition has been approached from the point of view of computer modeling. Attempts were undertaken to deal with musical information processing in a *dynamic* way, with the aim to simulate and better understand human intelligent musical information processing.

Many concepts were borrowed from linguistics and Artificial Intelligence to build theories based on a symbolic approach (Monelle, 1992; Minsky, 1982). The epistemic entities on which these theories were built described notes, chords and other musical features by means of symbolic names, used in inductive and deductive reasoning processes usually based on rules. Although these approaches contributed to better insights in music representation and manipulation, they failed to catch those aspects of musical information processing that related to sound, and subsequent processes in auditory perception and low-level perception. Those aspects were brought to the music theorists' attention by the developments in other disciplines such as sound and music synthesis (Risset, 1992), psychoacoustics (Schouten, 1940; Plomp, 1966) experimental psychology (Shepard, 1984; Krumhansl and Kessler, 1982), brain research (Schreiner and Langner, 1988), auditory modelling (Meddis and Hewitt, 1991; Van Immerseel and Martens, 1992), and, more recently, cognitive neuromusicology (Tervaniemi and Leman, 1999).

The availability of more powerful computing resources allowed to test working hypotheses of real musical perception with computer simulation models. In musicology, a modeling approach to music perception and cognition was conceived starting from the observation that the nature of the musical environment needs to be defined in terms of the physical properties of the sound and not in terms of its symbolic representations (Leman, 1989, 1995b, 1997, 1999). It was argued that real information content of a musical signal cannot be fully captured using symbolic procedures because these describe objects at a meta-level, with the consequence that compelling causal forces inherent in the object-level of the perception and cognition of physical objects cannot be directly taken into account at the meta-level. A central point of the *naturalist* or *sub-symbolic* approach concerned the transformations of a musical sound into auditory images and its inherent (musically) compelling constraints. The point of departure is the continuous, sub-symbolic and ephemeral acoustic substrate and the neurophysiological workings of audition together with the self-organizing

*Paper presented at Workshop Internalising Knowledge, Ieper, November 22-24, 2000

behaviour of neuronal assemblies which transforms this substrate into relative stable knowledge-schemata.

In this paper, an overview is given of memory models, that have been explored for the conceptualization of musical features of acoustical signals. The basic assumptions of our approach are first explained. Then the role of immediate memory, short term memory, long term memory, statistical and episodical memories is discussed and a general framework is proposed for dealing with brain-line computation models in connection with symbolic reasoning.

2 Basic Assumptions: Ecology and Naturalism

The underlying assumptions may be stated as follows. Music perception is to a large extent context-dependent, non-denotational, and dependent on cultural factors. Perceptual/cognitive representations are treated as auditory images, rather than as symbols, and they reflect the properties of the musical environment.

Different types of auditory images reflect the fact that the musical signal evolves into different representations at different levels of abstraction. The images that best mirror the constraints of the outer environment develop at the level of sensory information processing. At higher perceptual and cognitive levels, images may be completed and transformed into more abstract spatio-temporal representations that involve learning principles.

When designing a computer model, we want to specify the conditions on which the interactions between the physical and musical environment on the one hand and the model of human information processing on the other hand can take place, are defined. No relationships among musical entities are pre-defined, as is the case with symbol-based approaches. By mere exposure to musical sounds the model should be able to extract, through processes of self-organization that mimic the capacity of certain brain structures to adapt to stimuli and identify and retain the information content of the input, the musical content of a particular audio signal. This approach is called *ecological* because the processing system and environment are both taken into the model. The approach is *naturalistic* in the sense that methods from the natural sciences are adopted to study phenomena of the musical mind. The models developed can be said to have a functional equivalence with the neuron-based information processing. Hence, auditory modeling, and audio image processing, neural networks, and perceptually constrained spatio-temporal representations are core concepts in the approach (Leman, 2000b).

3 From images to the concept of schema

In the past, several computer simulations of context-dependent pitch perception and tonality induction have been based on the concept of *schema*. The concept appeared during the last two centuries in different contexts and with quite different meanings (Seifert, 1991).The term was used by Kant in 1781 in his philosophical constructs to denote a procedure relating concepts to data from the sensory domain. It was later used in clinical neurology around 1920 by Head and in the same area by Bartelett in 1932. In more recent times Neisser 1967 introduced a definition of "cognitive structures" or "schemata" for the memory stored information on earlier cognitive acts. With the progress of Artificial Intelligence research and the development of its representational formalisms in the 1970s/80s, the concept of schema was associated with internal representations within the mind/brain, but was also considered as a model of functioning of the mind and substantiated through data and algorithms. The contribution of Hebb to the definition of the theory of cell assemblies and of the related complex neuronal architecture opened the way to *connectionism* and to the vast domain of neural networks, while shifting the attention to neurology. A more general schema-theory, that tries to combine concepts from cognitive psychology, neurophysiology and AI was worked out recently by Arbib and others (Arbib, 1995).

The schema-theory that is here presented, tries to model and connect several of the latest findings in the above disciplines. The results of the simulations were confronted, whenever possible, with the results of similar psychological tests. Yet if we speak about conceptualization in music we should not limit ourselves to learning paradigms. It turns out that many features of music are

processed in the auditory periphery and that context-dependent information processing may be dealt with using short-term memories. Where and at what point schema-theory is really needed is still a matter of discussion but we believe that it is basically relevant when dealing with *style* concepts.

4 The perception of music

The various representations of the musical images, are inherently tied to the way the perception of sounds and the subsequent transformations of the musical signal at different levels of brain processing are understood and modeled. Given the complex nature of the auditory brain functions, theoretical modeling may represent a useful tool to test working hypotheses in a process of mutual refinement between the results of the simulations and the verification on the field. In a real listening situation the vibrations of the air molecules stimulate the ear and cause, through neuronal firing, the sensation of sound. The physiology of mechanical-to-neural transduction has been extensively studied and several auditory models exist.

Research in auditory physiology shows that temporal information is accurately transmitted to higher information centers but that the average frequency range of the represented temporal variations in the time patterns of neural responses of the auditory nerve is higher than in the inferior colliculus, which is still substantially higher than in any of several cortical fields. The rates of 1000 Hz that are found in the auditory nerve, are reduced to about 100 Hz in the inferior colliculus, and to about 10 Hz in the cortex (Langner and Schreiner, 1988). There is evidence (Eggermont, 1997) that higher brain functions operate a kind of time to place mapping. The mapping of neural time-code to neural place-code could be a central mechanism for the formation of a perceptual constrained spatio-temporal representational system, and could provide support to the hypothesis of the formation of topological maps, or schemata, that are the outcome of the model on tone center extraction later described.

5 Representation of auditory information with auditory images

The term auditory image is used in this context to denote a carrier of auditory information at selected levels of the brain information processing. Low level images, that are extracted at the level of the auditory nerve, are derived as the result of causal feature extraction processes and are strongly related to the original sound waves. They account for the auditory nerve spike train code or for the amount of discharges per time unit. Also phase locking synchrony at a particular frequency can be codified at this level. It is important to note that musical features, such as pitch, loudness, rhythm and timbre, that are precisely characterized with symbolic notation, are here implicitly contained in the neuronal representations (images), and can be extracted as emerging aspects of the underlying auditory processing with further processing of the auditory images.

Less detailed physiological information is available beyond the auditory periphery and, according to the actual knowledge of the higher auditory brain functions, a transformation from the temporal domain into spatio-temporal representations can be observed at higher levels. The spatio-temporal structures may entail low-resolution temporal and even motoric-temporal information. The coherence in the transformation of the images from low level brain processing to higher level ones can be assured by the causal pathways from sound to images, as process which in a certain way assumes an inherent logic. Physical constraints among neurons may define a limited spatio-temporal structure that allows a set of image transformations, involving operations in space and time. The representational system is thus assumed to define an inherent coherence of musical image transformations.

In the framework of this ecological theory the auditory images represent the information at different stages of brain processing, following the initial stimulus of sound waves. They are retained in memories which act as registers or stores of images. Several types of memories can be taken into account, with different characteristics and functions, whose role will be more precisely illustrated in the case studies:

- An immediate short-term memory that registers the activation of a set of neurons and whose contents are to be used immediately and that fade away in a few milliseconds.

- An echoic short-term memory that retains information for no longer than a few seconds and that plays an important role in music perception. The echo is here defined as the half decay time, which is shorter than a few seconds.

- A statistical long-term memory (SLTM) where the invariant features of a stream of auditory images are collected and stored.

- An episodic long-term memory (ELTM) that accounts for the temporal dependencies in music, such as the sequence of notes of a melody or a rhythmic pattern.

5.1 An immediate short-term memory for musical texture

The texture patterns can be conceived of as spatio-temporal patterns derived from auditory nerve patterns. The latter are closely connected to the sound patterns in the environment. The causal chain leading to the pitch images involves an Auditory Peripheral Module (APM):

$$APM: \; s(t) \rightarrow \tilde{d}(t) \; = \; < d_c(t) > \; for \; c = 1...C \tag{1}$$

where $s(t)$ represents the musical sound, and $d_c(t)$ the discharge pattern along a particular channel c of the auditory nerve. There are C such channels. An example is given in Fig. 1c. The horizontal axis represents time, and the vertical axis represents the auditory channels in terms of their frequency along a critical band scale. All models discussed below use this kind of transformation of the musical signal into representation of primary auditory images.

The texture pattern studied in the context of musical consonance and dissonance is called *roughness*. Since its introduction by H. v. Helmholtz von Helmholtz (1968), the terms *sensory dissonance* and *roughness* have both been used to characterize the texture of a sound in terms of *impure* or *unpleasant* qualities. The sensation is associated with the physical presence of beating frequencies and it relies on an immediate short-term memory.

Following v. Helmholtz, many researchers have used the term *sensory dissonance* when speaking about tone relationships but it is nowadays more appropriate to use the term *roughness* [German: 'Rauigheit', French: 'Rugosité'] because this term is more general and can be applied to characterize impure or unpleasant qualities of all kinds of sounds, including noises. (In dealing with the texture of noisy sounds it is indeed rather awkward to use the term sensory dissonance.)

We have modeled roughness using the notion of *synchronization*, a important concept in neurophysiology which indicates the degree of a neuron's total firing rate that is phase-locked to the corresponding stimulus component (Leman, 2000c). Below we apply SIM to a tone complex that defines different musical intervals of a timbre over one octave. The harmonic tone complex consists of a fundamental (f_0) at 500 Hz and 5 harmonics with equal amplitude. This tone is played together with a pitch shifted copy. The shift over 5 seconds is linear in frequency up to the upper octave (f_0 1000 Hz).

The model, apart from its good agreement with this theoretical model provides an additional cue in showing the 'spectral' (=excitation in the auditory channels) as well as 'temporal' (=synchronization index) factors that contribute to roughness, as shown in Fig. 2. The upper panel shows how the energy is distributed over the auditory channels, and the middle panel shows how the energy of the beating frequencies contributes to the roughness curve shown in the lower panel. According to the model, both the upper and middle panel lead to the same curve. The points of minimal roughness or sensory dissonance indicate a hierarchical order of intervals in terms of roughness. This hierarchy can be musically exploited as the points of minimum roughness may indicate candidates for a musical scale.

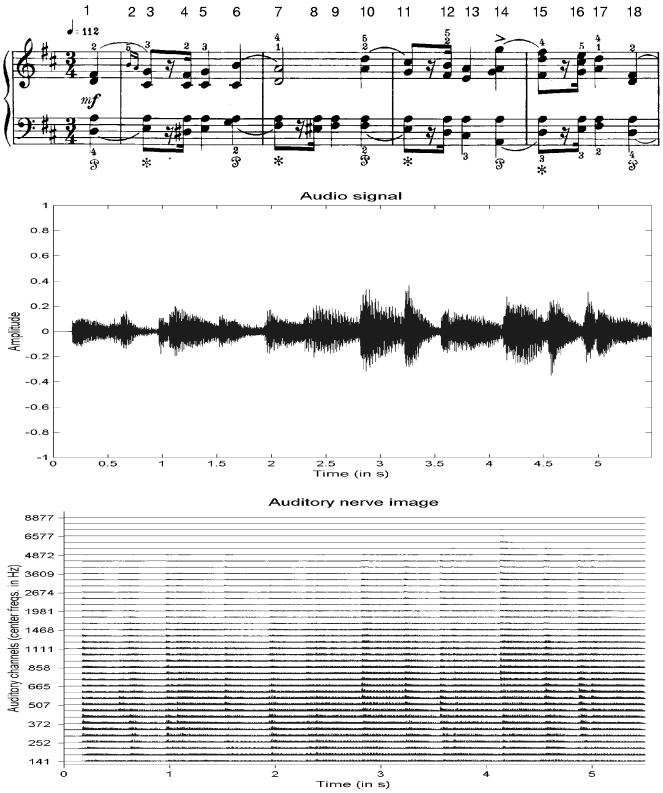


Figure 1: Transformation of a sound into auditory nerve images, an essential first step in music perception. (a) The score, (b) the sound represented as waveform, (c) the auditory nerve images (the vertical axis represents the auditory channels in terms of their frequency along a critical band scale)

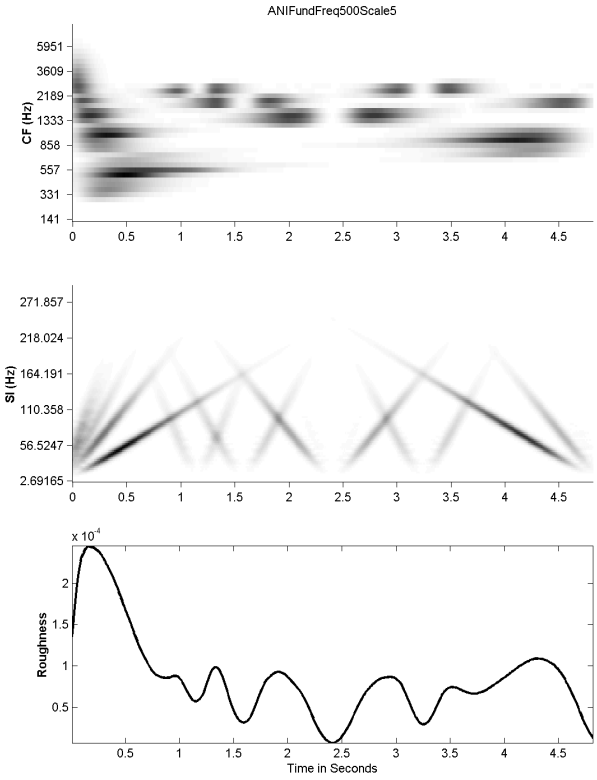


Figure 2: Roughness of a harmonic tone complex

5.2 An echoic short-term memory for context-dependent pitch perception

The pitch images can be described in terms of a transformation PCM (Pitch Completion Module) which implies a mapping from time-code to place-code. Thus $d_c(t)$ is a time-encoded image, while $\tilde{p}(t)$ is a place-encoded image. Both types encode the rate of neuronal discharges (see Leman (2000a); Leman et al. (2000) for more details).

$$PCM: \ \tilde{d}(t) \rightarrow \tilde{p}(t) \ = \ \sum_{c=1}^{C} \tilde{p_c}(t) \tag{2}$$

The echoic pitch images follow from a leaky integration of these images:

$$EMM: \tilde{p}(t) \rightarrow \tilde{p}_E(t) \tag{3}$$

The pitch images and echoic pitch images are shown in Fig. 3a-c.

An echoic short-term memory may account for certain aspects of context-dependent pitch perception. Although no schema is involved at this stage, it is important to study the effect of echoic short-term memories because they provide important information that may penetrate the schemata layers. An echoic short-term memory for pitch may build up pitch images into a short-term memory, lasting no longer than a few seconds. The echoic (short-term) memory is a very powerful representational device in music perception. Leman 2000a shows that two echoic memories for pitch, one operating with a half-time decay of 0.1 seconds, and the other one with a half-time decay of about 1.5 seconds, may account for the famous probe-tone rating experiments of Krumhansl and Kessler 1982. They had assumed that the task, involving the comparison of a tone with a previous tonal context, involves a long-term memory of tonal relationships. Our study

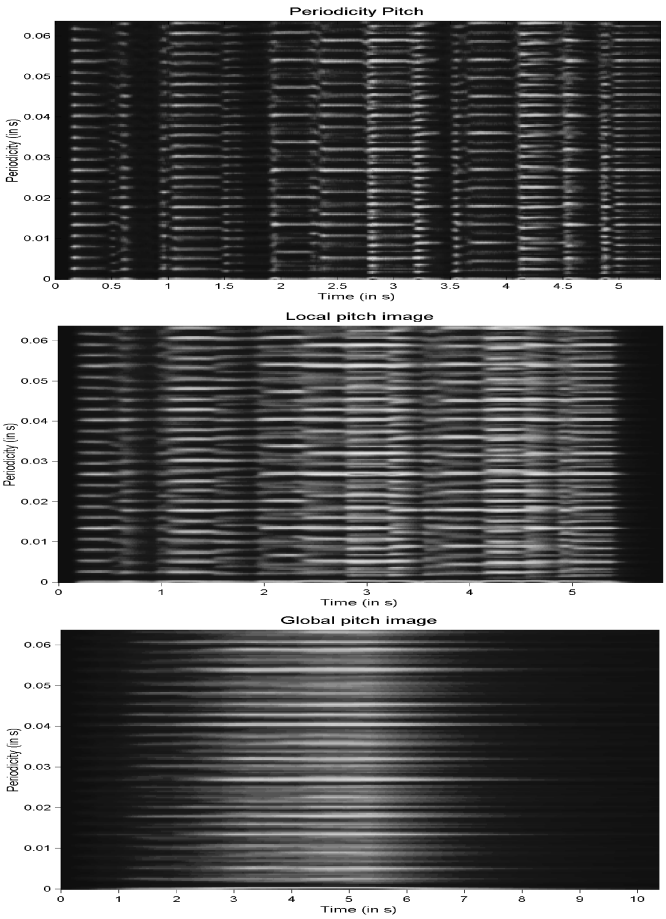


Figure 3: Pitch images, (a) periodicity pitch images, (b) echoic images using a half-decay value of 0.1s, (c) echoic images using a half-decay values of 1.5s. Observe that due to smearing of the pitch images, the time scale is slightly different

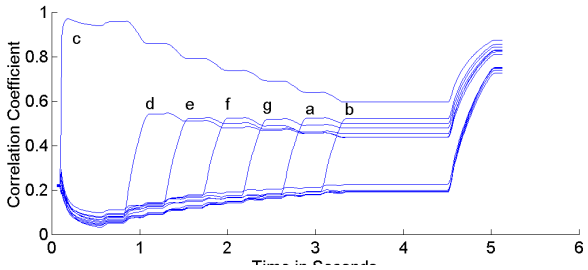


Figure 4: Extraction of the 'contextuality' feature which represents the degree in which a particular pitch contributes to a pitch profile which is considered a spatial representation of the scale

does not exclude the existence of a long-term memory for tonal relationships (see next paragraph) but it questions the need of this memory in the probe-tone task in particular. The results raise questions whether expectancy in probe-tone studies may be considered a genuine effect of imagery. It seems likely that tonal expectation is largely a stimulus-induced activity in which echoic memory plays an important role.

Figure 4 shows the the degree in which a particular pitch contributes to a pitch profile. Such a pitch profile could enter a long-term memory, but this cannot be determined using the probe-tone experiments.

5.3 A statistical long-term memory (SLTM) for tonal schema development

SLTM-learning can be described as:

$$SLTM_{learning}: \ \tilde{p}_E(t) \rightarrow \tilde{\tilde{P}} \ = \ < \tilde{P}_k > \ for \ k = 1...K \tag{4}$$

where $\tilde{p}_E(t)$ denotes a pitch image at time t, and $\tilde{P}_k$ a stable image in SLTM. The container of stable images $\tilde{P}_k$ is denoted $\tilde{\tilde{P}}$, and there are K such images (also called: classes). The pitch image is time integrated using a half-decay value, or echo, specified by E. The echo, which is assumed to be determined by neuronal integration mechanisms, accounts for the pitch context which may be short in the case of chords and long in the case of tone centers.

SLTM-resonance transforms the stimulus induced images into an activated response. This process can be described as:

$$SLTM_{resonance}: \ \tilde{p}_T(t) \bigotimes \tilde{\tilde{P}} \rightarrow \tilde{A}(t) \ = \ < A_k(t) > \ for \ k = 1...K \tag{5}$$

The operator $\bigotimes$ denotes some matching process such as correlation, or the retroactive dynamics as described (for key recognition) in Leman 1995a; 1995b. The resulting pattern $\tilde{A}(t)$ contains the degree of activation of each stable image $\tilde{P}_k$ in the schema. Each A_k at time point t represents a value that results from the matching of $\tilde{p}$ at time point t with $\tilde{P}_k$. The recognition trajectory is thus described by $\tilde{A}(t)$.

Statistical long-term models have been studied for several years now. At IPEM a model has been designed that produces an interesting example of the development of a topological structure, that captures the fixed inter-relationships among images, in a statistical long-term memory. The model is composed of an auditory front-end and a cognitive component based on a self-organizing neural network (Kohonen, 1995). Computer simulations were performed using as a sound input data set of 24 cadences in all 24 keys and, in a second test, the whole first book, preludes and fugues, of the Well Tempered Clavier by J.S.Bach. A harpsichord performance was used in the latter case and synthetized sounds, based on Shepard tones, in the former. The results of the simulations show that the model is able to autonomously develop on a bi-dimensional map a topological structure that recalls the circle of the fifths and that represents a perceptually constrained spatio-temporal

representation system on an array of neurons (Leman, 1995b; Leman and Carreras, 1997). This topology is similar to the mental representation maps proposed by Krumhansl and Kessler 1982. The statistical long-term memory may work in learning-mode, when it transforms the stream of data-driven images from an echoic memory into long-term stable images or in resonance-mode when it uses the representational structure as a resonance system for tone recognition. Once established, when the map is activated by a probe tone, a highly responding region will appear and neighboring regions will be activated as well, to denote the tonal inter-relationships among keys of the western tonal system and the importance of the topological distribution of tone centers on the map.

Echoic pitch images were obtained by processing the sampled music of 24 cadences in all keys and the first book of Bach's well Tempered Clavier and were used for training a long-term statistical memory in two distinct test cases. Here the LTSM was implemented with a self-organizing Kohonen neural network on a 100x100 grid of torus shape. The systems developed a map that represents the circle of the fifths. During the training process 24 stable images build up at precise points in the map and stay invariant with respect to further processing of input samples 5.

SLTM-learning instantiates in this case a categorization process where structural and time invariant information is extracted form instances of variant information (the echoic pitch images). The developed SLTM map can be used in resonance mode for tone recognition. A sound stimulus induces an activation response on the map with different degrees of activation of each of the 24 stable tone centers on the map. A recognition trajectory can so be traced, following a stream a sound samples.The recognition trajectory is thus described by $\tilde{A}(t)$, as shown in Fig. 6.

It is interesting to point out that the map presents a synoptic representation of the perception of the harmonic development of a musical excerpt, where various tone centers may be active at the same time with different strengths. A similar phenomenon, from a quite different perspective, can be observed in the chord decomposition experiment, later described, where a complex chord often decomposes into simpler chords with different fundamentals and different weights (Carreras et al., 1999).

Schemata can be used for practical recognition purposes. A chord recognition and decomposition model based on schema-theory has been worked out which starts from an analysis of the perceptual phenomenon of virtual pitch and a harmonic decomposition model, which relates the harmonic spectrum of a note to the first four different prime harmonics, for instance C-E-G-Bb for the note C. Each prime harmonic generates its own harmonic multiples that belong to the harmonic spectrum of the note. A set of the 132 chords was used as a base for training a neural network. Each chord was played using piano timbre and sampled for 200 ms. The sounds were processed into completion images and echoic pitch images. After training a SLTM a tesselation map emerges by self-organization, that shows how the best responding neurons to each of the 132 chords are distributed in a regular fashion, with chords related to the same fundamental inside connected map areas. The auditory periphery modules and the cognition modules are similar to the ones used for the 24 keys experiment. In addition, an onset detection module is used to separate the musical events of a piece of music that are to be presented to the SLTM when used in chord recognition mode. In this case the map is used in resonance mode. For each event the map activation is calculated and a set of candidate sub-chords, ranked according to their activation values, is produced. By choosing the first three sub-chords with different fundamental notes in the value list, the original chords are reconstructed. Chords of any type and complexity are here decomposed. Figure 7 shows a decomposition for Schumann's *Kuriose Geschichte*.

5.4 An episodic long-term memory (ELTM)

The build up of ELTM involves the Auditory Peripheral Module (Expression 5.1) which transforms a given sound $s(t)$ into neuronal patterns $d_c(t)$ at the level of the auditory nerve. This step is similar to the first step in SLTM-learning. It basically provides an analysis of the sound in different frequency bands such that a possible polyrhythmical pattern may be dealt with, provided that the different rhythms are in different frequency layers (which they often are in music). The next step,

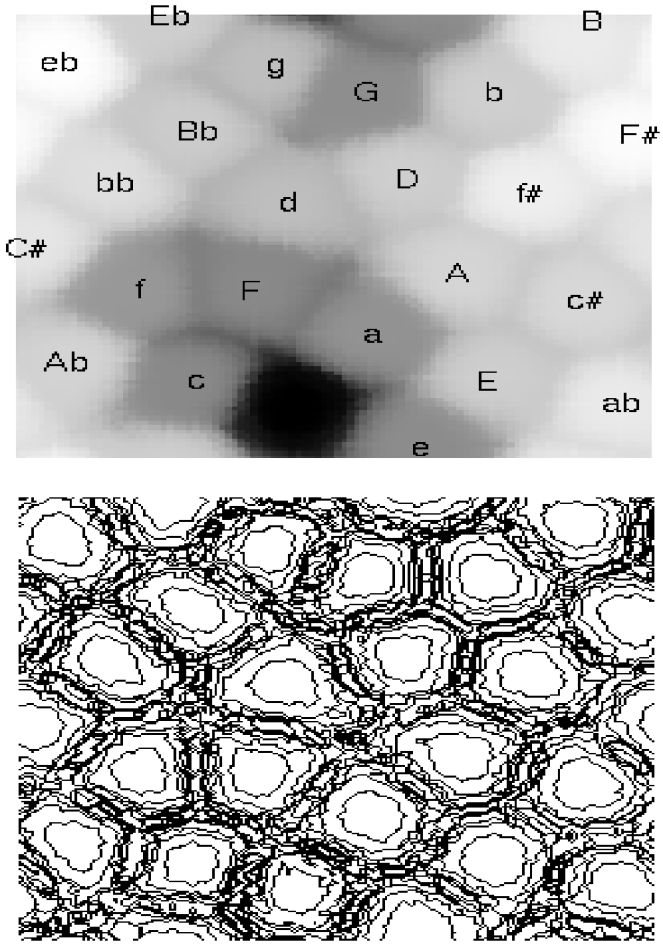


Figure 5: SLTM of learned tonal distributions. The top panel represents the output of SLTM to a cadence in C major. The neuronal carrier contains 10000 neurons ordered on a two-dimensional grid of 100 by 100 neurons. The structure is a torus (top and bottom are connected, as well as left and right). The black dots represent activation of neurons in response to the input. The labels are put on the places where the network gives a maximal response. In this case, the response region is centered on the black spot (which covers the label C). SLTM has been trained with 72 different cadences which got organized into 24 classes along circles of fifths. The lower panel gives an idea of the internal boundaries which define the class separations. The labels, representing the keys, are put on the neurons that give the best response to that particular key. The lower figure is obtained by visualization of the neuronal boundaries that develop by self-organization into the schema. The boundaries show the contours of the distinct key templates (see Leman and Carreras (1997))

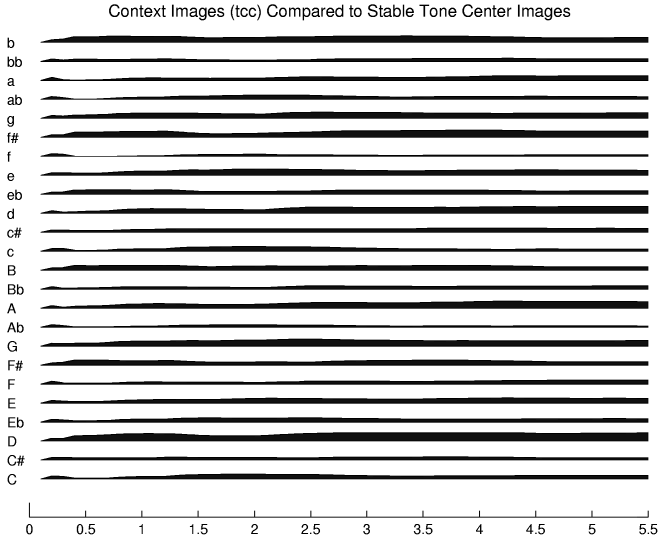
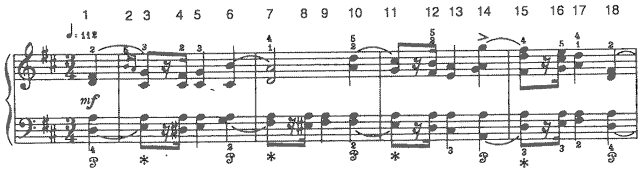


Figure 6: Example of a trajectory represented as a function of the invariant or stable images. The schema was learned by 72 different cadences (using Shepard tones). The input were the pitch images shown in Fig. 3c (Schumann) piece. The activations describe a trajectory on tone center images with respect to the topology




```
analysis file= schum1, threshold = 200
----------------------------------------
EVENT 1:onset time = 0.156000, energy = 145.510956:   f# a#              D
EVENT 2:onset time = 0.520000, energy = 16.096193:   d g# b              G
EVENT 3:onset time = 0.612000, energy = 39.294521:   c# e g a            A
EVENT 4:onset time = 0.960000, energy = 100.000000:   c# e f#           F#
EVENT 5:onset time = 1.076000, energy = 29.242847:   c# e g              A
EVENT 6:onset time = 1.516000, energy = 42.948112:   g g# b              E
EVENT 7:onset time = 1.940000, energy = 64.564095:   d f# a              D
EVENT 8:onset time = 2.296000, energy = 22.406389:   c f a               F
EVENT 9:onset time = 2.384000, energy = 23.174072:   f# a               D
EVENT 10:onset time = 2.816000, energy = 61.814308:   d f# a             D
EVENT 11:onset time = 3.216000, energy = 62.062859:   c# e a             A
EVENT 12:onset time = 3.560000, energy = 59.018021:   d d# f# g# a b     B
EVENT 13:onset time = 3.692000, energy = 13.763593:   e a                A
EVENT 14:onset time = 4.128000, energy = 76.476974:   g a a#             A
EVENT 15:onset time = 4.536000, energy = 68.283051:   d f# a             D
EVENT 16:onset time = 4.864000, energy = 91.117126:   c# e g             A
EVENT 17:onset time = 4.992000, energy = 34.103455:   d f#               D
```

Figure 7: Decomposition of Schumann's *Kuriose Geschichte*

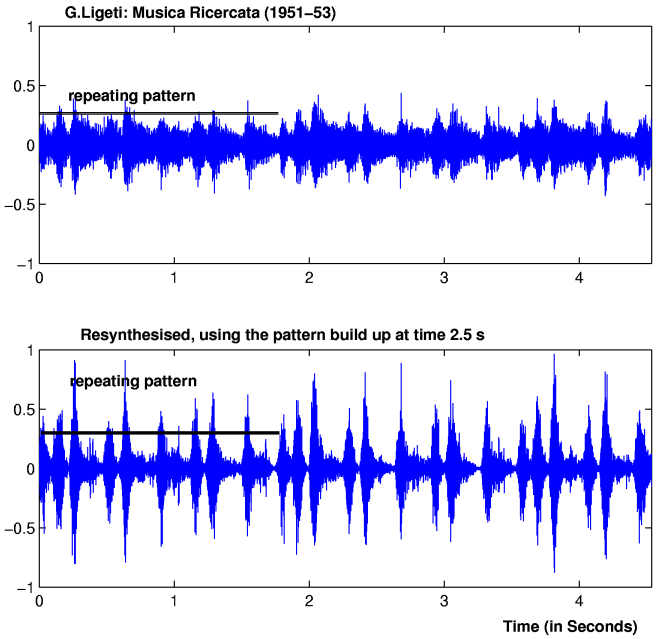


Figure 8: Original soundwave and resynthesised soundwave using ELTM. (a) Waveform representing an excerpt of the *Presto energico from the Musica Ricercate per pianoforte*(1951-53) by G. Ligeti, (b) Synthesised waveform based on the loop-patterns contained in ELTM after 2.5 seconds. The loop-patterns of the different auditory channels are repeated. The resulting modulation patterns are then multiplied with band-pass noises having the center frequency of the corresponding auditory channels

called Envelope Extraction(EE) extracts the energy envelope from the patterns $d_c(t)$ into $e_c(t)$. It entails a reduction in temporal resolution (i.e. lower *sampling rate*):

$$EE_c: \; d_c(t) \rightarrow e_c(t) \tag{6}$$

A Looping Analysis (LA) then transforms the neuronal pattern into a set of looping-patterns:

$$LA_c: \; d_c(t) \rightarrow \tilde{L}_c(t) \; = \; \vec{l}_{c\tau} \; for \; \tau = 1...T...\Theta \tag{7}$$

where $\tilde{L}_c(t)$ denotes a set of loops with different length τ, which goes from 1 to Θ. The next step builds up ESTM by choosing the best loop pattern and storing it into a memory.

$$ESTM_c : m_c(t) = \left[\tilde{L}_c(t)\right]_{best} \tag{8}$$

where $m_c(t)$ holds the memory with the best loop-pattern at time t.

Figure 8a is an example of a waveform representing an excerpt of the *Presto energico from the Musica Ricercate per pianoforte* (1951-53) by G. Ligeti (Bis-CD-53). The piece is characterized by a repeating rhythmic pattern which is indicated on the figure. Figure 8b shows the synthesized waveform based on the loop-patterns contained in ELTM after 2.5 seconds. A resynthesis of the original rhythmical pattern can be obtained by using band-limited noises covering the original frequency range of the auditory model, and then amplitude modulate the noises with the found rhythm patterns. Thus we have: $r_c(t)$ representing a band-limited noise signal with center frequency equal to the auditory channel c, a specific pattern m_c taken at 2.5 seconds which is repeated and which models the noise signal. The final signal $s'(t)$ is obtained by summing the

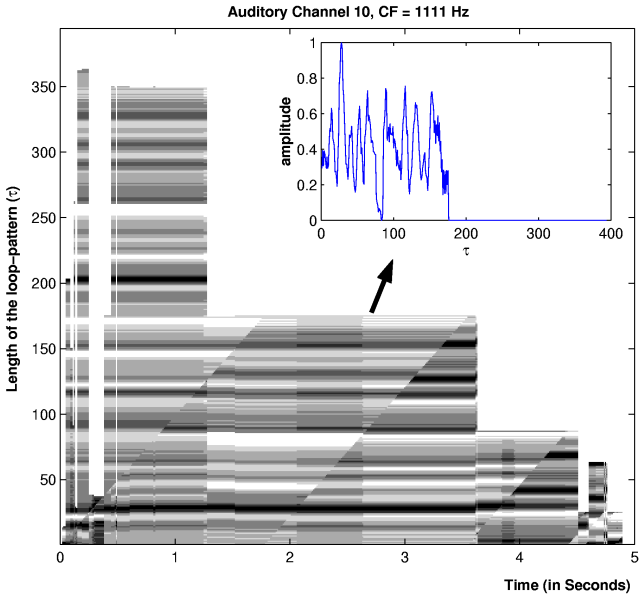


Figure 9: Picture of the patterns detected in auditory channel 10, which has a center frequency of 1111 Hz. The pattern at 2.5 seconds is shown in the small frame. This pattern has been selected for modulation of a band-pass noise with the same center frequency. The resynthesis accumulates the results of all auditory channels. The diagonal lines in this figure results from a phase correction such that stable period are seen as horizontal lines

amplitude modulated noises over all channels such that $s'(t) = \sum_{c=1}^{C} r_c(t)M_c(t)$, where $M_c(t)$ is the repeated pattern m_c at 2.5 seconds.[1]

The sequence of loop-patterns $m_c(t)$, for c corresponding to a center frequency of 1111 Hz is shown in Fig. 9. The figure displays the patterns detected in auditory channel 10, which has a center frequency of 1111 Hz. The pattern at 2.5 seconds is shown in the small frame. This pattern has been selected for modulation of a band-pass noise with the same center frequency. The re-synthesis accumulates the results of all auditory channels (Leman and Verbeke, 2000).

Although it has not been implemented thus far, the memory unit m can further be connected to a spatial memory, like the one considered in the previous section. This would allow an additional reduction of the data, with an additional classification and ordering.

As with LTSM, an ELTM can be conceived of as a resonance system, allowing stimulus-driven recognition of sequences or particular gestures, as well as top-down driven imagery. In the case of musical imagery, the top-down connection may activate a loop pattern providing a temporal unfolding of a musical idea (a performance, gesture, rhythmic pattern etc...). A musical structure can be conceived as a nested collection of such episodic memories. Musical imagery may reason about the memory units and activate them within the spatio-temporal memory system, hence producing a working memory for spatio-temporal images. Imagery may perform several operations on these units which are constrained by the principles of coherent transformations.

6 Links between brain-like computation and symbolic computation

Spatio-temporal representational entities can be connected to a symbol-based structure. The framework, as currently conceived, is a kind of two-way model, consisting in a link between lower-level perception-driven spatio-temporal representational structures on the one hand, and high-level conceptualizations on the other hand (See Fig. 10). The perception-driven part deals with the

[1]The sound examples are available at http://www.ipem.rug.ac.be/

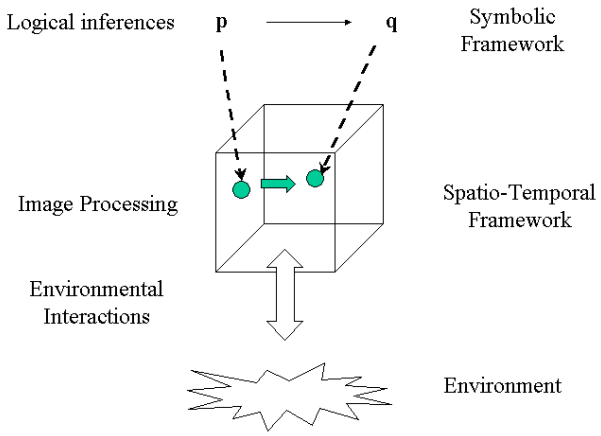


Figure 10: Connection between concept, images and sonoric objects

bottom-up processing of music signals into musical images. Starting from the acoustical waveforms, sounds are processed into images that serve as representational entities within a dynamic representational system which can be described by means of a logics.

The major advantage of this approach is that it ensures coherence between conceptualization in terms of inherent compelling forces (relating to sound and sound processing), rather than imposed meta-level relationships (established by an interpreter providing meaning to symbols). Our approach (Leman, ress) is based on the idea that logical reasoning grasps the way in which people reason in terms of conceptualizations (= abstract representations) of musical objects/events. A semantics of formal logical reasoning in music can be set up which relates certain formal expressions to image transformations, hence providing a model theory in terms of mappings to a domain of image transformations or interpretations that characterize truth.

In that sense, the truth of a particular formal expression can be guaranteed by an interpretation which in which coherence can be checked. Reasoning about musical concepts in the formal logical sense thus relies on a model which characterized the derivations in terms of perceptually constrained transformation principles. These principles guarantee logical coherence in terms of causal transformation principles. The implementation of this idea in terms of a computational approach requires a linking between logical inference and the mathematics of signal processing.

In the logical language, an operator is added which we denote by $\longrightarrow$. The expression
$p \longrightarrow q$
is read as "p image-implies q". It defines an operation on concepts whose truth-value is grounded in auditory-based modeling. The expression is validated (or falsified) by creating an interpretation of the expression in the domain of image transformation operators. This can be expressed by saying that in order to check the validity of $p \longrightarrow q$ we must check the validity of $\exists p', q' \Phi : \ p' \longrightarrow_\Phi q'$, where p' and q' are image instantiations (or interpretations) of p and q and where Φ refers to a(n) (chain of) image transformation operation such as APM, PM, EM, etc... The interpretation thus maps the expression $p \longrightarrow q$ onto a statement which has a truth value.

For example, the expression "this clarinet sound image-implies nasality" is valid if a (sufficient degree of) nasality can be obtained from an auditory-based signal analysis of this clarinet sound. It involves a set of distinguished image transformation principles that start with APM and subsequent analysis of the formant positions in derived spectral images. The truth of $p \longrightarrow q$, obviously, is contingent, which means that there are possible instantiations where it is false.

Rules of logical inference can be based on the image-implication operator, provided that on all instantiations of the image-implication operator, it cannot be false. In dealing with such an approach, we may keep the classical logical operators (such as conjunction ($\wedge$), disjunction ($\vee$),

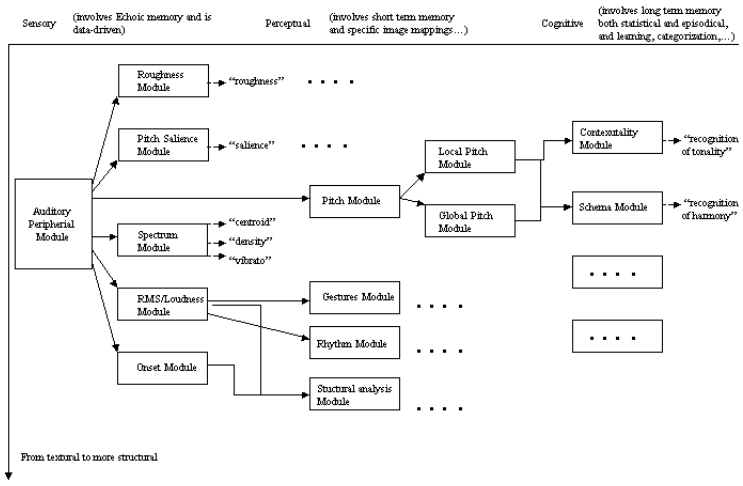


Figure 11: Chart of image transformation modules, organized along the distinction between sensory, perceptual, and cognitive information processing (horizontal axis), and from textural to structural (vertical axis). The chart is not exhaustive

negation ($\sim$), and even implication ($\supset$)) and their truth semantics in the classical, and just specify new rules of consequence for the image-implication operator. Such rules can be established if the expression allows any interpretation of the image-implication.

For example, we may add the following deduction rule:
$p \longrightarrow q,\ p \vdash q$
which says that if $p \longrightarrow q$ is true, and p is true, then we can conclude that q is true. For example, if a 75 Hz amplitude modulated sound image-implies high roughness, and we have a 75 Hz amplitude modulated sound, then we may conclude that we have high roughness. The truth of the whole expression is on condition that $p \longrightarrow q$ is true, which we take for granted.

In a similar way, we may characterize further deduction rules. For example, if a 75 Hz amplitude modulated sound image-implies high roughness, and we don't have high roughness, then we may conclude that we don't have a 75 Hz amplitude modulated sound. Hence, the following rule could be added to the inference system:
$p \longrightarrow q,\ \ \sim q \vdash \sim p.$
In a similar way, we may add $p \longrightarrow \sim q,\ q \vdash \sim p.$
(If a particular sound excerpt image-implies to be not in the key of C, and we are in the key of C, then we can conclude that we don't have this particular sound excerpt.

On the other hand, the following derivations cannot be accepted:
$\sim (p \longrightarrow q) \vdash q \longrightarrow p$
For example, if it is not true that a particular performance of a musical piece image-implies the expression of aggressivity, then we cannot derive that aggressivity image-implies that particular performance of a musical piece. In a similar way we don't accept:
$p \longrightarrow q \vdash p \supset q.$
It would mean that if p image-implies q, that p would logical imply q.

The framework, as currently conceived, is thus a kind of two-way model, consisting in a link between lower-level perception-driven spatio-temporal representational structures on the one hand, and high-level conceptualizations on the other hand.

Figure 11 gives an overview of modules as conceived from the viewpoint of a distinction between sensory, perceptual, and cognitive processing. This chart, showing some processing modules as organized from sensory to cognitive (horizontal axis) and from textural to more structural (vertical

axis), is not exhaustive. The modules are depicted as boxes, the associated indices as text, such as "roughness", "centroid", etc... Important for the current purpose is that each module, which stands for a signal/image function or operator, has one or more associated image types from which indices are derived.

7 Conclusion

Schema-theory played a central role in the models for tonal center recognition and chord decomposition, and provided support to the hypothesis of the development of stable long-term memory patterns, that are built by mere exposure to music according to a topological structure that reflects the invariant structural relationships inherent in the western tonal system. Progress in several related research areas is adding new information, from different perspectives, on many aspects of the transformation processes that a musical signal is undergoing while moving through the perceptive and cognitive pathways of the auditory system and brain components. The schema-hypothesis is now part of a larger frame: the focus is now shifted on the representational aspects of the sound images and the causal transformations that are operated in the various parts of the brain centers along the pathway from perception to cognition.

References

Arbib, M. (1995). Schema theory. In Arbib, M., editor, *The Handbook of Brain Theory and Neural Networks*. The MIT Press, Cambridge, MA.

Carreras, F., Leman, M., and Lesaffre, M. (1999). Automatic description of musical signals using schema-based chord decomposition. *JNMR*. submitted.

Eggermont (1997). Representation of amplitude modulated sounds in two fields in auditory cortex of the cat. In Syka, J., editor, *Acoustical Signal Processing in the Central Auditory System (The Language of Science)*, pages 303–319. Plenum Press, New York, N.Y.

Kohonen, T. (1995). *Self-organizing Maps*. Springer-Verlag, Berlin, Heidelberg.

Krumhansl, C. and Kessler, E. (1982). Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review*, 89:334–368.

Langner, G. and Schreiner, C. (1988). Periodicity coding in the inferior colliculus of the cat. Part I. Neuronal mechanisms. *Journal of Neurophysiology*, 60(6):1799–1822.

Leman, M. (1989). Symbolic and subsymbolic information processing in models of musical communication and cognition. *Interface – Journal of New Music Research*, 18:141–160.

Leman, M. (1995a). A model of retroactive tone center perception. *Music Perception*, 12:439–471.

Leman, M. (1995b). *Music and Schema Theory: Cognitive Foundations of Systematic Musicology*. Springer-Verlag, Berlin, Heidelberg.

Leman, M., editor (1997). *Music, Gestalt, and Computing - Studies in Cognitive and Systematic Musicology*. Springer-Verlag, Berlin, Heidelberg.

Leman, M. (1999). Naturalistic approaches to musical semiotics and the study of causal musical signification. In Zannos, I., editor, *Music and Signs – Semiotic and Cognitive Studies in Music*, pages 11–38. ASKO Art & Science, Bratislava.

Leman, M. (2000a). An auditory model of the role of short-term memory in probe-tone ratings. *Music Perception*, 17(4):481–509.

Leman, M. (2000b). Modeling musical imagery in a framework of perceptually constrained spatio-temporal representations. In God/oy, R., editor, *Musical Imagery*. Swets & Zeitlinger, Lisse, The Netherlands.

Leman, M. (2000c). Visualization and calculation of the roughness of acoustical musical signals using the synchronization index model (SIM). In Rochesso, D., editor, *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00), December 7-9*. University of Verona, Verona, Italy.

Leman, M. (in press). Expressing coherence of musical perception in formal logic. In Assayah, G., editor, *Music and Mathematics (provisional)*. Springer Verlag, Berlin, Heidelberg.

Leman, M. and Carreras, F. (1997). Schema and Gestalt: Testing the hypothesis of psychoneural isomorphism by computer simulation. In Leman, M., editor, *Music, Gestalt, and Computing - Studies in Cognitive and Systematic Musicology*, pages 144–168. Springer-Verlag, Berlin, Heidelberg.

Leman, M., Lesaffre, M., and Tanghe, K. (2000). *A Toolbox for Perception-Based Music Analysis (manuscript)*. IPEM - Dept. of Musicology, Ghent University, Ghent.

Leman, M. and Verbeke, B. (2000). Minimal 'energy' change (MEC) for recognition of repetitive rhythmic patterns in acoustical musical signals. *Journal of New Music Research*. submitted.

Meddis, R. and Hewitt, M. (1991). Virtual pitch and phase sensitivity of a computermodel of the auditory periphery I: Pitch identification. *The Journal of the Acoustical Society of America*, 89(6):2866, 2894.

Minsky, M. (1982). Music, mind, and meaning. In Clynes, M., editor, *Music, mind and brain: the neuropsychology of music*. Plenum Press, London.

Monelle, R. (1992). *Linguistics and Semiotics in Music*. Harwood Academic Publishers.

Neisser, U. (1967). *Cognitive Psychology*. Appleton-Centuri-Crofts, New York, NY.

Plomp, R. (1966). *Experiments on tone perception*. Van Gorcum, Assen.

Risset, J. (1992). The computer as an interface: Interlacing instruments and computer sounds; real-time and delayed synthesis; digital synthesis and processing; composition and performance. *Interface – Journal of New Music Research*, 21:9–20.

Schouten, J. (1940). The residue, a new component in subjective sound analysis. *Proc. Kon. Ned. Ak. van Wetensch.*, XLIII(3):356–365.

Schreiner, C. and Langner, G. (1988). Coding of temporal patterns in the central auditory nervous system. In Edelman, G., Gall, W., and Cowan, W., editors, *Auditory function: Neurobiological bases of hearing*, pages 337–361. John Wiley and Sons, New York, NY.

Seifert, U. (1991). The schema concept: A critical review of its development and current use in cognitive science and research on music perception. In Cammuri, A. and Canepa, C., editors, *Proceedings of the IXth Colloquium on Musical Informatics*, Genova. AIMI/DIST.

Shepard, R. N. (1984). Ecological constraints on internal representation: resonant kinematics of perceiving, imagining, thinking, and dreaming. *Psychological Review*, 91(4):417–447.

Tervaniemi, M. and Leman, M., editors (1999). *Cognitive Neuromusicology*. Swets & Zeitlinger, Lisse, The Netherlands. Special issue of *Journal of New Music Research*.

Van Immerseel, L. and Martens, J. (1992). Pitch and voiced/unvoiced determination with an auditory model. *The Journal of the Acoustical Society of America*, 91:3511–3526.

von Helmholtz, H. (1863/1968). *Die Lehre von den Tonempfindungen als physiologische Grundlage für die Theorie der Musik*. Georg Olms Verlagsbuchhandlung, Hildesheim.

Psycholinguistic Assessments of Language Processing in Aphasia
A Possible Frame-Work for Artificial and Human Communication

Erik Robert,
Head Department Speech Pathology
and Aphasiology AZ MM Gent,
Co-ordinator Neurological Speech Language Disorders KaHog Gent
and Foundation Institute Brain & Behaviour Rotterdam

The PALPA model is designed by Kay, Lesser and Coltheart to serve as a resource for speech and language therapists and cognitive and clinical neuropsychologists who wish to assess language processing skills for people with aphasia. (Aphasia is used in this context as an acquired language disorder). PALPA has brought a new approach into the clinical examination of individual patients with aphasia, one which is in tune with the philosophy of considering language assessment as an iterative procedure of hypothesis testing.

The model is primarily concerned with language as a complex series of mental processing steps and thus makes a somewhat artificial distinction between this and what we do when we use language to communicate. The justification for including artificial tasks into the PALPA (for example: reading or repeating non-words) is that these tasks can provide a window through which one can distinguish separate components or "modules" of language processing that might otherwise remain hidden if one considered only global measures of performance. Here, it is important to remark that the PALPA-approach is based on the assumption that the mind's language system is organised in separate processing modules, and that these can be separately impaired by brain damage.

Although the PALPA consists of 60 assessments, it is important to note that the PALPA is not designed to be given in its entirety to an individual with acquired brain damage. To appreciate how PALPA works, and to use it effectively, the language-processing model upon which each assessment is based must be thoroughly understood. One of the aims of this lecture is to provide a brief (and therefore necessarily basic) introduction on this matter. In order to do so, I will describe and explain the model's different modules, the communication pathways between these modules, and the type of language-processing function that depends upon each module or pathway.

Finally, I will place the symptomatology of the language of a (brain damaged) patient into the following diagram at page 38.

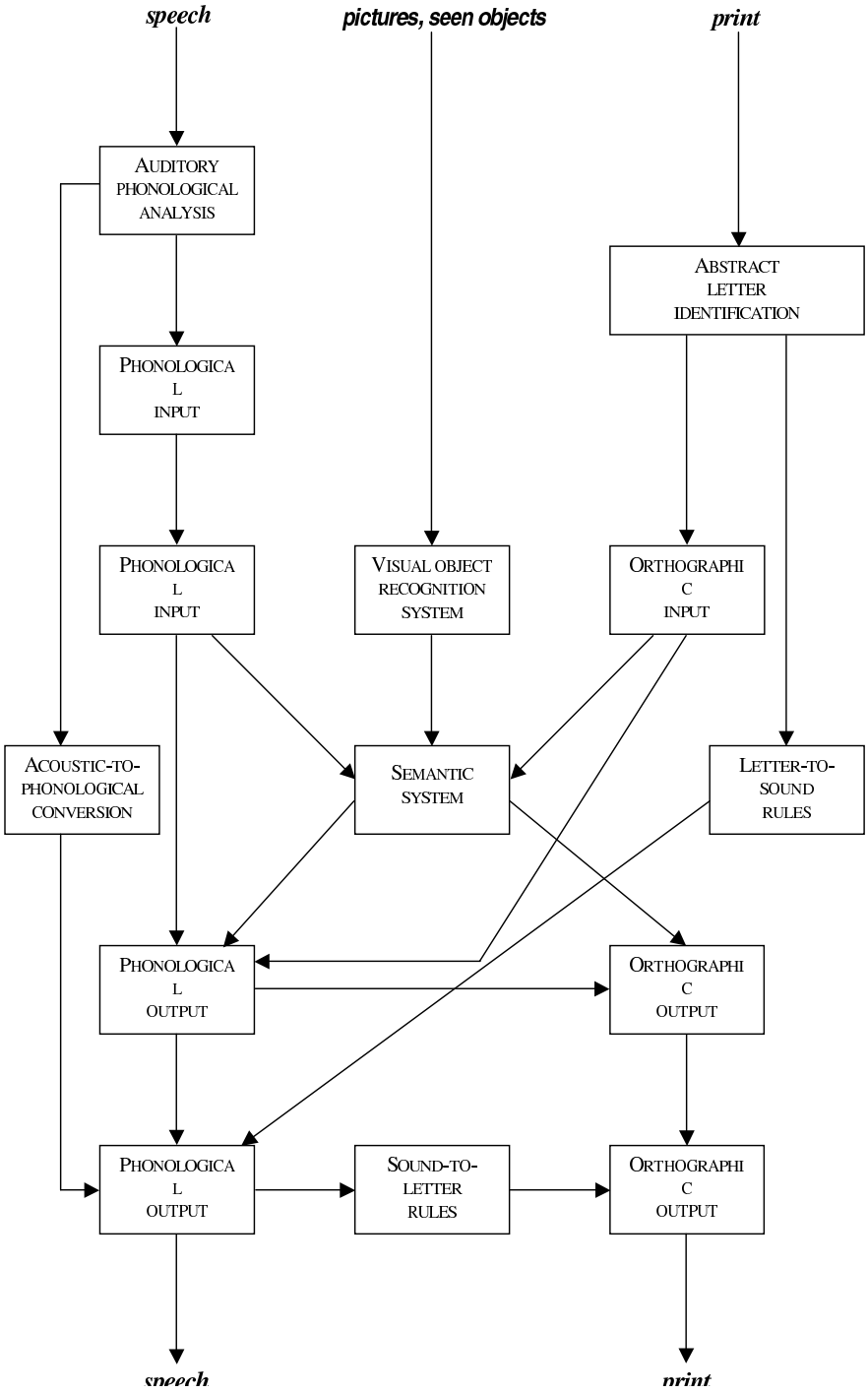


Figure 1: KAY J., LESSER R., COLTHEART M., 1995, p.172, figure 9

The remaining challenge is to apply this theoretical model to human communication. Is it possible to fit the pathology of people with acquired brain damage into theoretical models like PALPA? This seems to be problematic. Some videos will be shown to illustrate the difficult relationship between reality and theory. For several kinds of language-disorders cannot be understood by only considering speaking on the 'word level'. In addition, the pragmatic level, and the knowledge of the non-verbal reality remain essential aspects of the human communication.

Another major problem is the different modules' simultaneous activity. There is a fundamental difference between, on the one hand, a hierarchical mechanism which develops in successive stages that can be separated in time and, on the other hand, a neural circuit which different processing stages cannot be distinguished on a time scale.

Structuring language disorders in a theoretical model enables one to understand many elements but certainly not everything. Nevertheless, it remains useful in every new single casestudy to carry out the thought experiment. What models are not functioning for this particular patient and, very important in the light of future revalidation, what modules are still intact?

In case of a severe language disorder like global aphasia, the great frustration of the speech therapists is that, although the personality structure of the patient has not changed[1] , there is only little hope for a decent (humanlike) communication between him and the patient. These situations leave us, the therapists powerless in this sense that, at the end, we are forced to use non-verbal communication, although that even the functional possibilities of non-verbal communication are extremely restricted.

In this context, it is natural that a speech therapist living in an era of artificial intelligence and speech language technology starts dreaming aloud of a link between technology and his patient with aphasia. Is the day coming near where a microchip implantation into the brains can restore a language disorder or will this dream remain science fiction? It may be clear that it will only be accomplished through an intensive co-operation between speech therapists and computational linguists.

If it is true that technology enables a computer to speak, why then does it remain so difficult to let a human being with an often relatively limited brain damage speak again?

Combining the worlds of technology and pathology could lead to a real language revolution. The gains on both sides could be immense. On the one hand, more insight into the simulation of human communication with its imaginable and unimaginable applications would be acquired. On the other hand, one would be able to improve the therapy of aphasia, not only by giving alternative nonverbal solutions but also by giving speech back (at least partially) to a (mute) patient.

Therefore, may this lecture serve as an earnest call for a link-up between artificial and (brain damaged) human language.

One day, thousands of patients will thank us for this in Spoken and not in Broken English...

References

Kay, Lesser and Coltheart (1995). Dutch edition by Bastiaanse, Bosje and Visch-brink Lawrence Erlbaum Associates Ltd. ISBN 0–86377–366–4

[1] Unlike, for example, in the case of demantia

Dynamics of Sensorimotor Categorization and Language Formation: a Co-Evolution ?

Luc Berthouze
Electrotechnical Laboratory
Umezono 1-1-4, Tsukuba 305-8568, Japan
berthouz@etl.go.jp

Nadia Bianchi-Berthouze
Aizu University
Tsuruga, Ikki-machi, Aizu-Wakamatsu 965-8580, Japan
nadia@u-aizu.ac.jp

Abstract

We hypothesize that the evolution of language in an infant might be related to the sensorimotor categorization of its physical interaction with its environment and other agents placed in it. We discuss possible similarities between the dynamics of category formation and evolution in the brain and lexicon acquisition and we propose an experimental design to test this hypothesis.

Keywords: Categories, Dynamics, Chaotic itinerancy

1 Introduction

From the day of birth, a human baby engages in complex sensorimotor interaction with the world through its body. Through interaction, it acquires novel coordination skills which are used for further exploration, introducing new classes of interaction. And the cycle goes on. A typical interaction could be mother and baby "playing" with a ball for the first time, with the mother uttering the word "ball", possibly pointing at the ball. While the mother has built up an ontology as well as a vocabulary, there are no such things yet in the infant. From a perceptual point of view, the ball (as an object), provided that pointing by the mother removes ambiguity on the target of the utterance "ball", could only be segmented on the basis of a specific color channel, motion detection, size etc. While the infant shares a similar (in some extent) sensorimotor apparatus to the one of the mother, its sensorimotor experience is reduced. From an "evolution of language" point of view, the issues raised by such situation is two-fold, with two different time-scales to consider: (a) it is about the transmission of language from one generation to another (a macroscopic view of language evolution) and (b) it is about the evolution of language (or at least lexicon) in the infant, in a much shorter time scale.

The perspective from which we address the problem in our work is the following: should we and can we relate the *stability* of "ball" as a word in the evolution of the baby's language to the *stability* of "ball" as a meaning in the child's emerging cognition. The hypothesis we wish to discuss here is the hypothesis of *co-evolution*, i.e. lexicon formation dynamics and embodied interaction dynamics are intertwined and non separable. The question of "stability" relates to two observations: (a) if the child could possibly access the mother's "meaning" of ball, then it would simply be a matter of learning the right association. Instead, the only information on which to learn is an interactive feedback. Steels and Kaplan (1999) enounce the problem as follows: "agents must acquire word-meaning and meaning-object relations which are compatible with the word-object co-occurences they overtly observe but without observing word-meaning relations directly"; (b) if one assumes that there is no innate ontology in infants, then "ball" as a meaning can only

be "encoded" into specific color channel, motion detection, size, tactile feedback etc. But such information will evolve in time as interaction unfolds. This leads us to the question of structure and dynamics of categories in the brain.

2 Dynamics of sensorimotor categorization

Let's take a "dynamical systems" perspective and consider the phase space of a perception-action coupling (an infant) during its development. The system can be seen as following a path in a rugged landscape, shaped by the structure of the three-way interaction (system, environment, other agent), the system's own body structure (evolving in time). Driving forces are external forces (interaction with other agents, environmental) or intrinsic (value system, self-exploration). Categories emerge from sensorimotor interactions, trajectories in the manifold above as stable states (quasi-attractors). They are sensitive to initial conditions (history of sensorimotor experience). Let's consider the infant comes across two *large* objects. With the first object, the system effectively pushes it using its full body. However, it will experience the second object as a fixed obstacle, unable to displace it using its hand. Repetition of similar interactive patterns over time might lead to two distinct (from a perceptual point of view) categories of *large* object, different like two sides of the same coin. Yet, an external observer might utter the same word *large*.

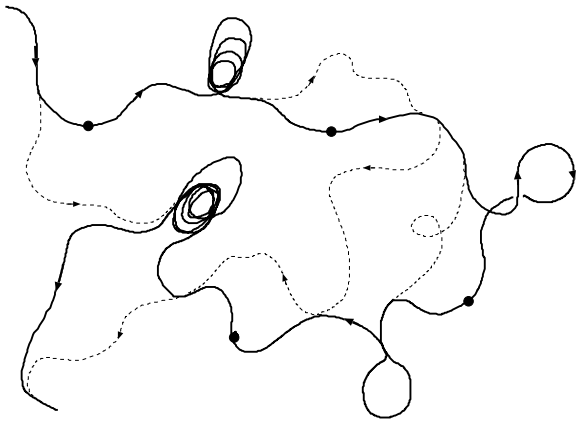


Figure 1: Complex manifold (in $\mathbf{R}^n$) for perception-action dynamics featuring two *categories* (the quasi-attractors) and redundant pathways between them (see details in text). Drawing inspired by a private communication from I. Tsuda.

From a neuroscience point of view, such categories are not available offline or statically encoded. Rather, they are activated as the system is involved in it (Berthouze, 2000; Yao and Freeman, 1991; Freeman, 1991). This idea is very similar to point attractor dynamics in memory models (Hopfield, 1982). Learned patterns are not recollectable without cues. However, by cueing the memory with a pattern that is *similar* to a learned pattern, the memory relaxes towards the learned pattern. In our hypothesis[1] that "meaning" of words relates to "sensorimotor categories", such fact has strong implications on the dynamics of the "meanings" of words. Indeed, external driving forces and exploration can lead the system to stray around existing categories. As interaction unfolds, the landscape transforms, shaped by new sensorimotor experience, new directions of instabilities appear. Some quasi-attractors (categories, i.e. meanings) might migrate, stabilize, or collapse. New categories can emerge. Such type of dynamics has been observed in interactive situations such as game play (Ikegami and Tajii, 1998) or in models of dynamical memories (Tsuda, 1992).

[1] Grounding "meaning" onto "sensorimotor categories" as discussed above does make sense in light of neuroscience studies. Those are among the only "stable traces" of neural activities over time. The reader should remember that there is no such thing as "static" stimuli as the brain undergoes uninterrupted sequences of stimuli. Yao and Freeman (1991) hypothesize that those are the constructs on which the brain performs pattern recognition in olfactory system for example.

Suppose now we build an agent endowed with a sensorimotor apparatus that relates to human sensorimotor apparatus, with a memory in which the above dynamics can occur, and in which words are labels for categories. If this agent is to interact with a human agent (with an existing ontology and vocabulary), could the dynamics of creation of "meaning", of "words" and "word-meaning" association [2] be independent of the dynamics of the categorization process ?

3 Learning interactive agent

We started an experiment dealing with how such agent can emerge a subjective lexicon, i.e. being able to communicate over subjective impressions with another agent in which it is assumed that an ontology and vocabulary exists but cannot be accessed by others except implicitly over interaction. In other words, human agent and system simultaneously observe a photograph and "discuss" it using words (or other photographs, as an extra-linguistic way to communicate). The system we have developed (Bianchi-Berthouze et al., 1999) is articulated on three main components (see Figure 2): (a) a sensorimotor apparatus, (2) a dynamical memory and (3) an active interface.

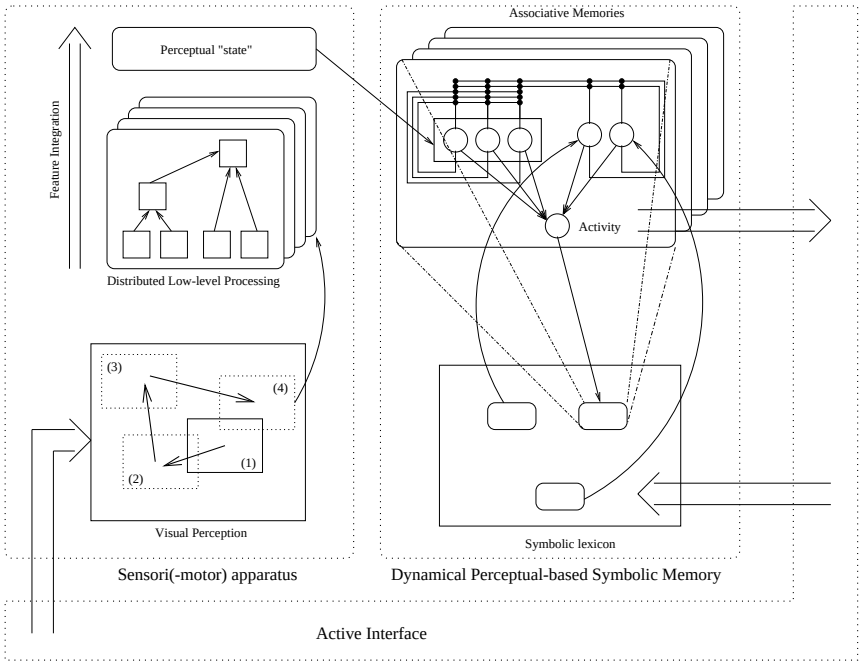


Figure 2: Architecture of the "learning" agent.

The perceptual apparatus consists of both a visuo-motor behavior which sequentially attends to specific areas of a photograph following a biologically plausible model of primary visual cortex by Rybak et al. (1998) and an array of low-level operators that process the retinal image on each area. Visual features considered include color distribution, brightness, edges orientation, homogeneity, contrast. This information, combined with the motoric activity corresponding to the scan-path is used to construct a "perceptual state" (sensorimotor pattern) that is viewpoint-dependent.

The dynamical memory is a replication of a model of hyppocampus (areas CA1 and CA3) proposed by Tsuda (1992) and which is an extension of Hopfield-type networks, i.e. by enlarging the basin of attraction by an algorithm of unlearning, it is possible to break the stability of recalled state and reach successive recall of stored memories. In such type of memory, *chaotic itinerancy* is achieved, i.e. creation, migration and collapse of attractors occur. By connecting these memories with the sensorimotor apparatus as shown in Figure 2, chaotic itinerancy on sensorimotor categories is possible.

[2] Assuming that none of the actors has control over the cognitive processes of the other one.

In our experiment, a "word" is defined as a label associated to a dynamical memory. When such memory is suitably cued, the memory relaxes into a learned pattern and the "word" is recognized. If the system cannot relax in any stable states in any memory, a new memory (i.e. a new word) is created. Similarily to classical Hopfield networks, one memory can learn multiple sensorimotor patterns. Hence, a "word" might be defined by multiple perceptions. This addresses the "grounding problem", i.e. the ambiguity in perception, change of point of view, feature selection etc., because categories are now dynamically stable yet flexible and variable.

Through the interface, human agent and system discusses their "perception": (a) whenever a new "word" is introduced, the system can request the human agent for further examples. It can also indicates to the human agent its Perceptual state through its own words in which case, it is up to the human agent to learn or not that word; (b) the system reflects on its own internal inconsistencies (i.e. two "words" being excited while mutually excluded), or the variability of the human (i.e. the human agent uses different words for a similar sensorimotor pattern) to prime further interaction so that inconsistencies can be seen as brainching points in the interactive dynamics. The human agent might ignore, correct or propose alternatives. As noted by Steels and Kaplan (1999), humans as well as autonomous agents only get feedback on the communicative success of an interaction, not on what meanings were used. Even communicative success may not be completely clear. If this source of uncertainty does contribute to make learning difficult, we believe it contributes to the livelihood of the dynamical systems by making the existence of "fixed point" or "trivial attractor" unlikely. Indeed, further interaction triggers adaptation in each actor of the dialog: adaptation of the "meaning" of the word (i.e. different selections of visual features) and adaptation of the "word" (i.e. the label given to a given set of visual features). Synchronous and/or asynchronous adaptation of each agent's processes may result in complex dynamics of the reconfiguration of the emerging shared language.

4 Conclusion

We proposed a platform for studying the entanglement between embodied interactive dynamics (such as mother-child interaction, linguist-tribe interaction) and sensorimotor categorization dynamics such as observed in the brain. A shared language emerges if only each agent can find and share meaning through the use of a word, for a same reality filtered by two distinct sensorimotor experiences. We have identified a few key mechanisms in the categorization process, namely creation, migration, collapse or stabilization, which we believe relate to typical mechanisms observed in the evolution of language (new word, damping of synonimy, polysemy etc.). They are also coherent with the phenomena of "overextension" and "underextension" by children in their first words (Clark, 1993) (reported in (Steels and Kaplan, 1999)). Similarily to Steels, we observed that some categories survive and not others but differently from him, we do not attribute this survival to the fact that such categories were easily agreeable on, or because inconsistencies were filtered out but simply because they form a stable state in the interactive process. In other word, a word may not survive because it has a maximal discriminative power but only because it maximizes the mapping of the agents' cognitive processes one onto another. This is also why we chose to address the issue of learning of "subjective" words because subjective responses are not stable over time.

References

Berthouze, L. (2000). Neural learning of sensorimotor categories. *Proceedings of 4th International Conference on Computational Intelligence and Neuroscience, Atlantic City (USA)*, pages 840–843.

Bianchi-Berthouze, N., Berthouze, L., and Kato, T. (1999). Understanding subjectivity: An interactivist view. *Proceedings of International Conference on User Modeling (UM'99), Banff (Canada)*.

Clark, E. (1993). *The lexicon in acquisition*. Cambridge University Press, Cambridge.

Freeman, W. J. (1991). The physiology of perception. *Scientific American*, pages 34–41.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of National Academy of Science*, 79:3088–3092.

Ikegami, T. and Tajii, M. (1998). Structures of possible worlds in a game of players with internal models. In *Acta Polytechnica Scandinavica – Mathematics, Computing and M anagement in Engineering Series*, number 91, pages 283–292.

Rybak, I. A., Gusakova, V. I., Golovan, A. V., ladchikova, L. N. P., and Shevtsova, N. A. (1998). A model of attention-guided visual perception and recognition. *Vision Research*, 38:2387–2400.

Steels, L. and Kaplan, F. (1999). *Linguistic Evolution through Language Acquisition: Formal and Computational Models*, chapter Bootstrapping grounded word semantics, pages –. Cambridge University Press,.

Tsuda, I. (1992). Dynamic link of memory – chaotic memory map in nonequilibrium neural networks. *Neural Networks*, 5:313–326.

Yao, Y. and Freeman, W. (1991). Model of biological pattern recognition with spatially chaotic dynamics. *Neural Networks*, 3:153–170.

Modeling Cognitive Processes with the Recommendation Architecture

L. Andrew Coward
School of Information Technology, Murdoch University,
Perth, Western Australia
landrewcoward@home.com

Abstract

The design of an electronic system with the recommendation functional architecture is described and the results of system learning with inputs simulating visual and verbal sensory inputs are presented. The differences from other cognitive architectural approaches are discussed, and it is emphasized that the recommendation architecture makes it possible for a system performing a complex combination of functions to be constructed, to recover from failures and damage, and to learn without affecting earlier learning. The properties of the implemented system supporting this conclusion are described. Given the understanding of the capabilities of the recommendation architecture derived from experience with the implemented system, complex cognitive processes can be realistically modeled.

Keywords: Cognitive Model, Connectionist Architecture

1 INTRODUCTION

Over the past 30 years there have been major advances in the technology for design of systems to perform very complex combinations of functionality. Design experience with such systems has emphasized that the needs to construct, repair and modify the functionality of such systems force the adoption of simple functional architectures. Coward [5] therefore proposed that biological brains have also been constrained to adopt simple functional architectures, and that because of the need for some biological brains to heuristically define their own functionality, the functional architecture adopted by such brains is the recommendation architecture. Coward [5] pointed out that many psychological and physiological phenomena could be understood as phenomena to be expected in systems with the recommendation architecture. Coward has further argued that the recommendation architecture can be the basis for understanding all cognitive phenomena in terms of physiology [7] up to and including human consciousness [8], [9].

Simple simulations of systems with the recommendation architecture demonstrated that memory phenomena analogous with biological memory occurred in such systems [6]. A more detailed recommendation architecture which included management of the context for partially ambiguous information exchange was described in [11]. Simulations of portions of this architecture have been described [10], [11], [12].

The purpose of the current paper is to describe the implementation of a full version of the architecture described in [11], and demonstrate its application to a simple categorization problem and a more complex problem involving recognizing objects, recognizing and generating speech, and imagining objects. The architecture is then compared with other proposed cognitive architectures.

2 FUNCTIONAL COMPLEXITY AND FUNCTIONAL ARCHITECTURES

Some current commercial electronic systems have billions of hardware components and tens of millions of lines of software code [19]. Some of the most complex systems are real time systems which use input data to directly control a physical system, such as a telecommunications network or a chemical processing plant, with no human intervention and with severe time constraints within which it must respond [3]. In such a system a response will depend upon what has previously happened, and a timely response depends upon many processing tasks which must be carried out concurrently. The problem in real-time system design is to partition the allowed end-to-end elapse times from

external event to the deadline for system action between the many modules which may require both time to generate their individual outputs and also input information from other modules which themselves require time to generate that information.

As the complexity of electronic systems increased, it became clear that an architecture which provided a multilevel descriptive hierarchy was required [19], and that in the absence of such an architecture it is extremely difficult to achieve integration of different functional modules [13]. A critical aspect of functional integration is provision of a context for information exchange between modules. In a commercial electronic system modules that exchange information must share enough context for the information to be acted on unambiguously. A major problem in integrating modules designed for different systems is the lack of common information contexts [13].

Coward [5], [6] pointed out that there are severe constraints on any system which performs complex combinations of functions. These constraints derive from the needs for processes by which such systems can be constructed, recover from failures and damage, and modify some functionality without affecting other functionality. All these processes require the existence of some means to relate functionality described at high level to functionality described at a detailed level. For example, any repair action requires that it be possible to derive from a description of a problem at high level where it is experienced (e.g. this system feature is not working) a description at a detailed level where it is possible to take corrective action (e.g. this device is not working). These requirements result in any such system being constrained into a simple functional architecture. In a functional architecture the functionality of the system is separated into modules, these components into smaller submodules and so on down to the smallest elements of functionality. This hierarchy of modules is the multilevel descriptive hierarchy as described by [19]. In a simple functional architecture all the modules on one level perform roughly equal proportions of system functions, and information exchange between modules, although essential for coordination of system functions, is minimized. These constraints permit the existence of usable logical paths from high level problem descriptions to detailed level descriptions [11]. It would be extremely difficult to construct, repair or modify a system which performed a complex combination of functions if the system did not have a simple functional architecture.

In commercial electronic systems the use of completely unambiguous contexts for information exchange between functional modules results in the memory/processing separation and instruction based operation of the von Neumann architecture [11]. If a system is to heuristically modify its own functionality (i.e. learn from experience), individual modules must determine the inputs they will receive from other modules. Under these conditions it is impractical to maintain an unambiguous context for all information exchange, but the requirement to maintain a partial, ambiguous context results in such a system being forced to adopt the recommendation architecture [11].

3 THE RECOMMENDATION ARCHITECTURE

A system with the recommendation architecture defines a portfolio of partially ambiguous information conditions out of its experience. The portfolio is defined in such a way that every experience is a repetition of one or more of these conditions. For any novel experience, condition definitions are dynamically extended so that the experience includes some of the extended conditions. Different combinations of repetition conditions are associated with different behaviours.

In the recommendation architecture there is a primary separation into clustering and competition. The clustering subsystem handles functional complexity and is therefore forced to adopt a simple functional architecture. The competitive subsystem is functionally simple and does not require such an architecture. The clustering subsystem selects and permanently records combinations of inputs from the environment and indicates by an output that they are occurring, and also indicates if a sufficiently similar repetition occurs in the future by generating a similar output. The clustering subsystem compresses a potentially huge input space into a much smaller output space. The competitive subsystem uses reinforcement learning to interpret the outputs of the clustering subsystem as alternative behavioral recommendations and selects one behavior.

The hierarchy of modules in the clustering subsystem is a hierarchy of repetition complexity. At the device level, combinations of device inputs are selected and detected if repeated in the future. Modules are made up of sets of devices, and a module condition repeats if a significant subset of its devices indicate a repetition. Supermodules are made up of modules and so on. The repetitions detected by modules on different levels can range in complexity from relatively simple direct combinations of system inputs at the device level to very complex combinations of such combinations occurring in a specified time sequence. To achieve this complexity, modules use the outputs of other modules as their inputs. Information combinations are recorded instantaneously and permanently at the device level. This recording algorithm is essential to maintain context for information exchange but is radically different from

conventional neural network algorithms. The reason that outputs must be stable is that any output may be distributed widely to many other modules, but the distribution of any specific output is determined heuristically at the individual receiving module level. In other words, when a module begins to produce an output, any one of the potentially huge population of other modules might decide to use that output as an input. If the conditions under which that output was produced were later changed, the meaning of that output would have changed, and it would in general be impractical to communicate the changed meaning of the output to all its recipients.

The high level recommendation architecture is illustrated in figure 1. Information combinations are heuristically selected by the clustering subsystem. The selection process is unguided and in general there will be some random element in the determination of which combinations are recorded. Architectural considerations can improve on completely random selection, but cannot eliminate the random element. All information communicated (i.e. all outputs indicating the presence of repetitions on different levels) within the clustering subsystem or from the clustering subsystem to the competitive subsystem is therefore partially ambiguous: for example, no outputs at any level correspond consistently with patterns or categories in the input space which are clear conditions for system behavior. All outputs must therefore contain enough information to provide context for the recipient function to use the output. An output may be available to any of a large number of potential recipients, each of which makes its own decision whether to use the output. The requirement to maintain context means that once an information combination has been recorded and indicated by an output, any exact repetition of the combination must always result in an output which includes exactly the same output.

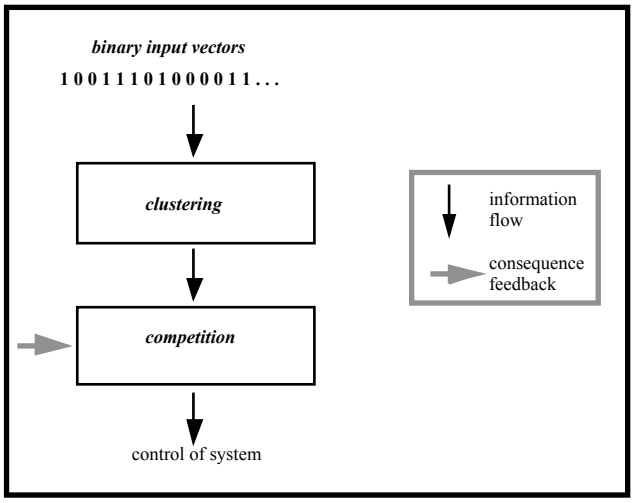


Figure 1. High level view of the recommendation architecture

There are four functions which an effective clustering subsystem must perform. The subsystem must select the input space in which it will search for repetition. It must select repetitions in such a way that there is an information compression between a potentially huge input space and its output space to the competitive subsystem. It must accurately detect the presence of any repetition, and it must indicate the presence of a repetition with enough information richness to allow the competitive subsystem to generate a high integrity behaviour. Much of the structure of the clustering subsystem is required to perform these functions effectively in such a way that proliferation of resources (such as total number of device level repetitions) is minimized. The requirement for a simple functional architecture must also be met: modules on one level must detect roughly equal numbers of repetitions in ongoing experience; and information exchange between modules, although essential, must be minimized. In the recommendation architecture there is a primary separation into clustering and competition. The clustering subsystem handles functional complexity and is therefore forced to adopt a simple functional architecture. The competitive subsystem is functionally simple and does not require such an architecture. The clustering subsystem selects and permanently records combinations of inputs from the environment and indicates by an output that they are occurring, and also indicates if a sufficiently similar repetition occurs in the future by generating a similar output. The clustering subsystem compresses a potentially huge input space into a much smaller output space. The competitive subsystem uses reinforcement learning to interpret the outputs of the clustering subsystem as alternative behavioral recommendations and selects one behavior.

Use of consequence information is essential to enable the system to learn to perform appropriate functionality. However, consequence information cannot be used to modify the outputs generated by clustering, because this would not maintain information context. For example, suppose that a set of repetition conditions detected by clustering were interpreted by competition as a recommendation to perform a particular behaviour, but the consequences of the behaviour were negative. The modules in clustering which were active in generating the recommendation will also be active in generating different behaviours in different circumstances. Hence these modules cannot be modified to give better performance for the current behaviour without introducing unknown changes to other behaviours. Therefore, as shown in figure 1, consequence feedback can only be applied to the interpretation of clustering outputs in the competition subsystem. The competitive function uses consequence feedback and reinforcement learning algorithms to converge on appropriate behavioral interpretations of clustering outputs.

4 MORE DETAILED VIEW OF THE RECOMMENDATION ARCHITECTURE

Inputs to the system as illustrated in figure 1 are vectors in which the elements indicate the presence or absence of some characteristic in an input space. These elements could, for example, be the presence or absence of pixels at different points in a visual field. As described below, in the implemented system these vectors are binary (i.e. either present or absent). Continuously variable elements are possible with greater architectural complexity.

4.1 THE CLUSTERING SUBSYSTEM

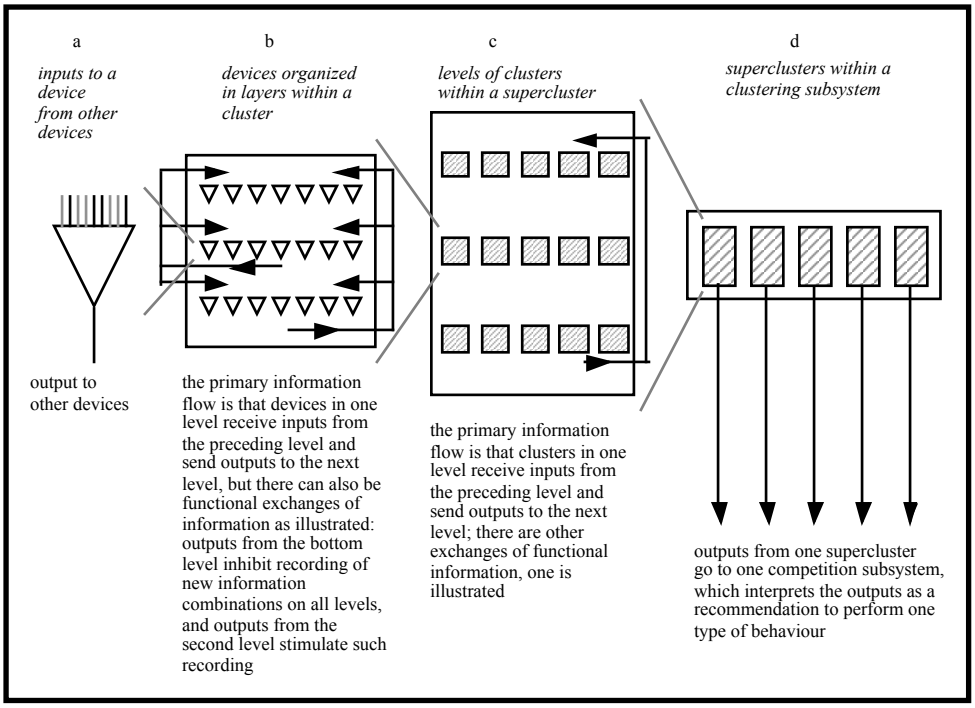


Figure 2. The architecture of a clustering subsystem at various levels of detail. A device is the basic element which records information combinations. The illustrated device had a number of provisional inputs selected by a random process. The black inputs were the inputs active when the device was triggered to select its information combination. The grey inputs were provisional inputs which were not active and are therefore deleted. The device will produce an output if a large subset of its black inputs repeats in the future. Devices are organized into layers, layers into clusters, and clusters into superclusters (additional levels are possible but not illustrated). Each level records and detects information combinations made up of sets of combinations at the more detailed level, and indicates any repetition of a large subset of its set by producing an output. Outputs are all ultimately combinations of device outputs.

The clustering subsystem uses devices which can record combinations of information and indicate any repetition of the combination by producing an output. Such a device is illustrated in figure 2a. A device initially has randomly selected inputs, all with the same weight, but a threshold set so high that it will not respond to any input combination. Given a significant number of active inputs plus a functional signal supplied as described shortly, the device produces an output, and also disconnects all its inactive inputs and sets its threshold so that any repetition of a large subset of its current inputs will cause it to produce an output. The device has effectively recorded an information combination. Devices which record multiple combinations are possible, but require somewhat more architectural complexity as described in [11].

The functional signal which triggers a device to record a combination can be understood from figure 2b, and from the more detailed view shown in figure 3. This figure illustrates a functional module called a cluster, which in the illustrations is made up of three layers of devices of the type described. Devices in the top α layer receive inputs from outside the cluster, devices in the middle β layer receive inputs from the top layer, and devices in the bottom γ layer receive inputs from the middle layer. All these inputs define information combinations which increase in complexity from top to bottom. In addition, devices in all layers receive functional inputs from a set of β layer devices which together indicate how much firing is present in the middle layer. These inputs stimulate any recipient devices which are able to record an information combination to do so. Devices also receive functional inputs from a set of γ layer devices which together indicate if any firing is present in the bottom layer and inhibit recording of new information combinations.

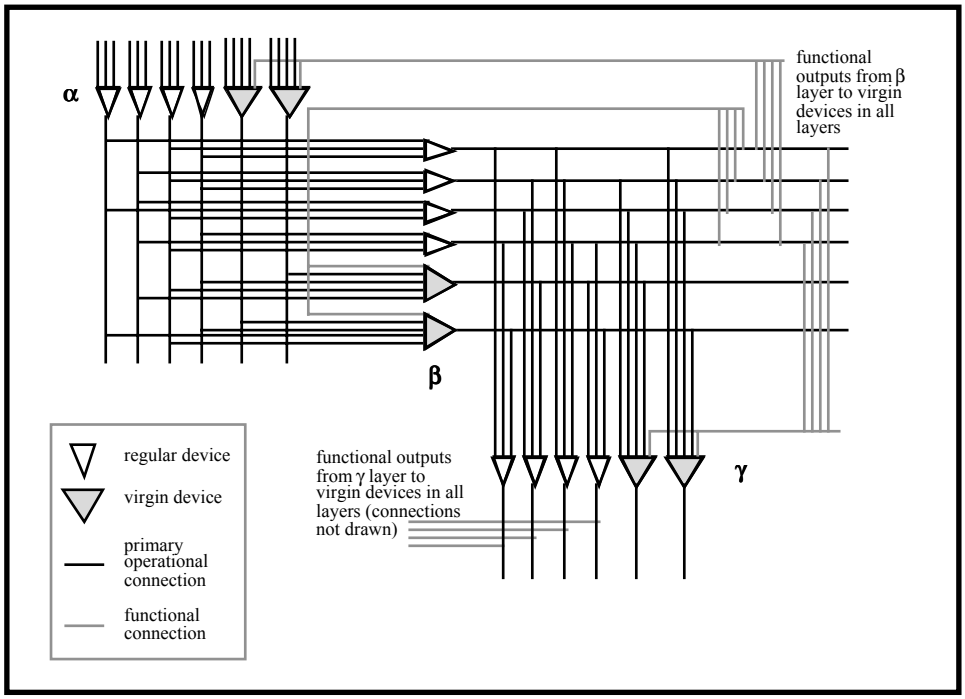


Figure 3. Connectivity within one cluster. Primary connectivity is excitatory. Functional connectivity from γ level to virgin devices inhibits virgin device imprinting, and is stronger that the functional connectivity from β level to virgin devices which stimulates such imprinting.

A layer of such clusters such as one of the layers shown in figure 2c can divide up experience into a set of heuristically defined repetition conditions. To understand this process, suppose that a system receives a sequence of sensory inputs derived from just apples, plums and blueberries. These fruits can be distinguished by size, shape, color etc. but the system has no knowledge of these categories in advance. Suppose further that the first object is an apple. No cluster produces any output because no combinations have been recorded so far. This absence of response triggers random selection of one cluster, and information combinations are recorded until a cluster output results. Any future object may produce an output from this cluster if it generates enough information repetition in the middle layer. The middle layer information combinations might happen to correspond with conditions like *some degree of green plus some degree of roundness*. Such a combination could be triggered by more green and less roundness or

vice versa, and therefore although on average an apple is most likely to trigger output, some plums and even some (unripe) blueberries might trigger output as well. As the cluster is exposed to more experiences, the set of input conditions to which it will respond broadens through recording of additional information combinations. Outputs from such a cluster might eventually indicate the presence of a spectrum of repetition conditions which might respond to most apples, some plums, and a few blueberries. An object which generated no response in this cluster would trigger creation of a new cluster which would develop a response to a different spectrum of repetition conditions. The clustering subsystem is managed to the point at which every experience results in some output from at least one cluster. In other words, the space of detected repetitions is expanded until every experience contains at least some repetition. Initially many experiences require recording of new information combinations, this recording tends to zero with time unless radically novel conditions are experienced.

The outputs from such clusters are the outputs of devices in the bottom layer, and the particular combination of outputs provides information about where in the spectrum of cluster repetition conditions the current input condition is located. Simulations have demonstrated that such a system can divide its experience up into a limited number of clusters, each indicating different (although sometimes partially overlapping) repetition conditions, and that although the outputs from one cluster do not correlate unambiguously with one object category, the outputs in combination contain information in a form which can readily identify the input category [11]. A separate competitive function using simple correct/incorrect feedback for early experiences will rapidly converge on high integrity identification of the correct object category. A complex input experience space may require subdivision of input clusters into repetition conditions which are different combinations of the first level clusters. This subdivision can be achieved by successive layers of clusters as illustrated in figure 2c. This hierarchy of clusters make up one supercluster. The clustering subsystem is separated into superclusters which independently select and indicate repetitions as illustrated in figure 2d. Outputs from different superclusters are directed to the competitive subsystems for different major types of behavior such as aggressive, fearful, food seeking, speech etc. The reason for separate superclusters is that the type of information combinations most useful for determining one type of behavior may be different from those useful for another type. Discrimination of this type could be provided by a genetically programmed selection of different input spaces for different superclusters. It can also be learned to some degree despite the restriction on changing outputs in response to consequences. If the same cluster output, when interpreted as an accepted behavior, sometimes results in good consequences, sometimes bad, the cluster is no producing a sufficiently rich output for functional purposes. If the cluster is then forced to accept additional inputs and record new combinations even when producing outputs, new outputs will provide additional discrimination. Previously recorded combinations are not affected, so the context for information is retained.

At the highest level, the clustering subsystem can thus be understood as arbitrarily dividing up all its experience into sets of repetition conditions. The presence of any of these repetition conditions is indicated by an output from the functional module detecting the condition. The competitive function then has the task of interpreting these combinations of repetition conditions into a system behavior. If the repetition conditions were generated by a supervised process, device level repetitions would be patterns, clusters would correspond with categories, superclusters with supercategories. In a clustering subsystem the correlation with such cognitive, functionally useful conditions is partially ambiguous.

4.2 THE COMPETITIVE SUBSYSTEM

The competitive system interprets the functionally ambiguous outputs of the clustering subsystem into a system behavior using a form of reinforcement learning. The algorithms for this reinforcement learning can be understood by consideration of figure 4. The devices, as illustrated in figure 4a, are similar to classical neural network devices. They can have both excitatory and inhibitory inputs with different input weights. In its simplest form the competitive subsystem is made up of a series of parallel pipes (figure 4c). Each pipe corresponds with one type of behavior, and the pipe with the strongest output is the behavior in response to the current input condition. A pipe has two layers of devices (figure 4b). Devices in the first layer receive combinations of inputs from devices in the corresponding supercluster. These inputs have a excitatory weight. Devices in the second layer receive stimulatory inputs from first layer devices in the same pipe, and inhibitory inputs from devices in the first layer of other pipes. The threshold of all devices in all pipes is lowered until there is an output from at least one pipe. The strongest output is taken as the behavior, if there are several outputs of equal strength, one is selected at random. If the consequences of the action are good, all recently active excitatory input weights are increased and inhibitory weights reduced, and vice versa for

bad consequences. Over a period of experience such a subsystem converges on a high integrity behavioral interpretation of the clustering subsystem outputs.

In a complex system with multiple superclusters, a more complex competitive architecture is required. A set of pipes is needed to interpret the outputs of each supercluster into one of a number of different behaviors of the same general type, then a later set of pipes to select behavior between supercluster type. The different types of behavior corresponding with different superclusters and corresponding pipe sets include actions on the system itself as well as external behaviors. One important behaviors of this type is to keep the current set of active repetitions active for longer. This behavior corresponds with, for example, keeping a recently read telephone number active for long enough to dial it. Another important type is to activate a new set of repetitions which are not currently active but have often been active in the past when the current clustering subsystem outputs have been present.

Although the clustering subsystem cannot be directly affected by feedback of consequences, there is an indirect mechanism which is functionally important. If similar outputs from the clustering subsystem at different time are interpreted as similar behaviors but result in opposite consequences, this condition is effectively an indication that the discrimination between repetition conditions is too coarse. A response which would not affect the context for existing outputs is to add input information to the relevant clusters and to generate additional outputs in experience conditions where there are already outputs. This mechanism would not change existing outputs but the additional outputs would allow the competitive subsystem to distinguish between conditions which were similar but different in a functionally important way.

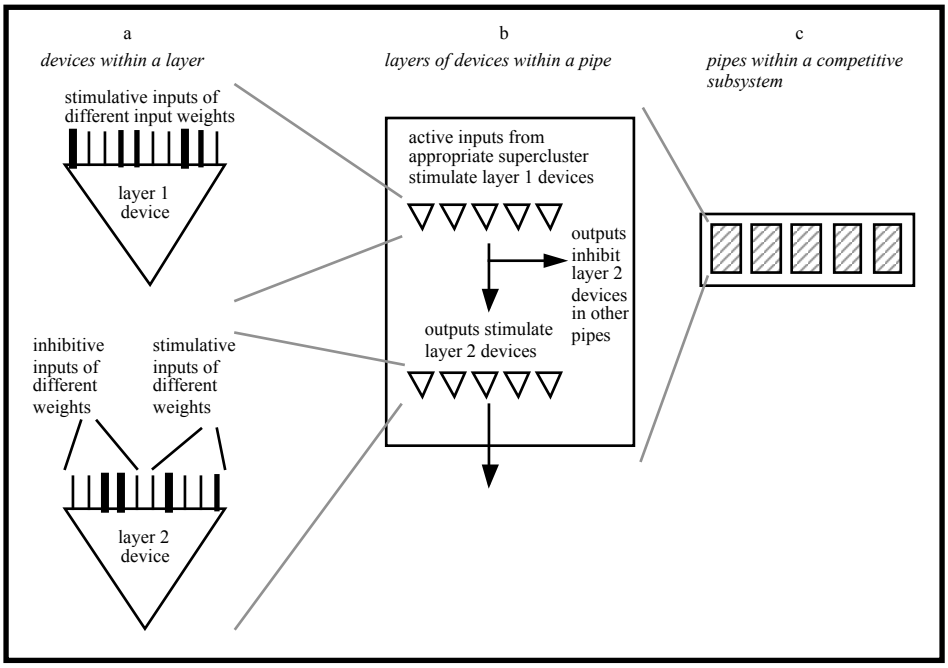


Figure 4. The competitive subsystem

Adding additional inputs increases the information distribution within the functional architecture. Such information distribution, although essential for the coordination of functionality, must be minimized. Minimization of information distribution also applies at the device level: the number of inputs must be minimized subject to the need to record functionally useful information combinations. Because modules at all levels heuristically determine information exchange, there must be a process to limit such exchanges to those most likely to be functionally relevant. The only accessible criterion to assess probable functional relevance is correlation of information in time: either simultaneous or with a constant separation interval. Hence a mechanism is required to limit provisional connectivity (at device, cluster, or higher levels) to inputs active when functionally relevant information has been active in the past. A rapid rerun of past experience would generate the environment in which such a mechanism could operate, and [5] therefore proposed that information distribution minimization is the primary function of dream sleep.

5 IMPLEMENTED RECOMMENDATION ARCHITECTURE SYSTEM

An electronic implementation of the recommendation architecture has been completed. Because the functionality of the system depends upon dynamic modification of connectivity (not just connection weights), devices and their connectivity are simulated in software rather than physically constructed. The system is written in Smalltalk V and runs on any Apple Macintosh platform. The system is presented with an input, and sequentially evaluates the state of each device in order to determine its response. Multiple passes through the system are required for each input because of various functional feedback paths described below. An inherent parallelism of the architecture (i.e. large groups of devices which could have their state determined in parallel) is not currently exploited. Inputs to the system are binary vectors which indicate the characteristics of the current input condition. These binary vectors can be of any size, simulations have been performed with input vectors up to 10,000 elements. Simulation time increases roughly linearly with input vector size, but if the inherent parallelism were exploited would be almost independent of vector size.

5.1 SIMPLE CATEGORIZATION SCENARIO

To determine the ability of a recommendation architecture to heuristically organize its inputs into a hierarchy of repetition and use the outputs to determine behaviour, an artificial scenario was defined similar to those used in earlier simulations [10], [11], [12]. In this scenario, an input space of 1000 possible binary inputs or properties was used. Input conditions were 1000 element binary vectors, where a one-element indicated the presence of a property and a zero-element its absence. An input condition was generated by random assignment of ones and zeros based on a relative probability distribution. Different probability distributions defined different types of input conditions. Such different types could be interpreted as different categories of input conditions. In the simple category recognition simulations, ten categories were defined in the input space. The probability distributions for the ten categories are shown in figure 5. Any one probability distribution had a non-zero probability for about 50% of the input properties.

The problem the system is required to solve can be understood by consideration of figure 6. In that figure, the properties of 10 objects from two adjacent categories in a 100 input space are shown. The category of an input condition cannot be determined on the basis of the presence or absence of a few individual properties. For example, for objects in the 1000 input space, only about 60% of the conditions of a category include both of the two most probable properties for that category, and 5% do not contain either of the two properties. However, 5% of the objects in the categories on either side (in figure 5) of that category contain both of these properties, and 50% contain one of them. 10% of the objects in the two categories next furthest away in figure 5 also contain one of these properties.

The recommendation architecture system was presented with a sequence of different input conditions (or objects) of the ten different types. Periodically the sequence was interrupted by a period of sleep to manage information distribution. A typical experience sequence was 100 input objects including 10 different objects of each category presented in the order A B C D E F G H I J A B C etc. This experience was followed by a period of sleep, and then another sequence of 100 objects . A simulation run included 760 different objects of each type being presented to the system, each object being presented only once.

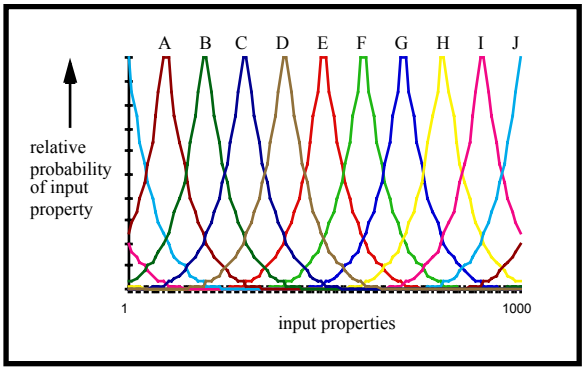


Figure 5. Probabilities of occurrence of properties in objects belonging to the ten categories

5.2 HIGH LEVEL ALGORITHMS

The clustering module organized the input conditions into a hierarchy of repetition. This hierarchy included devices which could be imprinted with an information condition, layers of devices, and clusters made up of several layers. The outputs from the clustering function were the outputs of devices in the final layer of each cluster. These outputs were presented to a competitive function with access to some correct/incorrect feedback to generate behaviour appropriate to the different input conditions. This feedback took a number of forms.

In one form of feedback, called regular, positive feedback was given if the selected category was correct, negative if it was incorrect. In a second form, no feedback was provided after feedback had been given for a certain number of presentations, to see if the behaviour continued undisturbed. 1000 individual input conditions of each category were created by randomly selecting properties from a set in which they occurred in proportion to the probability distribution for the category. 210 random selections were made to generate each input condition. Duplicates were discarded, resulting in an input condition having between 126 and 160 properties, with an average of 142. This process resulted in input conditions which were all different. One objective of the simulations is to determine whether the architecture can heuristically identify repetition conditions in its inputs which can usefully be associated with system actions under conditions in which no input condition ever repeats exactly.

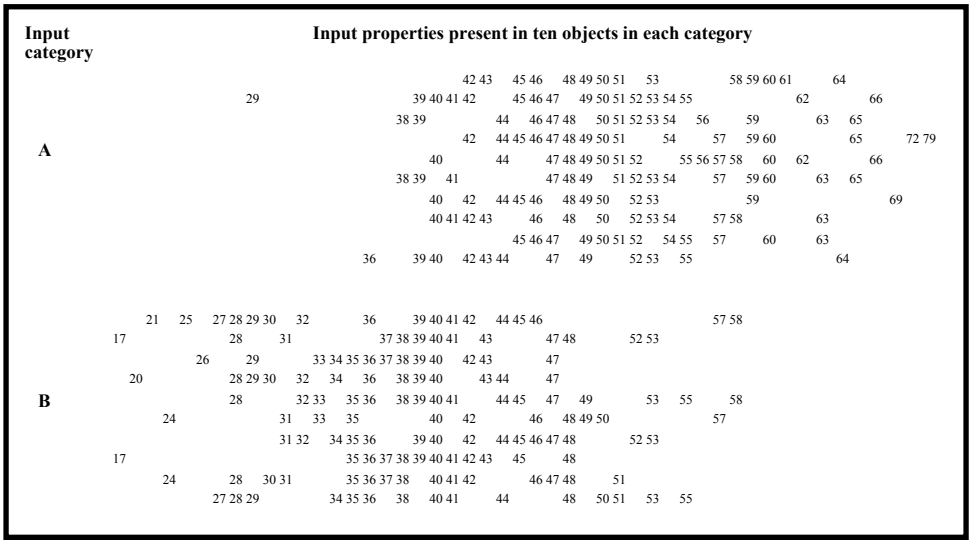


Figure 6. Examples of ten objects from each of two different categories constructed by random selection of properties from a 100 property input space. The numbers are the properties of the objects. The two categories had probability distributions of the form illustrated in figure 5. The actual objects used in the categorization experiments described in this paper were constructed in a 1000 property input space.

At the highest level, a series of objects are presented to the system. After a number of presentations, a sleep with dreaming process occurs. The algorithm followed by the clustering subsystem during one presentation is illustrated in figure 7. If the series of presentations is the first series experienced by the system, there will not yet be any clusters, and information is recorded from each presentation to permit configuration of the first cluster. If any clusters already exist, each cluster is presented with the input and the device activity without recording of any additional combinations determined. If there is γ level output from at least one cluster, this output is passed to the competitive subsystem and no further process occurs. If there is no γ level output, device imprinting occurs in the cluster in which device activity in the β level exceeds a criterion by the largest amount. The γ output from this process is passed to the competitive subsystem. If there is no cluster in which β level activity exceeds the criterion, an unused cluster is imprinted if one is available and if its inputs contain a high proportion of the inputs favoured in the sleep with dreaming process which configured the cluster. If these conditions are not met, no output is generated, but information about the input is recorded for use in configuring an unused cluster in the next sleep with dreaming process. However, if a criterion for activity in the α layer of at least one cluster is exceeded, the information will not be used to configure a new cluster, but in the next sleep with dreaming process the thresholds of β devices in that

cluster will be reduced. Such presentation conditions will eventually be included in one of the existing clusters. This overlay process permits gradual expansion of the similarity conditions of clusters, encouraging a compromise between clusters which are too broad and proliferation of excessively narrow clusters.

The process followed in sleep with dreaming is illustrated in figure 8. In all existing clusters, three processes occur. Firstly, in α, β, and γ layers, unused virgin devices are deleted and new devices configured. The inputs to these new virgin devices are biased in favour of inputs which frequently contributed to firing of regular devices in the same layer of the same cluster in the recent past. This bias effectively reduces the amount of information exchange required between devices. Secondly, if a cluster has responded to less that a minimum proportion of presentations in the preceding wake period, regular α layer device thresholds are reduced. This has the effect of roughly equalizing the proportion of input conditions to which each cluster responds. Minimization of information exchange and roughly equal modules are both requirements for a simple functional architecture as discussed earlier. Thirdly, if the cluster has inhibited the creation of a new unused cluster without producing output in the preceding wake period, thresholds of its β layer devices are reduced.

In addition, a new unused cluster is configured if required. The inputs to β and γ layer devices in the new cluster are selected randomly, but for inputs to α layer devices the random selection is biased in favour of inputs which frequently occurred together in input presentations which did not result in an output in the preceding wake period. This cluster will not be imprinted unless the input condition contains a high proportion of these favoured characteristics.

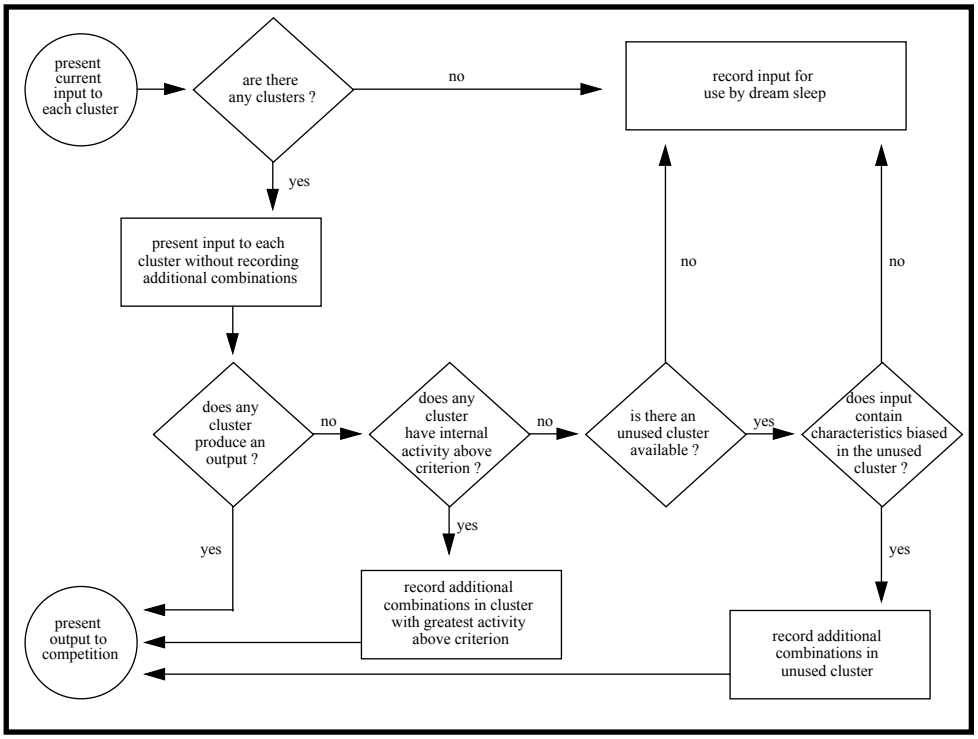


Figure 7. The process followed during the presentation of a single input condition to a simple (single layer) clustering subsystem.

This sleep with dreaming process has been demonstrated to improve the correlation of clusters with categories in the input conditions and to reduce the total device and connectivity resources required [11]. It is called sleep with dreaming because it is an off-line process which uses a rerun of recent experience for functional purposes, as proposed for the function of sleep with dreaming in mammals by Coward [5]. In the simulations the rerun is implicit for implementation reasons.

The process of cluster generation continued until every level input produced an output (with or without device imprinting). Cluster outputs are used by a competition function to determine behaviour. To be effective for this purpose, although individual clusters are ambiguous they should correlate fairly strongly with different major

functionally significant conditions in the input space, and outputs from those clusters should provide enough information richness to allow the functionally simple competitive function to resolve the ambiguous cluster outputs into a high integrity behaviour. In addition, there must be compression of information across a level of clusters, in other words the number of γ outputs from a level should be much less than the size of the input space to the level. There are several problems to avoid in this generation process. One is creation of excessive numbers of clusters, and its opposite of too few clusters. Another problem is excessive use of resources, using more devices and connections than necessary.

5.2 DETAILED ALGORITHMS

5.3.1 Clustering subsystem

The functional effectiveness of a set of clusters for a particular functional task depends upon a number of factors: the specification of the input space; the number of clusters and how they divide up the input space; and the information content of cluster outputs. The process of cluster creation is heuristic, and will vary with the input experience profile, and even with the same profile because of the random element in the device configuration process. The factors which determine functional effectiveness cannot therefore be specified directly, but can be influenced in a number of ways both by design and experience. Information derived from experience can be used to influence input space, number of clusters, and cluster outputs, but the better the design choices (or in a biological brain, the genetically specified parameters) the more effective and less complex the learning process.

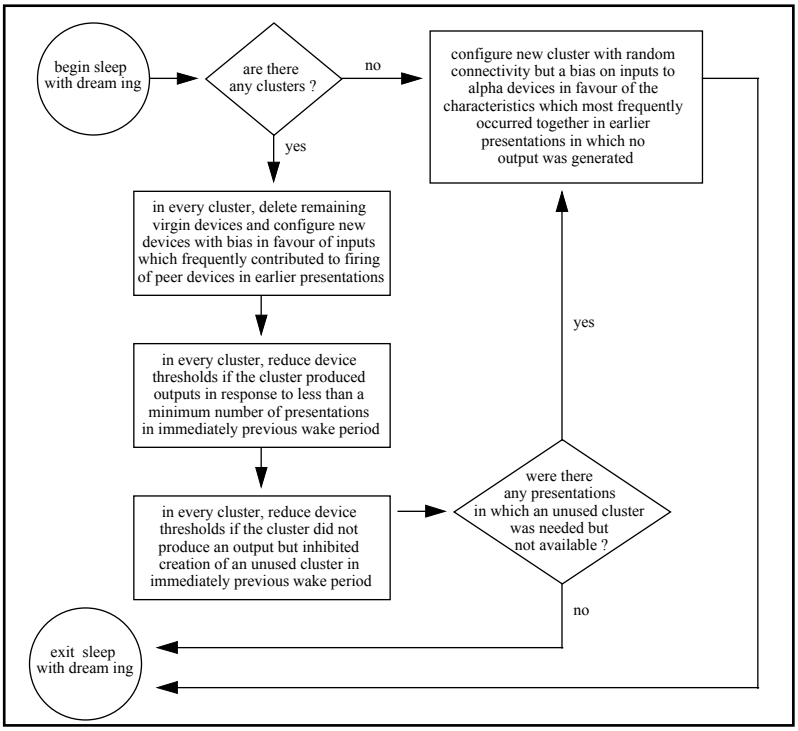


Figure 8. The process followed during sleep with dreaming in a simple (single layer) clustering subsystem.

The initial input space is strongly influenced by design, but α level neurons can in due course choose to accept inputs from sources which have often been active in the past at the same time as already imprinted neurons in the level were active. The number of clusters and the information content of cluster outputs are determined by a number of interacting factors including: 1. The degree of β activity required to generated imprinting in the absence of γ output; 2. The number of virgin neurons in each layer both initially and in subsequent sleep periods, the number of inputs assigned to each virgin neuron, and the statistical bias algorithm applied to the random input selection

process; 3. The number of virgin neurons in each layer which can be imprinted in response to the initial experience and in response to any single subsequent experiences; 4. The thresholds assigned regular neurons at imprinting and how those thresholds are reduced if a cluster only generates outputs in response to very few input conditions; 5. How many presentations must separate the creation of two clusters.

The number of inputs given to a virgin device is determined by the number of random selections made, and by whether duplicate inputs were permitted. As in the simulations described in [10], [11] and [12], no duplicate inputs were allowed, and the number of random selections made were:

Level	α	β	γ
Selections for initial cluster configuration in layer	30% of possible inputs	20	15
Selections for additional virgin devices in layer	20% of possible inputs	25	55

Elimination of duplicates meant that actual input count was somewhat less than the number of selections. For example, with an input space of 1000, the number of input selections for a virgin α device was 300, the actual number of inputs was typically in the range 240 - 280.

The number of virgin neurons configured in initial cluster configuration was typically set at 150 in the α and γ levels and 350 in the β level. During subsequent sleep periods all virgin neurons not imprinted in the previous wake period were deleted, and 40 -100 new virgin neurons were configured to be available in the next wake period depending on demand in the previous wake period. There was no limit on imprinting of α devices to generate first cluster output, and generally all 150 devices were imprinted. The limit on imprinting of β devices was 250, and that number were generally imprinted. If no β devices were imprinted in the first cluster imprinting, the cluster was deleted. The limit on imprinting of γ devices was 1. If the number of α and β devices imprinted at first cluster output was reduced, one effect was a substantial increase in the number of γ outputs in a mature cluster. The limits for device imprinting within a cluster in subsequent presentations were 30 for α and β layers and 1 for the γ layer. In general with the device numbers and input counts specified above the numbers imprinted in the α and β layers were less than these limits.

Imprinting would occur in a cluster in the absence of γ outputs if the number of β devices firing exceeded a few percent of the total number. However, it was found that if a simple percentage (e.g. 5%) were used, clusters tended to proliferate and the number of presentations required to generate a set of clusters which responded to every input also increased. A functionally effective set of clusters matured more rapidly with experience if initially imprinting occurred at a lower proportion which gradually increased with time. In other words, a cluster becomes more resistant to novelty as it matures. The algorithm used to determine the number of β layer devices which had to be active for imprinting to occur was:

$$\text{active } \beta \text{ devices } \geq (1/y) * (x^2/25 - 9x) \text{ with a minimum of one} \qquad (1)$$

where x is the total number of regular devices in the β layer and y is the number of β devices imprinted when the cluster was first imprinted. Initially, this criterion is about 0.2% of the number of devices in the β level, increasing to about 5% of the larger number in a mature cluster. The number of active devices required could be reduced if there were many presentations in which the cluster inhibited creation of a new cluster but was not itself imprinted as discussed below.

In the imprinting process, the thresholds of virgin devices were gradually lowered until a cluster output resulted. The algorithm used attempted first to obtain an output by imprinting only γ devices, then only γ and β, then α, β and γ. The functional role of β devices in determining when imprinting would occur is enhanced by adding inputs to regular devices after initial imprinting. These inputs are provisionally added during sleep from virgin devices in the α level to regular devices in the β level in proportion to firing frequency. The connection became regular if the β level device fired when the virgin α device was imprinted. Most of the inputs which resulted in the β device firing had to derive from regular α devices for the connection to become regular. The additional inputs did not affect the threshold, so the outputs of the β devices in response to previously experienced conditions which produced an output is not changed.

The threshold of a newly imprinted regular device was set initially at one below its new total imprinted input count for α and β devices, and at 20% (rounded down to an integer) below for γ devices. The thresholds of α and β devices could be reduced in sleep if the cluster did not respond to a significant number of input conditions within a

wake period. When initially imprinted, an α device had an input count which varied in practice over a range 20 - 60. In a mature cluster, thresholds had been reduced on average to about 50% of the initial level. At initial imprinting, a β device had an input count which varied over a range 8 - 20. In a mature cluster, some β device input counts could grow to up to 60, and thresholds had been reduced on average by about 15% of original value.

There are thus a large number of adjustable parameters in a clustering subsystem, such as the device numbers and connectivity, and the algorithms determining when learning occurs. Adjustment to these parameters can be by design prior to learning, or some parameters can be modified during learning in response to the system detecting that learning is not proceeding effectively. Ultimately, the algorithms by which such parametric changes occur must be specified by design. Design of a optimal system to learn a specific function would require tuning of all the adjustable parameters. The current values have been developed as a result of extensive simulation experiments with a wide range of input condition types. However, it is of interest to note that functionally effective clusters have been generated for a wide range of category types and input space sizes, with the same adjustable parameter values as described in this section except that the input counts to α devices varies in proportion to the size of the input space.

Competitive subsystem

The implemented competition subsystem was made up of a number of parallel pipes, one for each possible behaviour, as illustrated in figure 4c. Each pipe contained two layers of devices as in figure 4b. The devices are similar to classical perceptrons in having relatively fixed inputs with input weights which vary with learning. In the simulations, a fixed number of pipes were defined which corresponded with the categories present in the input. In a more sophisticated system outputs from clustering could used different combinations and permutations of the available fixed number of pipes.

Devices in the first layer had randomly selected inputs from the γ layer of clusters, and a reference threshold set at 100. The value of the device threshold under different input conditions was derived from this reference value as described below. The default value for the weight of an input was the reference threshold. However, this weight was changed by feedback. If feedback had influenced the weights of previously acquired inputs from the same cluster, the input weight given to a new input was set at the average value of the weights of the most recently imprinted outputs from the same cluster to the same pipe.

A second layer device received excitatory inputs from first layer devices in the same pipe, and inhibitory inputs from first layer devices in other pipes. Inputs to layer two devices were assigned at random. The default weights were the reference threshold value for excitatory inputs and the reference threshold value divided by the number of types of behaviour for inhibitory inputs. If feedback had influenced the weights of previously acquired inputs from the same pipe, the input weight given to a new input was set at the average value of the weights of outputs from the most recently created layer one devices from the same source pipe.

When the presentation of an input to the clustering subsystem resulted in some γ level output, the thresholds of all devices in the first layer of every pipe was varied by the same proportion until 25% of the devices (or all devices with an active input if this was less than 25%) in the first layer of at least one pipe were firing. Using these first layer outputs, the thresholds of all devices in the second layer of every pipe were varied by the same proportion until at least one pipe was generating an output. The behaviour corresponding with the pipe with the largest number of active second layer devices was selected as the behaviour in response to the input object. If more than one pipe had the same maximum output, one was selected at random.

Consequence feedback provided the information that the selected behaviour was either good or bad. If good, the weights of all active inputs to active first layer devices in the pipe corresponding with the selected behaviour were increased, and the weights of active inputs to active second layer devices in the same pipe were increased if excitatory and decreased if inhibitory. Changes in the opposite direction were made to the weights of active inputs to active devices in all other pipes. If consequence feedback were bad, the same changes but in the opposite direction were made in all pipes with active devices. The changes made in response to bad feedback were larger than the changes made in response to good feedback. Input weights to first layer could only be positive, and varied within a limited range, while input weights to the second layer were unconstrained.

In the simple category recognition simulations, there were ten pipes, one for each of the input condition categories. Each pipe had two layers with sixteen devices in each layer. Inputs to devices in the first layer were randomly selected outputs from γ devices in all clusters in a level, and inputs to layer two devices were randomly selected outputs from layer one devices in all pipes. Selection probability for any input to any device was 0.5.

The initial weight assigned to an input was +100 for inputs to layer one devices, +100 to inputs to layer two devices from layer one devices within the same pipe, and -10 for inputs to layer two devices from layer one devices in a different pipe. All device thresholds were set at a high nominal value which was the same for all pipes. When an input to the competitive subsystem was received, layer one device thresholds were lowered from this nominal value until at least 25% of the layer one devices in at least one pipe were active. The nominal threshold was then lowered in the layer two devices until there were active devices in layer two of at least one pipe. The pipe with the largest number of active layer two devices was the category selected. If more than one pipe had the same total output, one was selected at random.

Consequence feedback acted on all active devices in both layers. The weights of active inputs to active devices were increased by 10 times a consequence factor if the category selected was correct, and decreased by 20 times a consequence factor if the category selected was incorrect. Individual input weights to layer one devices could only vary between 0 and 200. In layer two, initially positive weights could not become less than zero, and initially negative weights could not become greater than zero. The consequence factor was a controllable variable which determines the proportion of the initial weight by which a weight changes in response to consequence feedback. It was set at either ±1 or ±5.

There were two types of competitive subsystem runs performed on the outputs from each of the nine clustering run outputs. In both types, the first 4100 object presentations only developed a cluster structure: no outputs or consequence feedback were provided to the competitive subsystem and no action was determined. After this initial period, the system configured the competitive subsystem and generated a category identification for the rest of the presentations. Consequence feedback was applied either for the rest of the presentations or was discontinued after 2000 presentations.

6 RESULTS OF SIMPLE CATEGORY RECOGNITION SIMULATIONS

6.1 CLUSTERING RESULTS

A series of clustering runs of three types were performed. In the first type, there were 76 wake periods separated by sleep. In each wake period ten objects from each of the ten categories were presented, so there were a total of 7600 presentations. All 7600 objects were different. In the second type of clustering run, there were 61 wake periods in which ten objects from each of eight categories were presented, followed by 25 wake periods in which ten objects from each of the ten categories were presented. There were thus a total of 7380 presentations. Again, all 7380 objects were different. The purpose of the two types of run was to investigate the effect of new learning on already learned behaviours. In the third type of clustering run the experience and sleep profiles were identical with the first type, but during the sleep following presentation 4100, functionally duplicated γ layer devices (i.e. devices which were always active at the same time) were eliminated. Configuration of the competitive function did not commence until after that point, and there were therefore no issues with maintaining information context.

Numbers of active gammas in each cluster during final 50 object presentations					
cluster #	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
category of object presented	A F	B G	C H	D I	E J
numbers of gammas active for set of ten objects of type A through J	(0 0 0 0 30) (6 0 0 0 0)	(0 6 0 0 0) (18 0 0 0 0)	(0 22 0 0 0) (6 0 0 0 0)	(0 6 0 0 0) (0 0 14 0 0)	(0 0 0 14 0) (0 0 6 0 0)
numbers of gammas active for second set of ten objects of type A through J	(0 0 0 0 29) (6 0 0 0 0)	(0 6 0 0 0) (18 0 0 0 0)	(0 22 0 0 0) (8 0 0 0 0)	(0 7 0 0 0) (0 0 13 0 0)	(0 0 0 19 0) (0 0 6 0 0)
numbers of gammas active for third set of ten objects of type A through J	(0 0 0 0 24) (5 0 0 0 0)	(0 4 0 0 0) (14 0 0 0 0)	(0 22 0 0 0) (8 0 0 0 0)	(0 5 0 0 0) (0 0 12 0 0)	(0 0 0 15 0) (0 0 6 0 0)
numbers of gammas active for fourth set of ten objects of type A through J	(0 0 0 0 29) (4 0 0 0 0)	(0 5 0 0 0) (12 0 0 0 0)	(0 21 0 0 0) (8 0 0 0 0)	(0 7 0 0 0) (0 0 12 0 0)	(0 0 0 18 0) (0 0 6 0 0)
numbers of gammas active for fifth set of ten objects of type A through J	(0 0 0 0 21) (5 0 0 0 0)	(0 6 0 0 0) (13 0 0 0 0)	(0 21 0 0 0) (10 0 0 0 0)	(0 5 0 0 0) (0 0 13 0 0)	(0 0 0 14 0) (0 0 7 0 0)

Figure 9. For one run, the number of γ devices which were active in each cluster in response to the final fifty presentations of objects from the ten categories are shown. For example, the top left result (0 0 0 0 30) indicates that for the one A object, thirty γ devices were active in cluster five, and no γ devices in any other cluster.

At the end of the clustering runs, the number of clusters was less than the number of categories, varying from five to eight. The information compression factor averaged 6.6. Compression was somewhat higher when functionally duplicated γ devices were eliminated at an intermediate point, and somewhat lower when new categories were introduced part way through a run. Figures 9 and 10 show more detail of the final five sets of presentations of one object from each category A through J in one of the runs. The numbers of γ devices active in each of the five final clusters are shown in figure 9. It can be seen from figure 9 that cluster one generates outputs in response to objects of type F, G, and H; cluster two in response to B, C, and D objects; cluster three in response to I and J objects; cluster four in response to E objects; cluster five in response to A objects. The clusters are therefore partially ambiguous with respect to the categories.

The actual γs active in cluster two outputs in response to the final five B, C, and D objects are shown in figure 10. It can be seen from figure 10 that for categories B and D there are no γs which are only active when an object belonging to the category is present. In other words there is ambiguity even at the device level. However, it can also be seen that there is enough difference between the combinations of active γs to distinguish between objects of the three different categories. As discussed below, differences of this type are adequate for a functionally simple competitive function to develop a high integrity category specific behaviour. The differences are clearer than those between input vectors for different categories as seen in figure 6. There is some functional duplication in the γ devices. In other words, some γ devices are always active at the same time as others. In a functionally complex system this duplication can only be eliminated before the information is used by some other function (another cluster, or a competitive function). Once the information is being used externally, duplication would require massive revision to connectivity which will in general be impractical.

	Gamma devices active in same cluster in response to one object
Gamma device numbers	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Gamma devices active in response to five objects in category B	16 17 18 19 20 21 16 17 18 19 20 21 17 18 19 21 16 17 18 19 21 16 17 18 19 20 21
Gamma devices active in response to five objects in category C	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22
Gamma devices active in response to five objects in category D	4 11 12 14 18 22 4 11 12 13 14 18 22 4 11 12 14 22 4 11 12 13 14 18 22 4 11 12 18 22

Figure 10. For the presentations shown in figure 9, the number of γ devices which were active in cluster two in response to the final five presentations of objects from the categories B, C, and D are shown. There were a total of 22 γ devices in the cluster.

The key conclusion from these results is that significant information compression can be achieved in such a way that the outputs provide more usable discrimination between different categories in the input conditions than the input conditions themselves. It is probable that a greater and more consistent compression could be achieved by tuning the controllable parameters for the type of input space. There is a need for further work in this area. It should be pointed out that if the size of the input space is increased, and the size of the input conditions increases at the same rate, the compression achieved increases faster than the size of the input space. When the input space is expanded to 10,000 under these conditions a compression by a factor of several hundred is achieved, with functionally adequate output information discrimination [12].

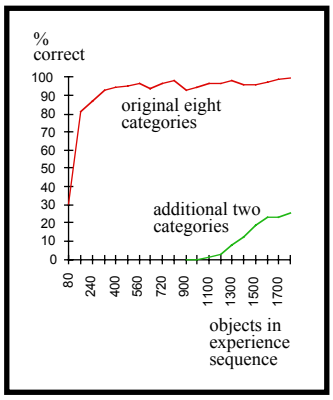


Figure 11. Recognition of categories already learned is only slightly affected when new categories are introduced.

6.2 COMPETITION RESULTS

As described earlier, the competitive subsystem was operated with two different consequence factors, and in addition, consequence feedback could be discontinued after 2000 presentations. These two options resulted in four different competitive algorithms which were applied to the outputs of each clustering run. Two parameters indicating the effectiveness of learning are reported here: one is the average percentage of correct category identifications in the

final 1000 presentations; the other is the rate of improvement in correct identifications when consequence feedback is first applied.

For all types of clustering and competitive conditions, correct recognitions in the final 1000 presentations ranged from 79.7% to 99.7% with an average of 94.9%. None of the controllable parameters made a consistent difference to the recognition accuracy. Recognition accuracy reached 80% within the first 500 presentations after consequence feedback was provided. The magnitude of the consequence factor had no significant effect on the rate of early learning, but when functionally duplicated neurons had been eliminated at an earlier point, early learning was somewhat slower, although the final accuracy achieved was the same.

The ability of the system to retain existing learning when new categories are introduced can be seen from figure 11. Prior to the introduction of the two additional categories, recognition of the eight exceeded 95%. Immediately after the introduction of the categories, there was on average a 5% drop in accuracy of recognition of the original categories. The slower learning rate for new categories reflects the need to expand existing clusters or add new clusters to generate the information for the competitive function to use.

7 APPLICATION OF THE APPROACH TO A MORE COMPLEX PROBLEM

The application of the architecture to a somewhat more complex problem can be understood by consideration of figure 12. The system is presented with a sequence of visual inputs, which are initially received by cluster set 1 in figure 12. These visual inputs model objects of different types using an input space of 1700 binary bits. 600 of these bits provide information on colour, the remaining 1100 on shape. There are six colour categories (red, green, brown, black, white, and clear) and eleven shape categories (cup, coffee, tree, grass, book, pen, paper, water, chair, apple, and car) which exist in the input space, the categories are modeled as probability distributions in the input space as in figure 5, and inputs corresponding with individual objects are constructed by random selection with the appropriate bias. A complete object input is constructed by merging a colour and a shape input. For example, a red cup is the merging of one red input with one cup input. In addition, there are four location categories (home, work, street and laboratory) which are modeled as probability distributions across the full 1700 bit space.

The system is also presented with a sequence of sound inputs, which are initially received by cluster set 3 in figure 12. These sound inputs model twenty-one different sounds (R, G, B, W, C, T, P, H, S, L, D, E, A, I, O, U, F, K, N, CH, and M) as probability distributions in a 2100 bit input space. Initially, clusters are created in cluster sets 1 and 3 in response to sequences of typical inputs are described earlier. The objects presented to cluster set 1 are shapes with a specific range of possible colours for each shape. Thus cups can be red, white or clear; coffee can be brown or black; trees can be green or brown; grass is always green; books can be red, brown or black; pens brown or black; paper always white; chairs green or brown; apples red or green; and cars can be red, black or green. As before, the system never sees two identical input vectors.

Clusters in cluster set 2 are then created by receiving a sequence of inputs each made up of a set of three outputs from cluster set 1. These three outputs are those generated by two objects plus one location. Clusters in clusters set 4 are similarly generated by presentation of a sequence of presentations of sets of three outputs from cluster set 3. The sets of inputs to cluster set 3 are limited to sound combinations corresponding with 28 words (red, green, brown, black, white, clear, cup, coffee, tree, grass, book, pen, paper, water, chair, apple, car, home, work, street, laboratory, eat, drink, walk, sit, drive, read, write). Some words may have only two sounds, in which case the third component in the input vector is empty. Different outputs from cluster set 4 are thus associated with the presence of different words.

Connectivity is then established between γ layer devices in all clusters in cluster set 4 and α layer devices in all clusters in cluster set 1. A series of presentations were made of a word to cluster sets 3 and 4, and simultaneously an object containing the concept referred to by the word to cluster set 1. For example, a word "red" presented at the same time as an object red cup. The input strengths of the connections to α layer devices from γ layer is set proportional to how frequently the corresponding α and γ devices were active at the same time in this word training period.

The two competitive subsystems were then trained. Competitive subsystem 1 was trained by reinforcing association between γ outputs from cluster set 1 and behaviours corresponding with speaking the 21 words (red, green, brown, black, white, clear, cup, coffee, tree, grass, book, pen, paper, water, chair, apple, car, home, work, street, laboratory). Competitive subsystem 2 was trained by reinforcing association between γ outputs from cluster set 2 and behaviours corresponding with the eight actions (eat, drink, walk, sit, drive, read, write and think). These behaviours could also be interpreted as speaking the corresponding words.

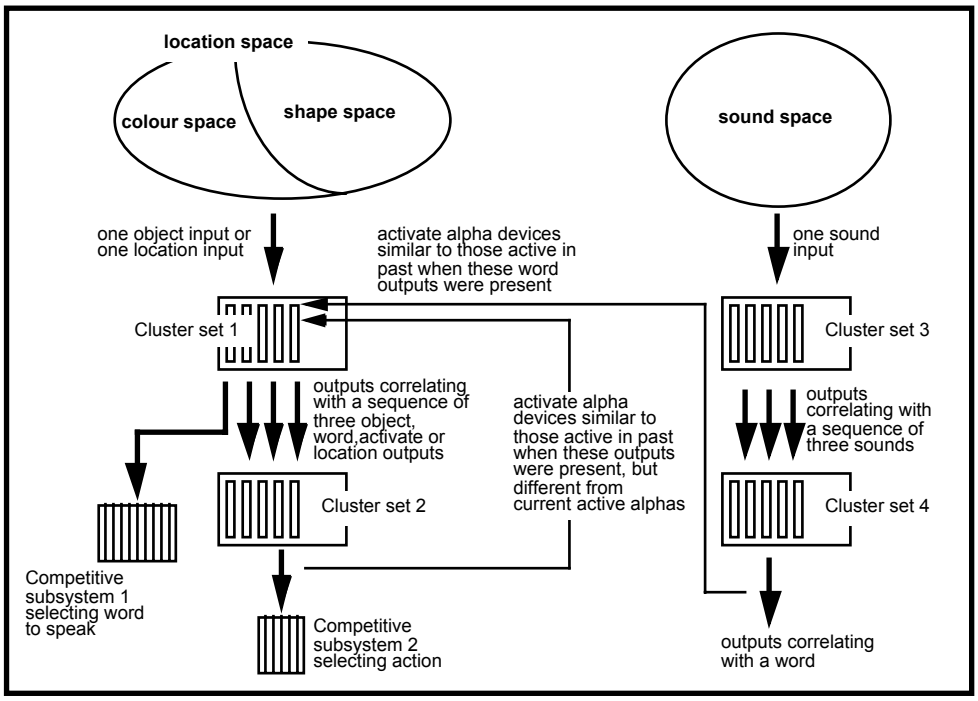


Figure 12. Architecture of a system which can take actions, learn to associate words with visual objects, and imagine objects which are not present. The system takes inputs from a colour input space, a shape input space, a location input space which is the combination of shape and colour spaces, and a sound space. The system organizes its experiences of inputs from these spaces into a hierarchy of repetition and interprets combinations of repetitions as system behaviours.

In the final training stage, connectivity is established between γ layer devices in all clusters in cluster set 2 and α layer devices in all clusters in cluster set 1. The connectivity is similar to that established from cluster set 4 to cluster set 1, but the weights are updated by every experience. These connections activate cluster set 1 any time there is neither a visual object nor a word presented to cluster set 1. However, the α devices activated are those which are not currently firing but have often fired in the past when similar outputs have been present. It can be interpreted as generating an activation corresponding with an object often present in the past at the same time as the object just presented.

The consequence feedback applied to competitive subsystem 2 rewarded the object combination, action combinations shown in table 1.

Combination of input conditions	*Target action*
cup + coffee + home/work/street	drink
cup + water + home/work	drink
book + anything + home/work	read
pen + paper + home/work	write
apple + anything + home	eat
chair + anything + home/work	sit
car + anything + street	drive
tree + grass + street	walk
anything else	no action

Table 1. Conditions under which positive consequence feedback was applied

8 RESULTS WITH THE MORE COMPLEX PROBLEM

Results with the implementation of the architecture illustrated in figure 12 demonstrate that it has the capability to learn to identify colour and shape conditions in its inputs from objects, and to generate the appropriate action in response to groups of objects.

In addition, the outputs generated by words from cluster set 4 can replace one of the object inputs to cluster set 1 and generate the appropriate action. For example, a red-cup object combined with "water" word and home location generated an output from cluster set 2 which was interpreted by the competitive subsystem selecting action as drink. In other words, the word outputs from cluster set 4 generated actions similar to those generated by the actual visual object.

Finally, if cup and home objects were presented without a third object, and the combination generated an activation in cluster set 1 via the connection path from cluster set 2 outputs to cluster set 1 alpha level, this activation generated outputs from cluster set 1 correlating with those generated directly with the presentation of object water. In other words, cup plus home reminded the system of water.

9 COMPARISON WITH OTHER COGNITIVE ARCHITECTURES

There are a number of major differences between the recommendation architecture approach and other cognitive modeling approaches, which can be classified as architectural differences, algorithmic differences, and capability differences.

The coordination of large numbers of interacting functions in the recommendation architecture depends upon providing and maintaining a context for partially ambiguous information exchange. Other investigations of concepts for more complex networks describe the organization of the network in terms of feature combinations (e.g. [1]). Even when modules are organized into hierarchies, different levels in the hierarchy are interpreted as detection of features of different complexity (e.g. [18]). The typical approach of targeting module outputs on specific vectors provides inadequate information richness for partially ambiguous communication and in fact drives such outputs towards unambiguous category identifications. For example, a typical problem attempted by Kohonen Nets is self organization of natural language information [15] in which map outputs indicate individual words and relative position on the map indicates the syntax of the word. The use of device algorithms in which relative input weights are changed makes it difficult to maintain information context as discussed earlier. There is therefore a reliance on exchange of unambiguous feature identifications to coordinate different modules, an approach which [11] has demonstrated leads inevitably to some form of the memory, processing architecture once functionality becomes sufficiently complex.

The recommendation architecture has a hierarchy of functional signals which determine when learning will occur. The only comparable other example is in Adaptive Resonance [4] which makes use of an indication of input similarity as a precondition for learning. This is a single higher level functional signal, but multifunctional, multilevel functional signals comparable with those used in the recommendation architecture have not been developed.

Other approaches do not discuss issues like rough operational equality of functional modules and minimization of information exchange between them, which are explicit in the recommendation architecture and essential for any system which manages a complex functionality.

There are significant algorithmic differences from neural network algorithms used in other approaches. Within a competition subsystem the device algorithms are similar to the widely used perceptron algorithm, although the higher level structure of parallel pipes corresponding with different behaviours, with learned inhibitory signals between the pipes is not used elsewhere. The instantaneous, permanent recording of a currently present information condition used in the clustering subsystem is unique to the recommendation architecture, although in general the processes used in the clustering subsystem could be regarded as a form of competitive learning [14].

In terms of capability, the recommendation architecture derives all its functional knowledge from its experience, unlike approaches like ACT [1] in which such functional knowledge is used explicitly in the design process. The recommendation architecture can handle extremely large and noisy input spaces under the condition that no input ever repeats exactly. New behaviours can be learned without significant effect on previously learned behaviours, an

extensive relearning is not required. This is in contrast with other neural networks in which a learning phase is separated from an operational phase with very limited learning possible in the operational phase.

As a result, other approaches can only be applied either to stable, functionally simple problems or under conditions in which significant explicit functional knowledge is supplied by a designer. An example of a stable, functionally simple problem is the application of neural networks to recognition of features in MRI brain scans [17]. Such problems are algorithmically complex but functionally simple in that system outputs do not dynamically change system inputs in real time [7]. They are stable in that any required change to the problem such as recognition of new types of feature would require an extensive relearning process including previously learned features.

It is of interest to compare the recommendation architecture to the work of Marr on cognitive architectures [16]. In his model of the neocortex, Marr supposed that there is a potentially infinite number of input patterns which can be grouped into a number of different classes. He used a class of type "poodle" as an example. As discussed in work to simulate Marr's model [20]:

"Each single event that represents a different occurrence of a poodle contains certain, but not all, of the features of the poodle class.....Two events that represent different poodles have many features in common - many more than the number of features shared by an event representing a poodle with one that does not. Marr referred to such clusters of events over the set of input fibres as *mountains*. The simplest problem that can be considered is where a network has a single output cell, and it has to learn to distinguish between the inputs from two mountains, occurring equiprobably.....The problem is to find values for the weights and the threshold such that the outputs cell responds to events from one mountain only."

In Marr's approach the implicit objective is to link cell outputs with functionally unambiguous "mountains" of similarity in a sequence of input conditions, and indicate the presence of such mountains by an unambiguous output (in the example, the output from one cell). In the recommendation architecture, the system defines clusters of repetition analogous with Marr's mountains, but these clusters do not correlate unambiguously with functionally relevant categories of input conditions. Instead, the outputs indicating the presence of the ambiguous "mountains" have enough information richness that they can be combined with outputs from other mountains to generate high integrity behaviours (simple examples would be category identifications). Once the objective becomes "to find values for the weights and the threshold such that the outputs cell responds to events from one mountain only" the system is being implicitly designed using unambiguous information contexts, and for a sufficiently complex functionality will either be unrepairable and impossible to modify or will be forced into the von Neumann architecture.

10 CONCLUSION

Electronic implementation of a system with the recommendation architecture demonstrates that such a system can heuristically organize noisy inputs from a very large input space into a hierarchy of repetition. A functionally simple competitive structure can use the indications of repetition to generate appropriate behaviour with high integrity. Such a system can learn how to behave appropriately, using only the inputs themselves and simple correct/incorrect feedback. Once the system has learned a behaviour, later learning of different behaviours has limited effect on existing learning.

A system with the recommendation architecture has been shown capable of simple verbal behaviour: generating words in response to visual inputs; and generating activations in response to words which are similar to the activations generated by the corresponding visual object. The system has also demonstrated simple associative imagination: generation of activations corresponding with visual objects which have often been present in the past at the same time as the current visual objects.

The construction from clustering and competition components of a system capable of simple verbal and imaginative behaviours indicates that the construction process could be extended to systems capable of more realistic cognitive processes. The recommendation architecture thus has substantial potential both for the design of systems to perform cognitive tasks and for understanding of biological systems.

REFERENCES

[1] Andersen, J.R. and Lebiere, C. (1998) *The Atomic Components of Thought*. NJ: Erlbaum.

[2] Andersen, J.A. and Sutton, J.P. (1997). If we compute faster, do we understand better? *Behavior Research Methods, Instruments & Computers Methods, Instruments & Computers* 29 (1), pages 67-77, 1997

[3] Avrunin, G.S., Corbett, J.C., & Dillon, L.K. (1998). Analysing Partially-Implemented Real-Time Systems. *IEEE Transactions on Software Engineering* 24, 8, pages 602-614, 1998.

[4] Carpenter, G.A. and Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. *IEEE Computer*, 3, pages 77-88, 1988.

[5] Coward, L. A. (1990). *Pattern Thinking*. New York: Praeger.

[6] Coward, L. A. (1997). The Pattern Extraction Hierarchy Architecture: a Connectionist Alternative to the von Neumann Architecture. In Mira., J., Morenzo-Diaz, R., and Cabestanz, J. (eds.) *Biological and Artificial Computation: from Neuroscience to Technology*, 634-43, Berlin: Springer.

[7] Coward, L.A. (1999). The Recommendation Architecture: Relating Cognition to Physiology. In Riegler, A. and Peschl, M. (eds.) *Understanding Representation in the Cognitive Sciences*, 91-105, Plenum Press.

[8] Coward, L.A. (1999). A physiologically based approach to consciousness. *New Ideas in Psychology*, 17, 3, pages 271-290, 1999.

[9] Coward, L.A. (1999). A physiologically based theory of consciousness. In Jordan, S. (ed.), *Modeling Consciousness Across the Disciplines*, pages 113-178, Maryland: UPA.

[10] Gedeon, T., Coward, L. A., and Bailing, Z. (1999). Results of Simulations of a System with the Recommendation Architecture. *Proceedings of the 6th International Conference on Neural Information Processing*, Volume I, pages 78-84.

[11] Coward, L.A. (2000). A Functional Architecture Approach to Neural Systems. *International Journal of Systems Research and Information Systems*, 9, pages 69-120, 2000.

[12] Coward, L.A. and Gedeon, T. (2000). Optimization of Architectural Parameters in a Simulated Recommendation Architecture. *Journal of Advanced Computational Intelligence*, in press.

[13] Garlan, D., Allen, R. and Ockerbloom, J. (1995). Architectural Mismatch or Why it's hard to build systems out of existing parts. *IEEE Computer Society 17th International Conference on Software Engineering*. New York: ACM.

[14] Grossberg, S. (1987). Competitive Learning from interaction to adaptive resonance. *Cognitive Science* 11, pages 23-63, 1987.

[15] Honkela, T, Pulkki, V. and Kohonen, T. (1995). Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map. In Fogelman-Soulie, F. and Gallinari, P. (eds.), *ICANN-95 Proceedings of International Conference on Artificial Neural Networks*, 2, pages 3-7. Paris: EC2 et Cie.

[16] Marr, D. (1991). *From the Retina to the Cortex: Selected Papers of David Marr*. Edited by L. M. Vaina. Boston: Birkhauser.

[17] Sabisch, T., Ferguson, A. and Bolouri, H. (1997). Automatic registration of complex images using a self organizing neural system. *IEEE International Joint Conference on Neural Networks*, Anchorage, Alaska.

[18] Sabisch, T., Ferguson, A. and Bolouri, H. (1998). Rotation, translation and scaling tolerant recognition of complex shapes using a hierarchical self organizing neural network. *Proceedings of International Conference on Neural Information Processing* 2, pages 1174-78. Berlin: Springer.

[19] Soni, D., Nord, R.L. and Hofmeister, C. (1995). Software Architecture in Industrial Applications. *Proceedings of the 17th International Conference in Software Engineering*, pages 196-207. New York: ACM.

[20] Willshaw, D., Hallam, J., Gingell. S., and Lau, Soo Leng (1997). Marr's Theory of the Neocortex as a Self-Organizing Neural Network. *Neural Computation* 9, pages 911-936, 1997.

Induction of a Second Language by Transformation and Augmentation

A.Dhunay, C.J. Hinde and J.H. Connolly
Department of Computer Science,
Loughborough University,
Loughborough LE11 3TU,
United Kingdom
{A.Dhunay|C.J.Hinde|J.H.Connolly}@lboro.ac.uk

Abstract

Following Chomsky's conjecture on Universal Grammar (UG) it is assumed that a language can be acquired faster by parameter sing the UG rather than inducing an arbitrary grammar with no initial constraints. Induction of a second language requires the knowledge of a first language therefore the second language being strongly influenced by the structure of the first language learnt.

This project can either induce possible grammar rules from nothing or use an existing syntactic and semantic structure to aid the induction process. In either case the presentation of the Punjabi needs to start with simple requests and build up to more complex requests as any language already learnt will be useful in resolving ambiguities in the possible syntax and semantics of the language to be acquired.

The development of the second language proceeds by entering conjectures, or assumptions, into the truth maintenance system based on evidence gained from presentation of the example sentences and their English equivalents and amplifying the belief in those assumptions as new evidence is presented. Several queries resulting in the same assumption will result in that assumption being held more strongly and so the possible syntax and semantics will build up.

Keywords: Second Language, Punjabi, Chomsky, Augmentation, Transformation, Language acquisition, language induction,

1 INTRODUCTION

There is a large body of work that addresses the problem of language acquisition by computer, although this is mainly focused on acquisition of an arbitrary language.

Chomsky (1981) argued that human's have an ability to acquire their natural language by parameterising a universal grammar rather than acquiring a language from nothing. Acquiring a context free language typically requires the presentation of negative examples as well as positive examples, whereas a regular grammar can be acquired from only positive examples (Fu 1974).

Following Chomsky's conjecture it is reasonable to assume that a language can be acquired faster and easier by parameterising a universal grammar rather than inducing an arbitrary grammar with no initial constraints. Inducing a second language typically draws on knowledge of the first language and so the second language may be strongly influenced by the structure of the first language learnt. In the initial stages of learning Chomsky asserts that the child will have access to the universal grammar to draw on while learning both languages.

This project starts with a natural language data base query system which accepts a subset of English and translates this to SQL. The system is therefore capable of generating the relationship between English and

SQL; given an equivalent Punjabi statement this would have the semantics, (SQL) generated by the English query leaving the Punjabi with the correct semantics associated.
The task would then be to generate the association between Punjabi and the associated semantics.

Two approaches are possible at this stage, either induce the required connection between Punjabi and the SQL from nothing or use an existing syntactic and semantic structure to aid the induction process. In either case the presentation of the Punjabi/SQL pairs needs to start with simple requests and build up to more complex requests as any language already acquired will be useful in resolving ambiguities in the possible syntax and semantics of the language to be acquired.

The latter approach has been taken so Punjabi is to be induced as a second language based on English as the first, or host language.

The heart of the system is an Assumption based Truth Maintenance System (ATMS) which has been used to support several areas from Computer Aided Design and Manufacture (Hinde et al. 1989a & Hinde et al. 1990) to Gesture analysis (McKenzie-Mills et al. 1997), and of course natural language processing (Hinde et al. 1989b). The Truth Maintenance system is essentially a multiple context reasoning system that is able to maintain inconsistent views simultaneously and keep the derivations distinct as required (de Kleer 1987). It also has mechanisms for reasoning about uncertainty (Hinde 1990), although these have been developed substantially since 1990.

The development of the second language proceeds by entering conjectures, or assumptions, into the TMS based on evidence gained from presentation of the example sentences and their English equivalents and amplifying the belief in those assumptions as new evidence is presented. Several queries resulting in the same assumption will result in that assumption being held more strongly and so the possible syntax and semantics will build up.

2 CHOMSKY'S UNIVERSAL GRAMMAR

There has been much previous research that tackles the problem of language acquisition by computer, but the work has mainly focused on the acquisition of an arbitrary language. The main concept of this paper is to look at the problem of inducing a second language via alteration and augmentation.

There are many factors which have motivated this research, mainly Chomsky's conjecture on Universal Grammar and also personal experience of being bi-lingual and acquiring a language through natural methods.

Chomsky believed all human beings shared their knowledge of language, regardless of which language it is – Universal Grammar (UG) is their common inheritance. He believed that UG is present in every child's mind as a system of principles and parameters which are set in order for the child to acquire their language fluently. Setting the principles allows the child to constrain grammar and reduce learning space from a variety of possibilities hence making language acquisition easier whereas setting values to parameters allows the child to differentiate from one language to another – i.e. setting head parameters to head first or head last. (Chomsky,1981)

English speakers set head-parameters to head first
The London train arrived at platform 5
(VP consists of V as the head first.)

Punjabi speakers set head-parameter to head last.
London di train panjvi platform thai ponchi hai
London 's train fifth platform on arrived
(VP consists of V as head last)

3 WHO ARE THE MAIN USERS OF UG ?

According to Chomsky, the main users of UG are children who acquire their first language naturally. Children will start off at the initial state (S_0) where they will have no knowledge of a language and after having access to the UG, will slowly progress to the steady state (S_S) where they can efficiently use the native language. Children are born with the knowledge of learning a language but need environment and

experience for it to mature. An example of this maturity given by Chomsky is a 'Seed'. A seed has the potential to turn into a plant, but needs the environment (sun and water) to succeed. In the same way children have the potential from the birth to acquire a language but need the environment to mature and guide it.

During the process of learning a language, children will only come into contact with positive evidence. They will only hear grammatical correct sentences from parents, adults, society – but Chomsky believed that this is not enough for them to acquire a language – children also need negative evidence to be able to acquire a context free grammar. Children need to be able to make mistakes and correct them.

4 DO ADULTS STILL HAVE ACCESS TO UG WHILST LEARNING A SECOND LANGUAGE?

Adults usually learn a second language not for pleasure, but mainly for educational or employment purposes, and because of this reason, it is important to see how these second languages are learnt.

The word 'second' can mean a second language subsequent to the mother tongue therefore can mean the learning of the third and fourth language or it could be referred to a 'foreign' language which is learnt naturally as a result of living in the country where it is spoken as a native tongue.

Basically, 'Second Language Acquisition' is learning a second language other than the mother tongue inside or outside of a classroom.

Chomsky looked at second language acquisition in equivalent terms to first language acquisition. Whereas children start at S_0 and slowly progress to S_S state, adults learning a second language would start at S_1 state (because they already have knowledge of a first language) and slowly progress to S_t state (which Chomsky believes will never be equivalent to the S_S state in the first language and will be different to each individual depending on how they have learnt their second language).

Chomsky examined 3 ways UG can aid in the acquisition of a second language:-

*'**direct access** to UG'*- where a child would use UG in the same way as learning their first language. The parameters would be set differently to the first language and this procedure is usually used by bi-lingual children acquiring 2 languages at once.

***'indirect access** to UG'* – where the parameters set for the first language are used as a springboard to learn the second language. An example of this method is Surjit who learnt Punjabi as her first language and then later learnt English as an adult. Even after years of contact with the English language, her native tongue still influences her English accent , syntax and grammar. (Dhunay, Hinde and Connolly 2000)

*'**no access** to UG'* – where many adults learn a second language at a later stage in life via non-natural methods – ie books, tapes, classes, imitation etc.

One of the reasons why adults are less likely to achieve their second language at the same level as the first language could be due to Lennenbergs theory on 'Critical Period Hypothesis'. Lennenberg believed that a child can learn more than one language before the critical period (before the age of 12) and after that age it becomes more and more difficult to learn a language. There is not clear evidence on this hypothesis because it's hard to find a first language learner in their later teens to compare the difference in language proficiency.(Cook, 1998)

Some early research (Ferguson 1975, Meisel 1975) undertaken in the area of native speakers speech to foreigners indicated that linguistic environment might prove to be a very important area in the difference between first and second language acquisition. They believed that speakers of English, French, German and Finnish tended to switch to ungrammatical sentences when addressing Non Native Speaker's. This process was known a 'Foreigner Talk' (Ferguson 1975) and was the result of three main processes: omission, expansion and re-arrangement.

Omission included the deletion of articles, copulas, conjunctions, subject pronouns and inflectional morphology whereas expansion involved the addition of analysed tags to question – i.e. yes, no, okay ? Other reasons for ungrammatical sentences are, insertion of subject pronoun – *you* before imperatives and also replacement which involved forming negatives with 'no' plus the negated item i.e. 'no like'.

The role of environmental factors in first or second language acquisition has been of great interest to linguists. Linguistic environment is not the only factor that determines second language acquisition. There

are many more – for example individual age, language aptitude, social-psychological factors, personality etc. All normal children, under normal circumstance will acquire their native language – some faster than others, but language mastery is not that simple in Second Language Acquisition?

5 AUTOMATED LANGUAGE ACQUISITION

Over the last fifteen years, the field of language acquisition has seen great changes. There has been a great interest in the language acquisition of young children and in the area of phonology, syntax, semantics and lexicon. (Clark 1993, Gleitman and Landau 1994, Jusczyk 1996). Together with these interests, there has been a change in the computational side of language acquisition. These included a shift from hand-crafted grammars to grammars learned automatically from corpora of natural occurring language (Armstron 1994, Bahl et al. 1993, Joshi 1991).

According to Shapiro(1992), cognitive scientists tend to build computational models that show how children learn because they believe that building these models allows them to achieve a degree of explicitness that is impossible to achieve when working with pen and paper. These models allow the scientist to compare the results in greater depth.

One of the earliest applications to acquire language acquisition was developed in 1969 and known as 'Teachable Language Comprehender' by Quillian. This system learnt how to understand English text and was developed as a theory of language understanding rather than language acquisition.(Quillian, M.R., 1969) The first computer simulation of language acquisition was written by Kelly, K.L.,(1967) in a linguistic manner. It was able to learn simple grammars through experience. This model was given basic syntactic categories and it had to guess whether sentences, which were inputted, were grammatically correct or incorrect. It was intelligent enough to ignore sentences it did not recognise and only dealt with the ones that were clear.

Another application, which acquired language, was developed by Harris, L.R., (1974, 1977) where a robot was able to learn a subset of the English language from a teacher. Others included LAS (Language Acquisition System) by Anderson, J.R., (1977). This system, together with that of Harris, failed to learn in the same manner a child would learn. (Shapiro, S.C.,1992). The LAS system did manage to learn to generate and understand sentences but it was similar to that of an adult learning a second language rather than a child learning its first. The reason for this was that the system had to already be programmed with all the concepts before the mapping of words and concepts can occur. The original idea was to develop a language acquisition system which made the same errors as a child and would correct these errors as a child would whilst learning.

Cognitive models of language acquisition are not only concerned with whether or not a computer can learn a language but it is also concerned with whether it can learn a language the same way a child does. There has been a wide range of research by psycholinguists on how children learn their first language, but attention has been focused on errors children make (and don't make) and how they correct them. An example of a typical error by children whilst acquiring language is overgeneralization of plurals of nouns and of past tense of verbs in English, therefore a computer system acquiring a language must also make mistakes as a child would and also be able to correct them after further learning has occurred. Models on language acquisition are based on data which are collected via psycho-linguistic study.

A connectionist model looks at language acquisition in a different way. A model developed by Rumelhart and McClelland looks at the way a child learns past tense of verbs in English. This is an area of language acquisition where many children make the same common mistakes. For example - looking at the verb '*go*'. Children at an early stage of learning are usually taught the word '*went*' and they use it grammatically correctly. At a later stage they start to use '*goed*'. Gradually a child will learn that this is grammatically incorrect and will change its vocabulary accordingly - but the interesting factor is that both words are in the child's vocabulary for a particular period until the correct form is learnt.

5.1 SELFRIDGE (1982) MODEL (CHILD)

Selfridge's model shows the 3 stage learning process:

Correct form ⟶ *Overgeneralised form* ⟶ *Correct form*

but was still unable to explain the stage where both correct and incorrect verb is in the child vocabulary.

Selfridge's (1982) model, known as the CHILD model looks into how semantics is used in the acquisition of language. Firstly the model was able to understand only commands with the output as actions, but after further work it was then able to generate language and the output was in the form of verbal communication.
The input to the model is in the form of adult sentences and the output is a childlike response.

Selfridge believes that concepts exist before a language is learned. His data was collected from research undertaken from a child named Joshua. Joshua was learning to understand a language he had no knowledge of. Selfridge argued that learning a language is the same as learning to fill slots within a sentence in relation to other words in the sentence.

Selfridge's model begins at the same stage where a child begins with no language skills. The basic idea of the model is to have poor language skills, gain experience, make errors whilst learning and correct those errors in order to understand, It first learns to understand single words and then goes on to learning syntactic information and then use this information with other words and eventually extend the knowledge up to an adult level.

5.2 LANGLEY'S (1982) MODEL (AMBER)

Langley's (1982) AMBER is a model which acquires language through a process of error recovery over a gradual period of time. The system does not understand and has a very small amount of knowledge about the world therefore not answering any questions. The input to the system includes adult sentences together with the meaning representations and an idea of the main topic which the system learns to mimic in all adult like complexities. The system outputs a sentence and then compares it with the original adult one. If the sentence is not the same, the system will adjust its output to account for a difference between the two. The main concept is to slowly describe the main topic of the sentence.

According to Langley, the learning process is slow but accurate, the reason for this is that it starts off with learning one-word utterances and slowly deals with suffixes, prefixes etc. Some rules which are acquired can later be used to learn more complex constructions. According to Langley, the rules are learned many times before the system realizes that it is the same as a previous learned rule. Like children's learning behaviour, the more complex the input, the more time it takes for learning to take place.

5.3 HILL'S (1986) MODEL

She believes that this phenomenon can't be explained if a child's lexicon contains rules for past tense for every verb because then the rules would have to be learned and unlearned many times.

Like Langleys model (AMBER), Hill's model also follows the acquisition of syntax but is unable to learn suffixes, prefixes and morphemes. It has the ability to understand and generate language at a level of a 2-year-old and relies on word order in the same way a child does. The model is based on the results of the study taken from a child named Claire who was learning a language. Input to the model is in the form of adult sentences and the outputs are childlike sentences which repeats and responds to adult input. The system itself consists of a dynamic data structure which encodes the child's grammar and knowledge of the child. Learning process of this model is dynamic therefore if the same input is given twice, two different sets of grammar schemata and additional lexical information are learned.

Hill's model has the capability to repeat adult sentences correctly and also can answer questions but unfortunately it does not progress beyond the level of a two-year-old.

One important aspect of language acquisition is the acquisition of lexicon. Siskind (1990) developed a system which acquired core meanings of words. His system was not motivated by psycholinguistic data, but was based on how a child learns its word meanings in the early stages of language acquisition.

Input to the system consists of sentences which describe visual pictures. The system itself includes a parser, linker and an inference machine. The parser used a context free grammar to produce a parse tree (syntax), whilst the inference machine uses the pictures to produce the semantic structure and finally the linker links both the syntactic information and semantic information to produce the output. The output of the model is a lexicon which allows the sentence to describe the picture.

5.4 MACWHINNEY'S MODEL

MacWhinney's model (Shapiro, 1992) generates language and is known as the competition model. The model looks at different linguistic data from different languages. MacWhinney believes that a child will try to learn words for meanings it is trying to express therefore the model looks at word order and word positions as pre-verbal and post-verbal.

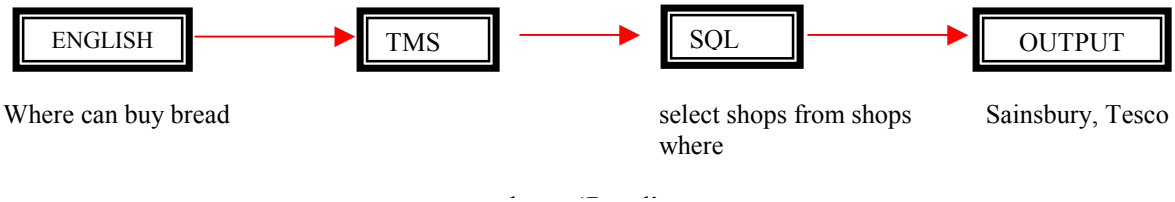
Like the other model, this model also begins with single word utterances and then gradually proceeds to acquire more words. The model is given very detailed meaning representation to deal with the input sentences and is anticipated that it has the capability to deal with touch problems like relative clauses.

The problem with the cognitive models described above is that they expect the child to have acquired some knowledge of the language before they start to learn. Models do make errors of overgeneralization like children do but they do not rely on overt correction of errors. Selfridge's CHILD lacks a system of weighting factors (weights are attached to rules to indicate the relative strengths and weaknesses) which the other models do possess therefore it learns immediately rather than gradually. Selfridge's model starts off with no language skills, but gains experience through errors and learns to correct them as a child would. It first learns basic words, then syntactic information. On the other hand, Langley's model learns to utter sentences at the same level of an adult. It develops the ability to learn morphemes, suffixes etc. through gradual but slow learning. It lacks the ability to understand because it just copies adult utterances therefore it does not need to acquire world knowledge due to the fact that it will not be used to answer questions.

Learning ideas of Langley, Hills and MacWhinney's models are closely related to connectionist models as it requires a teaching phase. Selfridge's and Hill's model rely on generalization (going from a specific plan to a more general plan) and Langley's model uses discrimination (starting from overly general idea and then finding and adding missing conditions) whereas MacWhinney model both generalizes and discriminates.

6 ECHO SYSTEM

6.1 INITIAL STATE



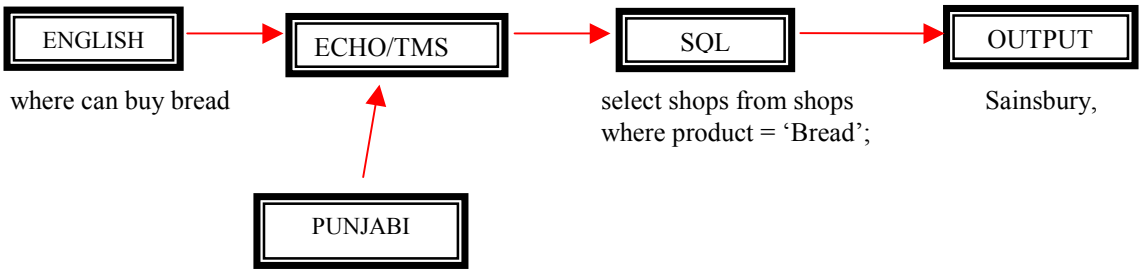
This research project starts with a natural language database query system based on accepting English and translating it into SQL statements. A natural language interface to a database is a system that allows the user to access the information which is stored inside the database by simply requesting it via some natural language which does not necessarily have to be syntactically correct.(i.e. English).

The heart of the system is a Truth Maintenance System (TMS) which has been used to support several areas from Computer Aided Design and Manufacture (Hinde et al. 1989a & Hinde et al. 1990) to Gesture analysis (McKenzie-Mills et al. 1997), and of course natural language processing (Hinde et al. 1989b). The Truth Maintenance system is essentially a multiple context reasoning system that is able to maintain inconsistent views simultaneously and keep the derivations distinct as required (de Kleer 1987).

The query passes through the TMS which derives a number of possible answers which are assigned a measure of belief. The highest ranked queries are presented to the user in the form of SQL statements.

One of the main advantages of NLIDB (Androutsopoulos et. al 1995) is that the user does not have the problem of learning artificial query languages such as SQL which take up considerable time and effort. Menu-based applications are also available but to some extent the occasional user still needs some training on some areas, for example, linking frames and selecting restrictions from menus.

NLIDB's can only deal with a subset of natural language which is not always obvious to the user and help is not always available. (Tennant et. al 1983; Hendrix, G., 1982; Cohen,1991, Sidhu and Hinde, 1997.). It is impossible to try and implement a system which could deal with every aspect of NL - due to new words being invented and words being disused.



As mentioned earlier, this project starts with a natural language database query system. The system is capable of generating the relationship between English and SQL; given an equivalent Punjabi statement this would then have the semantics (SQL), generated by the English query, leaving the Punjabi with the correct semantics associated. The task would then be to generate the association between Punjabi and the associated semantics.

The research can take two approaches; it can either induce the required connection between Punjabi and SQL from nothing or use an existing syntactic and semantic structure to aid the induction process. In either case the presentation of the Punjabi/SQL pairs needs to start with simple sentences and build up to more complex sentences as any language already acquired will be useful in resolving ambiguities in the possible syntax and semantics of the language to be acquired.

The development of the second language will take place by entering conjectures, or assumptions, into the TMS and amplifying the belief in those assumptions as new evidence is presented. Several queries resulting in the same assumption will result in that assumption being held more strongly and so the possible syntax and semantics will build up.

6.2 Rules of Second Language Acquisition used by Echo

The Echo system has followed one of the factors proposed by Chomsky on 'Indirect Access to UG'. As an adult learning a second language using the first language as a springboard, the Echo system also uses the first language – English as a springboard to acquire the second language– Punjabi.

Echo has looked at three main rules in-order to acquire the second language – Re-arrangement , Omission and Addition. In terms of access to the Universal Grammar, there is no need for access to the UG to perform re-arrangement

Omission similarly does not require access to the U.G. and so indefinite and definite articles can be omitted as they are not present in Punjabi syntax.

Addition requires access to the UG as there is no other source for the additional syntactic forms.

6.2.1 Re-arrangement

The re-arrangement rule takes an English and Punjabi phrase and re-arranges each word to a different position within a phrase till the right Punjabi structure is found. It is apparent that Punjabi and English have different word orders. In order for Echo to learn the Punjabi sentence, it must take the English sentence and swap the word order around several times to find all possible grammatical structures of the sentence and eventually it will find the structure that fits the Punjabi phrase.

For example – a simple English sentence structure is

Sentence → eat the bread [verb, article, noun]

Whereas the Punjabi structure is:

Sentence → roti kha – [noun, verb] (bread eat)

The Echo system with go through the following steps of re-arrangement: (bearing in mind that it will delete the article with the 'omit' rule, see later)

Step 1	[verb, article, noun]
Step 2	[verb, noun]
Step 3	[noun, verb]

Here it can be seen that both the correct English and Punjabi phrase structures are present, however they will generally have different levels of belief associated with them.

6.2.2 OMISSION

The omission rule in Echo takes out a selected word from a phrase which does not occur in the target language. It does this by using the add_omit rule:

```
add_omit([indefinite_article,noun_phrase],[noun_phrase],20).
```

The indefinite_article is deleted from the noun_phrase and the 'noun_phrase' alone is parsed together with the Punjabi 'noun_phrase'. An example of a Punjabi sentence where this may be used is:

English: The shops
Punjabi: Dukaana (shops)

6.2.3 ADDITION

In Echo, the addition rule adds a word onto a phrase. Currently, if there is a single word in English that corresponds to two words in Punjabi, it will add on an adjective at the beginning of the phrase to make it equivalent to the Punjabi sentence. For example:

English: train (noun)
Punjabi: rail gaddi (adjective, noun)

Echo will see that Train is a noun therefore one of the words 'rail' or 'gaddi' has also got to be a noun. It will then add an adjective to the phrase because the adjective complements the noun.

English noun
Punjabi adjective, noun

At this stage of acquisition, Echo does not know which word in the Punjabi phrase is a 'noun' or an 'adjective' therefore it will look at all possibilities by stating that 'Gaddi' and 'Rail' could both be either nouns or adjectives.

noun(gaddi)	1.55
adjective(rail)	1.55
adjective(gaddi)	1.55
noun(rail)	1.55

This also reflects results found in the report written on Surjit (Dhunay, & Hinde, 2000). In the report, it is evident that Surjit, being a native Punjabi Speaker, omits articles when speaking the English language. i.e.

```
Line 73 : living altogether in  village
                               ^
                  Missing article 'the'.
```

Punjabi does not use articles. Surjit learnt English at as an adult and so would not have access to the UG. She unconsciously leaves them out whilst translating from Punjabi to English as the concept of articles is not available at the subconscious level.

6.3 THE LANGUAGE ACQUISITION PROCESS

The acquisition process takes a sentence from the host language and an equivalent sentence from the target language and uses the correspondence to suggest hypothetical lexical and grammatical structures, which are assigned levels of belief.

bread roti

This example results in knowledge that the English word "bread" is a noun and the highest grammatical structure is a noun_phrase.
Echo derives the following two hypotheses. The belief system takes a range of values from 0 to 100.

noun_phrase ::= noun 2
noun(roti) 1.80

This knowledge is internalised and the following sentences are presented

eat the bread roti kha (bread eat)

In view of the fact that the system already suspects that "roti" is a noun, this constitutes further evidence that "roti is a noun but also includes the possibility that it might be a transitive verb, at a lower level of belief. The system mistakenly derives the rule:

verb_phrase ::= transitive_verb, noun_phrase

The syntactic rules derived are:

verb_phrase ::= noun_phrase, transitive_verb 1.998
verb_phrase ::= transitive_verb, noun_phrase 2

So the inherited English rule has a marginally higher level of belief associated with it as the system has a bias towards inheriting rules directly and initially penalises rearranged or otherwise edited rules.

noun (roti)	1.88
trans_verb (roti)	1.80
imperative_verb(roti)	1.80

The above shows that confidence in the hypothesis that "roti" is a noun has increased.

Sell the bread roti bech (bread sell)

This example shows the strength of belief in the rule:

verb_phrase ::= noun_phrase, transitive_verb
has increased to beyond that for
verb_phrase ::= transitive_verb, noun_phrase

indicating that evidence from the equivalent sentences is enabling the system to re-inforce the correct rules.

verb_phrase ::= noun_phrase, transitive_verb 2.09
verb_phrase ::= transitive_verb, noun_phrase 1.89

noun [roti]	1.96
trans_verb [roti]	1.89
imperative_verb[roti]	1.80

As can be seen from the results table above, there has clearly been an increase in the ratings in words which have re-occurred on several occasions. In this example- the word that has been used repeatedly is 'BREAD'.

Echo started off with no Punjabi knowledge, but after being presented with a few Punjabi words and short utterances it has learnt that a Punjabi sentence is different to an English sentence and the headedness in a Punjabi 'verb_phrase' is head last.

It also has come to learn that a 'roti' is a noun; this acquisition has taken place by the TMS increasing the belief in assumptions as new evidence is presented. Results that have had the same assumptions has caused that rating to be held more strongly, thus possible syntax and semantics have built up.

The syntactic structure of the English language has previously been programmed into Echo, therefore the English language can be used in two different ways.

Firstly, Punjabi can be induced from English as a first language. A universal grammar to work from is unavailable, therefore, English can be seen as the UG and Punjabi being derived from it in similar terms as the acquisition of a first language, i.e. setting parameters for the target language by having full access to Chomsky's Universal Grammar. Secondly, we can assume that English has already been acquired by having full access to the UG and Punjabi is now being induced from English as a second language and having indirect access to the UG parameters that have already been set by the first language, hence, the source language affecting the target language. In either case, the acquisition procedure has to start with simple requests and slowly build up to more complex requests as any language being acquired would go through.

Giving it a simple sentence, the Echo system manages to differentiate several different phrase structures.

Example:

Product of shop is bread	Dhukaan	di	cheez	roti	hai	
	Shop	of	product	bread	is	

The phrase used here is a comparison phrase which includes a noun phrase and also a possessor phrase. In order to distinguish between the two possible interpretations of "dhukaan di cheez", "shop of product" and "product of shop" it is necessary to associate the semantics of the words and phrases; knowing that "dhukaan" and "shop" are equivalents from previous sentences would enable the differentiation to be made. Syntactically they are very similar.

The rule that is indicated here strongly is:

comparison_phrase ::= noun_phrase, noun_phrase,comparator 12.08

whereas the English rule is:

comparison_phrase ::= noun_phrase, comparator, noun_phrase 8.74

In order to achieve this differentiation and to establish the rule the system needed to know that dhukaan, cheez and roti were all nouns. Knowing that just "roti was a noun enabled the following strength to be established. Existence of articles in English is a substantial barrier to learning from a host language without articles. The Echo system can easily omit the articles when deriving Punjabi from English but unless some additional information is provided, analogous to having access to the UG the system can never acquire the concept of either the definite article or the indefinite article. Giving Echo access to some simple addition rules illustrates the point. Deriving English from Punjabi using the sample sentences "dhukan" and "the shop" results in the system deriving several hypotheses. In particular:

noun_phrase ::= definite_article, noun 2.34
noun_phrase ::= indefinite_article, noun 2.34

show up as two possibilities amongst:

noun_phrase ::= noun, indefinite_article 1.63
noun_phrase ::= noun, definite_article 1.63
noun_phrase ::= noun, adjective 1.63
noun_phrase ::= adjective, noun 2.34

This was after presenting evidence suggesting that shop might be a noun. The choices available are reinforced according to evidence gained and so it is possible to ascertain which grammar rules are the most likely to form the basis of the new language. The ability to distinguish between indefinite and definite articles is something that is not clear that the system will be able to accomplish without recourse to the semantics. In terms of syntax acquisition the distinction appears unnecessary.

7 CONCLUSION

There are two reasons why Punjabi is a preferred language to acquire. The first academic reason is that there are several important differences between the syntax of Punjabi and the syntax of English, the "headedness" is different; comparisons in English place the comparator between the objects whereas Punjabi places the comparator after the two objects; English uses "infix", Punjabi uses "postfix". The second reason is that the first author is bilingual in English and Punjabi.

A recent paper by Sidhu and Hinde (1997) indicates that one of the bottlenecks of setting up a natural language query system is the introduction of a new database or a new language. The possibilities resulting a successful project include being able to produce a much simpler system of introducing a new language requiring less time by a bilingual data base administrator being replaced by time from a bilingual user; the system would then acquire the language used by the user rather than that employed by the data base administrator. Multi-nationals would be able to provide management information system tailored to the local language or dialect.

REFERENCES

[1] Anderson, J.R., 1977, Induction of ATN, Cog. Sci 1, 125-127

[2] Androutpoulos. I., Ritchie, G.D., Thanisch, P., 1995, Natural Language Interfaces to Databases – An Introduction. Journal of NL Engineering, Cambridge University Press.

[3] Chomsky, N., 1981, Lectures on Government and Binding, Dordrecht, Netherlands: Foris Publications.

[4] Chomsky,C.,1969, Acquisition of Syntax in Children from 5-10, Cambridge, MA:MIT Press.

[5] Cook, V,J., 1988, Chomsky's Universal Grammar, An Introduction. Basil Blackwell Inc., 432 Park Avenue South, Suite 1503, New York, USA.

[6] de Kleer, J., 1986, An Assumption-based TMS. Artificial Intelligence Journal, Vol. 28, pp. 127-162.

[7] Dhunay, A., Hinde,C.J., & Connolly, J.H., 2000, An Analysis of English Spoken as a Second Language. Internal Report Number 1050, Computer Science Department, Loughborough University, Leicestershire, UK.

[8] Ellis, R.,1997, Second Language Acquisition, Oxford Introduction to Language Study, Oxford University Press.

[9] Epstein, S.D., Flynn S, Martohardjono, G., 1996, Second Language Acquisition, Theoretical and Experimental Issues in Contemporary Research. Behavioural and Brain Sciences, 19. 677-758.

[10] Fu, K.S., 1974, Syntactic methods in pattern recognition, Mathematics in Science and Engineering, Publisher New York;London:Academic Press,Vol 112

[11] Gibson, E., & Wexler, K., 1994, Triggers, Linguistic Inquiry, 5, 47-454

[12] Harris, L.R., 1974, Natural Language Acquisition by Robot, Technical Report TR 74-1, Dartmouth College, Dartmouth, New Hampshire

[13] Harris, L.R., 1977. A system for Primitive Natural Language Acquisition, Int. J. Man-Machine Stud.9. 153-206.

[14] Hendrix, G., 1982, Natural Language Interface(panel). Computational Linguistics, 8(2):55-61, April-June 1982.

[15] Hinde, C.J., 1990, Loughborough University Manufacturing Planner, Business Benefits of Expert Systems, sponsored by the S.G.E.S., Sept. 1990.

[16] Hinde, C.J., Bray, A.D., Herbert, P.J., Launders,V.A. & Round, D., 1989a, A Truth Maintenance Approach to Process Planning. in ed Rzevski, G., 1989, Artificial Intelligence in Manufacturing, Computational Mechanics Publications: Southampton Boston Springer-Verlag: Berlin Heidelberg New York London Paris Tokyo.pp 171-188.

[17] Hinde, C.J., Lawson, R.J. & J.H.Connolly, J.H., 1989b, Natural Language: the ultimate user-friendly interface. San Francisco 1989 Interex H.P. users Conference.

[18] Kelly, K.L., 1967, Early syntactic acquisition, PhD Dissertation, University of California, LA.

[19] Langley, P., 1982, Language acquisition through error recovery, Cog. Brain Theory 5, 211-255.

[20] McKenzie Mills, K. and Alty, J.L., 1997, 'Investigating the Role of Redundancy in Multimodal Input Systems, Gesture and Sign Language in Human-Computer Interaction, Proceedings of the Gesture Workshop 1997 , Wachsmuth, I. and Frohlich, M. (eds), Springer Verlag, Proceedings of Gesture Workshop 1997 , Bielefeld, Germany, 1997, pp 159-171, ISBN 3-540-64424-5 .

[21] Quillan, M. R., 1969, The teachable language Comprehender: A simulation program and theory of language, Communication. ACM 12, 459-476.

[22] Selfridge, M., 1982, Inference and learning in a computer model of the development of language comprehension in a young child, in W. Lehnert & M. H. Ringle, eds, Strategies for NLP, Hillsdale, N.J. pp299-326.

[23] Shapiro, S.C., 1992, Encyclopaedia of Artificial Intelligence, 2nd Edition, John Wiley & Sons, Inc, Canada.

[24] Sidhu, J. and Hinde, C.J., 1997, An Analysis of the use of natural language processing systems in business, Behaviour and Information Technology, Vol 16 No 3, Taylor and Francis, pp 145-157.

[25] Siskind, J. M., 1990, Acquiring core meanings of words, represented as Jackendoff-style conceptual structures, from correlates streams of linguistic and non-linguistic input. Proceedings of the 28th annual meeting of the association for computational linguistics, University of Pittsburgh, Penn.

[26] Tennant, H. R., Ross, K. M., Saenz, M., Thompson, C. W., and Miller, J. R., 1983, Menu-Based Natural Language Understanding. In Proceedings of 21st Annual Meeting of ACL, Cambridge, Massachusetts, pages 151-158.

Modeling Reaction Times with Neural Networks using Leaky Integrator Units*

Thomas Liebscher†
University of Potsdam
Institute of Linguistics
P.O. Box 601553
D - 14415 Potsdam
Germany
liebsche@rz.uni-potsdam.de

Abstract

A leaky integrator unit is presented that is grounded on the idea of cell assemblies proposed by Hebb (1949). Their time-course of activation is characterized by a gradual increase or decrease towards an asymptotic value. It is argued that this captures the biological reasons for reactions times. A back-propagation procedure is derived that provides a method to learn weights as well as leakage parameters in networks made of leaky integrator units. The memory of these units is exponentially decreasing and their effective time-window is decreasing with the leakage term. A network of leaky integrator units is used to simulate data from a reaction time experiment. The results show a good fit of the data that becomes worse with increasing time.

Keywords: Neural networks, reaction times, leaky integrator units, back-propagation, time-dependend networks, cell assemblies, symbolic representation

1 Introduction

The temporal characteristics of neural networks have most often been used in the domain of time-series analysis (e. g. Weigend and Gershenfeld (1994)), where the goal is to predict the time-course of activation from information provided earlier. They have much less been used to model human reaction times, where not a specific time-course is to be modelled but some unspecific accumulation of information that leads to a reaction. This accumulation is unspecific with respect to location where and how it happens in the nervour system. It is influenced by at least three temporal components: the velocity of nervous signals within each neuron, the velocity of nervous signals between neurons through the synaptical gap and the speed of accumulation of incoming signals in a neuron (compare Abeles (1991)).

One of the first network models that took into consideration the fact that incoming information has to be accumulated in successive neural layers before a reaction can be observed was McClellands Cascade Model (McClelland, 1979). The central idea of this model was that the rate of activation of a unit depends on the difference between its actual input and the already reached level of activation ((McClelland, 1979), eq. (2)):

$$\frac{\mathrm{d}}{\mathrm{d}t} act_i(t) = \alpha_i \left[net_i(t) - act_i(t) \right] \tag{1}$$

*The work presented here was partly founded by the DFG (Deutsche Forschungsgemeinschaft) as part of the Innovationskolleg "Kognitive Komplexität"

†The author thanks Peter beim Graben for a careful ckeck of the mathematical contents and Ulrich Mayr for his guidance regarding the psychological aspects.

($act_i(t)$ denoted the activation of unit i at time t, $net_i(t)$ is the actual input of this unit and α_i is a rate parameter that denotes how fast the unit reacts on the input.)

Later, equation (1) that operates in continuus time was used to create an activation function that operates in discrete time but still kept its leaky integrator characteristics[1] in the PDP account of the Stroop effect (Cohen et al., 1990):

$$act_i(t) = \frac{1}{1 + e^{\overline{net_i}(t-1)}} \tag{2}$$

with

$$\overline{net_i}(t) = \tau_i net_i(t) + (1 - \tau)\overline{net_i}(t-1) \tag{3}$$

Note that here $net_i(t)$ is the actual netinput from incoming connections, but the actual activation is computed from $\overline{net_i}(t)$.

Both equations (1) and (2) could be used to model the time between presentation of a stimulus and the resulting reaction as an accumulation of information in each unit in their networks, but neither approach could provide a learning rule for the rate parameters α and τ, resp., that are critical for the correct behavior of the model.

Another approach that was persued to model the fact that information has to be accumulated in the neural system was to use *recurrent* connections. In contrast to leaky integrator units here the standard activation function (e. g. (Rumelhart et al., 1986b)) does not have to be modified and well known learning rules can be applied to train all the parameters of the net (e. g. *recurrent backpropagation* by Pineda (1987); Almeida (1987), *back-propagation through time* by Rumelhart et al. (1986b), *real-time recurrent learning* by Williams and Zipser (1989) and *time-dependend recurrent back-propagation* by Pearlmutter (1995)). The most prominent example of this approach is the Brain-State-in-a-Box (BSB) Model by Anderson et al. (1977); Anderson (1991). The reaction time is measured as the number of cycles needed for the net to settle in the attractor associated with the specific reaction. recurrent networks have the disadvantage of establishing (generally) more attractors than those intended to represent the reaction (*phantom attractors*, Anderson (1991)). Depending on the input the output is then approaching an undefined state that could be hard to interpret in terms of symbolic reactions. Besides, the attractors of recurrent nets are not always *fixed points* but could also be *fixed cycles* or even *strange* when the output shows chaotic behavior (Hertz, 1995). This can be avoided by using symmetric connections between units. However, this technique requires an increased number of connections in the net and rules out the possibility of self-recurrent connections. Moreover, in recurrent nets the units generally have a subsymbolic representation which moves them closer towards biological units but which also makes them improper for models in which the activation of single units represents a symbolic meaning (e. g. in the Interactive Activation Model by McClelland and Rumelhart (1981)).

Kaplan et al. (1991) argued for the cell assembly (Hebb, 1949) as a construct that bridges this gap between the subsymbolic representation of single neurons and "a symbol-like unit of thought". The cell assembly is proposed as an assembly of neural units that are recurrently connected to exhibit *reverberatory circles* in which information needs to cycle around until the symbolic meaning is fully established. Kaplan et al. (1991) presented a series of experiments in which they made use of physiological principles that should be present in the functioning of cell assemblies: temporally structured input, the dependency on prior experience, competition between assemblies and control of its activation. A main result is that after a cell assembly is provided with input its activation gradually increases until an asymptotic activation is reached or the input is distracted. After distraction of the input the activation grdually decreases until it comes back to its resting level. This behavior is very similar to the time-course of activation in the leaky-integrator units mentioned above.

In this paper a leaky integrator unit is proposed that differs from that in equation (3) because it considers integration and leakage not in terms of netinput but of activation. This preserves the

[1] *Leaky integration* is a process that integrates information within a couple of time steps while at the same time some amount of the already accumulated information is decreased due to leakage.

accumulating characteristics of a leaky integrator unit but prevents it from becoming stuck at an asymptotic value when the netinput is very large. A training algorithm developed for recurrent networks is extended to learn both weigths and rate parameters. In the next section the learning algorithm is derived and some properties of the leaky-integrator units will be presented. In the third section an example is presented on how this approach can be used to model human reaction time data. The final section will discuss the results and give an outlook on work still to be done.

2 Back-Propagation for Leaky Integrators

2.1 Derivation of the formulas

Most often, neural networks operate in discrete time where the activation of a unit in cycle t depends on the activation of connected units at cycle $t-1$. Here, we will follow a more general approach (motivated by Pearlmutter (1995)) where the activation is described by differential equations and networks operate in continuus time. Consider a unit whose activation is described by the following equation:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} act_i(t) &= -act_i(t) + (1-\alpha_i)\, act_i(t) + \alpha_i f_{act}\left[net_i(t)\right] \\ &= -\alpha_i act_i(t) + \alpha_i f_{act}\left[net_i(t)\right] \end{aligned} \tag{4}$$

The symbols have the following meanings:

$\frac{d}{dt} act_i(t)$	Change of activation of unit i at time t
$act_i(t)$	Activation of unit i at time t
$net_i(t)$	Netinput of unit i at time t
α_i	Leakage rate of unit i

The leakage rate α_i tells how much a unit depends on the actual netinput. Its value is between 0 and 1. The higher the value of α, the stronger the influence of the previous level of activation and the less the influence of the actual netinput. $\alpha = 1$ means that the previous activation doesn't have any influence and the new activation is only determined by the netinput (this is the case e. g. for the standardunits used by the PDP group (Rumelhart et al., 1986a)). $\alpha = 0$ means that the actual netinput doesn't have any influence and the activation remains constant. ($1-\alpha$ could also be regarded as the strengh of its *memory* with respect to earlier activations.)

The netinput of unit i is given as the sum of all incoming signals:

$$net_i(t) = \sum_j w_{ij}\, act_j(t) + bias_i + I_i^{ext}(t) \tag{5}$$

With

$net_i(t)$	Netinput of unit i at time t
w_{ij}	Weight of connection from unit j to unit i
$bias_i$	Bias of unit i
I_i^{ext}	external Input to unit i

Fig. (1) shows how the dynamic of the leaky integrator unit described by equations (eq:actdg) and (eq:net) can be simulated with standardunits.

In order to be able to compute the activations numerically one can use Euler's method to change from differential equations to difference equations:

$$\begin{aligned} act_i(t+\Delta t) &\approx act_i(t) + \frac{\mathrm{d}}{\mathrm{d}t} act_i(t)\Delta t \\ \Rightarrow \qquad \frac{\mathrm{d}}{\mathrm{d}t} act_i(t) &\approx \frac{act_i(t+\Delta t) - act_i(t)}{\Delta t} \end{aligned} \tag{6}$$

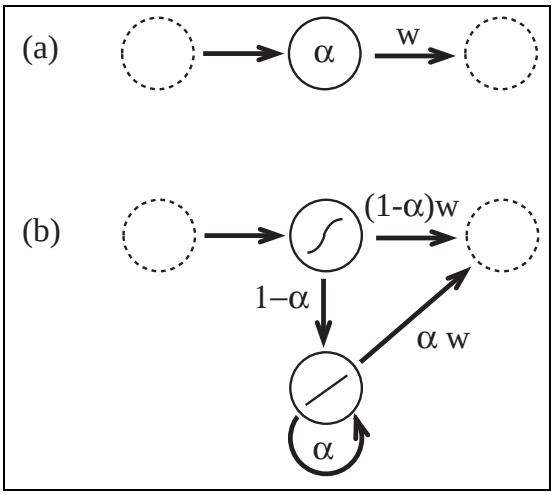


Figure 1: Simulation of a leaky integrator unit (a) and a recurrent combination of two standard-units (b). The function of the leakage term α is mimicked by two parallel standardunits with a logistic and a linear activation function, resp.

Combining (4) and (6) yields

$$\begin{aligned} \widetilde{act}_i(t+\Delta t) &= (1-\Delta t)\widetilde{act}_i(t) + \Delta t \left\{(1-\alpha_i)\widetilde{act}_i(t) + \alpha_i f_{act}\left[\widetilde{net}_i(t)\right]\right\} \\ &= (1-\Delta t\alpha_i)\,\widetilde{act}_i(t) + \Delta t\alpha_i f_{act}\left[\widetilde{net}_i(t)\right] \end{aligned} \tag{7}$$

(Tildes above variables ($\widetilde{act}$) denote continuus functions that have been made discrete.)

Figures 2 and 3 show the time-course of activation for a leaky integrator unit with different values of external input I and leakage parameters α with $I \neq 0$ for $t \in [0, 20]$.

In order to train a network one needs to define an error function

$$E = \int_{t_0}^{t_1} f_{err}\left[\boldsymbol{act}(t), t\right] \mathrm{d}t \tag{8}$$

Here we choose the least mean square function

$$E = \frac{1}{2}\sum_i \int_{t_0}^{t_1} s_i \left[act_i(t) - d_i(t)\right]^2 \mathrm{d}t \tag{9}$$

$d_i(t)$ is the desired activation of unit i at time t and s_i is the relative importance of this activation ($s = 0$: unimportant, $s = 1$: most important).

If one changes the activation of unit i at time t for a small amount, one gets a measure how much this change influences the error function:

$$e_i(t) = \frac{\partial f_{err}\left[\boldsymbol{act}(t), t\right]}{\partial act_i(t)} \tag{10}$$

with

$$f_{err} = \frac{1}{2}\sum_i s_i \left[act_i(t) - d_i(t)\right]^2$$

With (9) as error function we get

$$e_i(t) = s_i \left[act_i(t) - d_i(t)\right] \tag{11}$$

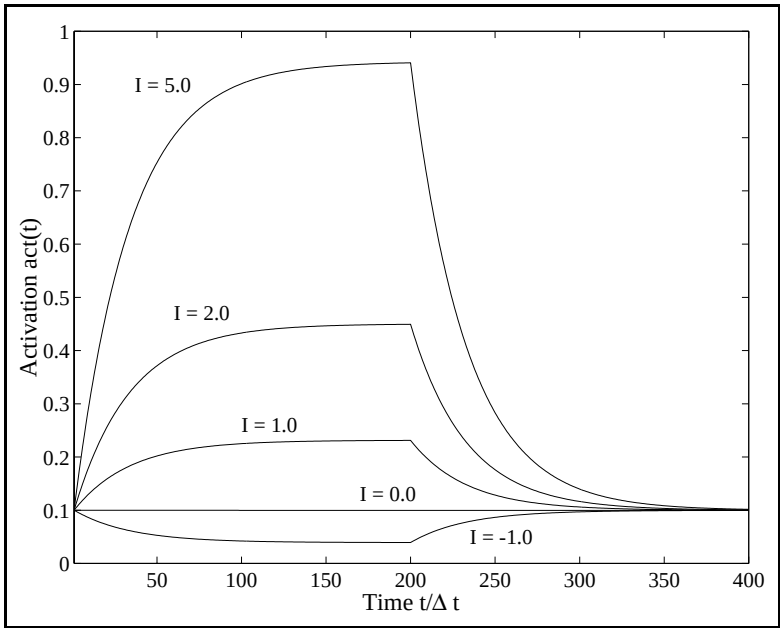


Figure 2: Time-course of activat ion for different input with $\Delta t = 0.1$, $\alpha = 0.3$ and $bias = -2.2$.

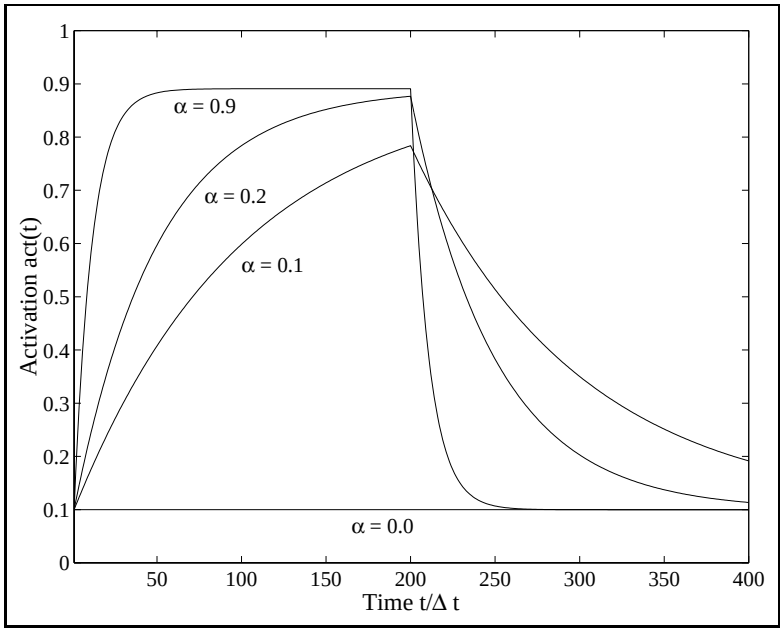


Figure 3: Time-course of act ivation for different leakage rates with $\Delta t = 0.1$, $I^{ext} = 4.3$ und $bias \approx -2.2$.

(10) and (11) describe the influence of a change of activation only for t. In a neural net that is described by (4) und (5) each change of activation at t also influences the activation at later times t' ($t < t'$). The amount of this influence can be described by using derivations ordered by time (Werbos, 1990, 1989):

$$\begin{aligned} \tilde{z}_i(t) &= \frac{\partial^+ E}{\partial \widetilde{act}_i(t)} \\ &:= \frac{\partial E}{\partial \widetilde{act}_i(t)} + \sum_{t'>t} \sum_j \frac{\partial^+ E}{\partial \widetilde{act}_j(t')} \frac{\partial \widetilde{act}_j(t')}{\partial \widetilde{act}_i(t)} \end{aligned} \tag{12}$$

with $j = 1, 2, ..., n$ n number of units
$t' = t + \Delta t, t + 2\Delta t, ..., t_1$ t_1 last defined time

$\tilde{z}_i(t)$ measures how strong a change of activation of unit i at time t influences the error function for all times.

Performing the derivations in (12) with (9), (11), (7) and (5) and setting $t' = t + \Delta t$ gives:

$$\frac{\partial E}{\partial \widetilde{act}_j(t)} = \Delta t e_i \tag{13}$$

$$\frac{\partial \widetilde{act}_i(t + \Delta t)}{\partial \widetilde{act}_i(t)} = (1 - \Delta t \alpha_i) + \Delta t \alpha_i w_{ii} f'_{act} \left[\widetilde{net}_i(t)\right] \tag{14}$$

$$\frac{\partial \widetilde{act}_j(t + \Delta t)}{\partial \widetilde{act}_i(t)} = \Delta t \alpha_j w_{ji} f'_{act} \left[\widetilde{net}_j(t)\right] \tag{15}$$

$\forall$ units j that are connected with unit i

All other derivations are zero.

With this one gets

$$\begin{aligned} \widetilde{z}_i(t) &= \Delta t e_i + (1 - \Delta t \alpha_i)\, \widetilde{z}_i(t + \Delta t) \\ &\quad + \sum_j \Delta t \alpha_j w_{ji} f'_{act} \left[\widetilde{net}_j(t)\right] \widetilde{z}_j(t + \Delta t) \end{aligned} \tag{16}$$

The back-propagated error signal $\mathbf{z}(t)$ is equivalent to the δ in standard back-propagation.

After the last defined activation $d_i(t_1)$ there is no further change of E, so $z_i(t_1 + \Delta t) = 0$.

Making use of Euler's method in opposite direction one finds that the back-propagated error signal can be described by the following differential equation:

$$\frac{\mathrm{d}z_i(t)}{\mathrm{d}t} = \alpha_i z_i(t) - e_i - \sum_j \alpha_j w_{ji} f'_{act} \left[net_j(t)\right] z_j(t) \tag{17}$$

With (16) it is possible to calculate how the error function changes if one changes the parameters α_i and w_{ij}. Each variation also changes the activation act_i. The influence of this activation on E can be calculated using the chain rule of derivations.

If w_{ij} changes for Δt for ∂w_{ij} then the influence of this change on the error function can be described by

$$\begin{aligned} \left.\frac{\partial E}{\partial w_{ij}}\right|_t^{t+\Delta t} &:= \frac{\partial^+ E}{\partial act_i(t + \Delta t)} \frac{\partial act_i(t + \Delta t)}{\partial w_{ij}} \\ &= z_i(t + \Delta t) \alpha_i act_j(t) f'_{act} \left[net_i(t)\right] \Delta t \end{aligned} \tag{18}$$

A change of ∂w_{ij} during the *whole* time $t_0 \leq t \leq t_1$ produces:

$$\frac{\partial E}{\partial w_{ij}} = \alpha_i \int_{t_0}^{t_1} z_i(t) act_j(t) f'_{act}\left[net_i(t)\right] \mathrm{d}t \tag{19}$$

For the influence of a change in α_i on E one finds

$$\begin{aligned} \left.\frac{\partial^+ E}{\partial \alpha_i}\right|_t^{t+\Delta t} &= \frac{\partial E}{\partial act_i(t+\Delta t)} \frac{\partial act_i(t+\Delta t)}{\partial \alpha_i} \\ &= z_i(t+\Delta t)\left\{f_{act}\left[net_i(t)\right] - act_i(t)\right\}\Delta t \end{aligned} \tag{20}$$

For the whole time:

$$\frac{\partial E}{\partial \alpha_i} = \int_{t_0}^{t_1} z_i(t)\left\{f_{act}\left[net_i(t)\right] - act_i(t)\right\}\mathrm{d}t \tag{21}$$

Now we have nearly all equations that are needed for training a neural net of leaky integrator units. Finally we have to pay respect to the fact that the leakage term α must only have values between 0 and 1. This can be done by using

$$\alpha = \frac{1}{1+e^{-\bar{\alpha}}} \tag{22}$$

and learning $\bar{\alpha}$ instead of α. With this replacement one finds

$$\begin{aligned} \frac{\partial E}{\partial \bar{\alpha}_i} &= \frac{1}{1+e^{-\bar{\alpha}_i}}\left(1 - \frac{1}{1+e^{-\bar{\alpha}_i}}\right) \\ &\quad \int_{t_0}^{t_1} z_i(t)\left\{f_{act}\left[net_i(t)\right] - act_i(t)\right\}\mathrm{d}t \end{aligned} \tag{23}$$

Overview of the learning procedure

To start training one needs to have the following information:

1. topology of the net with number of units (n) and connections
2. values of the parameters $\boldsymbol{w_0}$ and $\bar{\boldsymbol{\alpha}}_{\boldsymbol{0}}$ at $t = 0$
3. activations $\boldsymbol{act}(t_0)$ at $t = 0$
4. time-course of the input ($\boldsymbol{I}^{ext}(t), t_0 \leq t \leq t_1$)
5. time-course of the desired output ($\boldsymbol{d}(t)$)
6. activation function f_{act} for each unit
7. error function E
8. time-step size Δt that resembles the required resolution of the time-course ($\Delta t = 0.1$ turned out to be a good default value)

The goal is to find a combination of $\mathbf{w}$ and $\bar{\boldsymbol{\alpha}}$ that gives a minimum for E.

1. At first one has to calculate the netinput (5) for each unit successively for each time-step *forward* in time. Parallely, the activations are calculated with (7).

2. With (9) one calculates the main error E and the error vector $\mathbf{e}(t)$ using (11).

3. Then the error signals are propagated *backwards* through time with (16), making use of the condition $\tilde{z}_i(t_1 + \Delta t) = 0$.

4. Now one calculates the gradient of each free parameter with respect to the error function E with the discrete versions of (19) and (23):

$$\frac{\partial E}{\partial w_{ij}} = \frac{1}{1+e^{-\bar{\alpha}_i}} \sum_{t=t_0}^{t_1} \widetilde{z_i}(t+\Delta t)\widetilde{act_j}(t) f'_{act}\left[\widetilde{net_i}(t)\right]\Delta t \tag{24}$$

$$\frac{\partial E}{\partial \bar{\alpha}_i} = \frac{1}{1+e^{-\bar{\alpha}_i}}\left(1 - \frac{1}{1+e^{-\bar{\alpha}_i}}\right) \sum_{t=t_0}^{t_1} \widetilde{z_i}(t)\left\{f_{act}\left[\widetilde{net_i}(t)\right] - \widetilde{act_i}(t)\right\}\Delta t \tag{25}$$

5. After this, the parameters are changed along the negative gradient (*gradient descent*):

$$w_{ij} = w_{ij} - \eta_w \frac{\partial E}{\partial w_{ij}} \tag{26}$$

$$\bar{\alpha}_i = \bar{\alpha}_i - \eta_{\bar{\alpha}} \frac{\partial E}{\partial \bar{\alpha}_i} \tag{27}$$

   With η_w and $\eta_{\bar{\alpha}}$ as learning rates. ($\eta = 0.1$ is commonly a suitable starting value.)

   The gradient can be used for *steepest descent*, *conjugate gradient* or other numeric approximations (see e. g. Press et al. (1996)).

6. Having obtained the new values $\mathbf{w}$ and $\bar{\alpha}$ the procedure goes back to step 1 and is continously followed until the main error falls below a certain value in step 2 or this criterium is not reached after a maximal number of epochs.

2.2 Memory of a leaky integrator unit

Mozer (1994) has suggested the classification of time-dependend neural networks according to their convolution kernel. To obtain the convolution kernel for a leaky integrator unit we develop the activation of a unit as a series of its activation and netinputs at previous times:

$$\begin{aligned}
\widetilde{act_i}(t) &= (1-\alpha_i\Delta t)\widetilde{act_i}(t-\Delta t) + \alpha_i \Delta t f_{act}\left[\widetilde{net_i}(t-\Delta t)\right] \\
&= (1-\alpha_i\Delta t)^2 \widetilde{act_i}(t-2\Delta t) + (1-\alpha_i\Delta t)\alpha_i\Delta t f_{act}\left[\widetilde{net_i}(t-2\Delta t)\right] \\
&\quad + \alpha_i \Delta t f_{act}\left[\widetilde{net_i}(t-\Delta t)\right] \\
&\vdots \\
&= (1-\alpha_i)^t \widetilde{act_i}(0) + \sum_{\tau=1}^{t-1} \alpha_i (1-\alpha_i)^{(t-1)-\tau} f_{act}\left[\widetilde{net_i}(\tau)\right]
\end{aligned}$$

where the time steps $\Delta t, 2\Delta t, 3\Delta t, \ldots$ were indexed with $\tau = 1, 2, 3, \ldots$.
Regarding continuus instead of discrete values gives

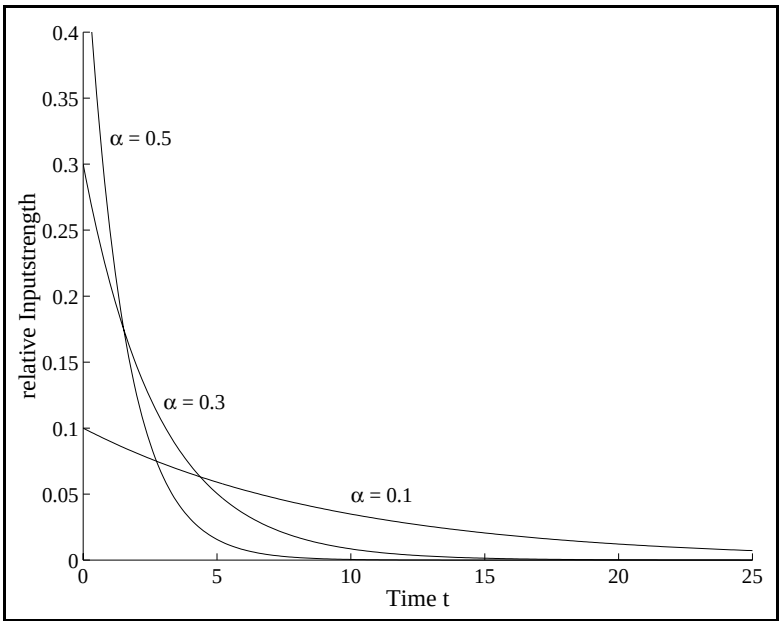
$$
\begin{aligned}
act_i(t) &= (1-\alpha_i)^t act_i(0) + \int_1^t \alpha_i (1-\alpha_i)^{t-\tau} f_{act}\left[net_i(\tau)\right] \mathrm{d}\tau \\
&\equiv b_i(t) + f\left[net_i(t)\right] * k_i(t)
\end{aligned}
$$

with $f\left[net_i(t)\right] = 0 \; \forall t \leq 1$.

The term $b_i(t) = (1-\alpha_i)^t act_i(0)$ describes the decreasing influence of the activation at $t = 0$, $k_i(t)$ is the convolution kernel:

$$
k_i(t) = \begin{cases} \alpha_i (1-\alpha_i)^t & \forall t \in [0, t_1] \\ 0 & \text{otherwise} \end{cases} \tag{28}
$$

This result reveals that a leaky integrator unit exhibits an exponentially decreasing memory (fig. 4).



li-*Kernel* Funktion

Figure 4: *Kernel* functions of a Leaky Integrator unit.

2.3 Comparison to the moving average method

Poddar and Unnikrishnan (1991) have shown that the activation of a leaky integrator unit can be regarded as an exponentially weighted *moving average* window over its input.

When analysing time-series with the moving average method (e. g. Weigend and Gershenfeld (1994)) the mean age of the last n data in a sample are computed as

$$
\begin{aligned}
A_{ma} &= \frac{1}{N} \sum_{n=1}^{N} n \\
&= \frac{N+1}{2}
\end{aligned} \tag{29}
$$

Each data entry is weighted with $1/N$, independend of the time of measurement.

The convolution kernel shows that a data entry in a leaky integrator unit is weighted dependend of the time of measurement. This yields

$$
\begin{aligned}
A_{lima} &= \int_1^\infty \alpha(1-\alpha)^t t \mathrm{d}t \\
&= \alpha(1-\alpha)^t \left[\frac{t}{\ln(1-\alpha)} - \frac{1}{\ln^2(1-\alpha)}\right]\Bigg|_1^\infty \\
&= \alpha(1-\alpha)\left[\frac{1}{\ln^2(1-\alpha)} - \frac{1}{\ln(1-\alpha)}\right] \\
&\qquad\qquad \text{because } \alpha \in [0,1]
\end{aligned} \tag{30}
$$

To compare both methods the *effective* width of a time window can be calculated that would be necessary to abtain the same mean age of the data using the moving average method. Setting $A_{ma} = A_{lima}$ gives

$$
N_{\mathit{eff}} = 2\alpha(1-\alpha)\left[\frac{1}{\ln^2(1-\alpha)} - \frac{1}{\ln(1-\alpha)}\right] - 1 \tag{31}
$$

The larger the leakage term α the smaller the mean age of the used information. The effective width of the time window increases with decreasing α.

3 An application on reaction times

3.1 Time-scale and output criterium

Figure 4 shows that for a leakage term around 0.5 the relative strength approaches zero for inputs that were applied about six cycles before the actual activation is obtained. This means that only those inputs can have an impact on the actual activation were not applied earlier. Reaction time (RT) data typically have an order of 1000 ms. To model significant differences the resolution of the output schould have an order of 10 ms. In order to be able to capture the influence of the *previous* reaction on the actual reaction it is necessary to model a time-window of 1000 ms within six cycles. The difference between two time-steps should be 10 ms. This leads to the following equation for the time-step size:

$$
\begin{aligned}
\Delta t &= \frac{\text{no of cycles}}{\text{order of RT}} \cdot \text{desired resolution} \\
&= \frac{6}{1000} \cdot 10 \\
&= \frac{6}{100}
\end{aligned} \tag{32}
$$

$\Delta t = 6/100$ now resembles a difference of 10 ms between two time-steps.

This approach makes a unit less flexible because the activation cannot change very much between two such time-steps (eq. (7)). Though, this behavior resembles some physiological plausibility if the leaky integrator units are considered to represent cell assemblies rather than single neurons, as cell assemblies are supposed to change their activation much slower (Kaplan et al., 1991).

When modelling reaction time data one has to consider how the reaction should be coded by the output units. At least three approaches have been considered: the response criterium could depend on the absolute activation of the output value (Heathcote, 1998), on the derivation of the output activation (Heathcote, 1998) or on the value of an additional accumulator (Cohen et al., 1990). Here, it is suggested to use two linear output units without leakage term ($\alpha = 0$) that accumulate activation during the whole time-course of a simulation. A response is given after a certain amount of activation has been accumulated. This approach makes it possible to integrate the output units in the training, provided that it is taken care of the strength of the error signals because of the linear activation function.

3.2 Simulation of a task switching experiment

The leaky integrator approach has been used to model the data obtained in the *Task Decay Experiment* by Mayr and Liebscher (2000). In this experiment a group of young subjects and a group of old subjects had to react on stimuli presented on a screen that could differ in form (circle or square) and/or color (yellow or blue). If the instruction was to care for the form, subjects were asked to press the left arrow key in case the stimulus was a circle, otherwise the right arrow key. If the instruction was to care for color the subjects were asked to press left in case of a yellow stimulus, right otherwise. The not important dimension should not be regarded. For a block of 120 trials the instruction remained constant (*single* condition, see Mayr and Liebscher (2000) for more details on the experiment).

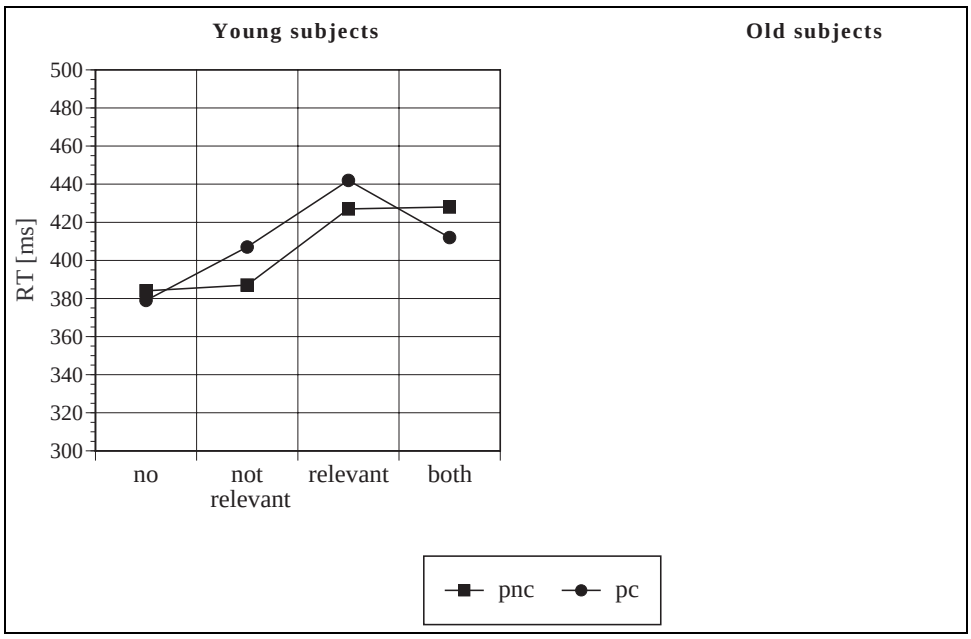


Figure 5: Reaction times for the single condition in the *Task Decay Experiment*. The x-axis denotes the change of stimulus dimension between successive trials. (**pc**: previous trial was congruent; **pnc**: previous trial was incongruent. See text for more details.)

One of the critcal factors was the congruency of the stimulus. Congruency was given when both dimensions required the same reaction (e. g. a yellow circle). A stimulus was considered incongruent if both dimensions required differend reactions (e. g. a yellow square). Another factor was the change of the stimulus dimension between two successive trials. Either there was **no** change (e. g. yellow square → yellow square), a change of the **not relevant** dimension (e. g. yellow square → yellow circle if the instruction is color), a change of the **relevant** dimension (e. g. yellow square → blue square if the instruction is color) or change of **both** dimensions (e. g. yellow square → blue circle). The results of the experiments on both age groups are shown in figure 5.

The net that was used to model these reaction times is shown in figure 6 (See Liebscher (2000) for more details.)

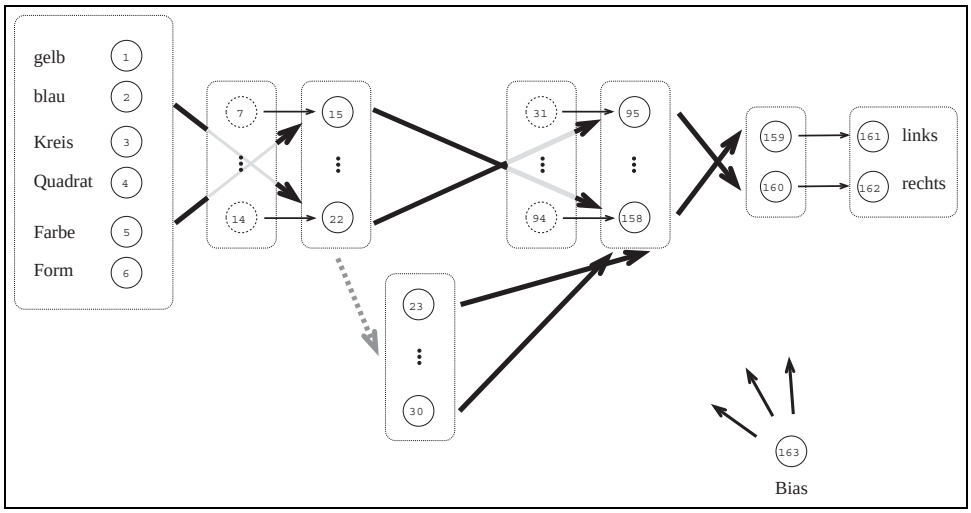


Figure 6: Topology of the neural net for modeling the results of the *Task Decay Experiment.*

The net was trained to show an activation of 35 at the appropiate (accumulating) outputunit after the desired number of time-steps. After an extensive training phase of 1000 epochs with a resolution of $\Delta t = 10$ ms the net was tested on all possible inputs. The results are shown in figure 7.

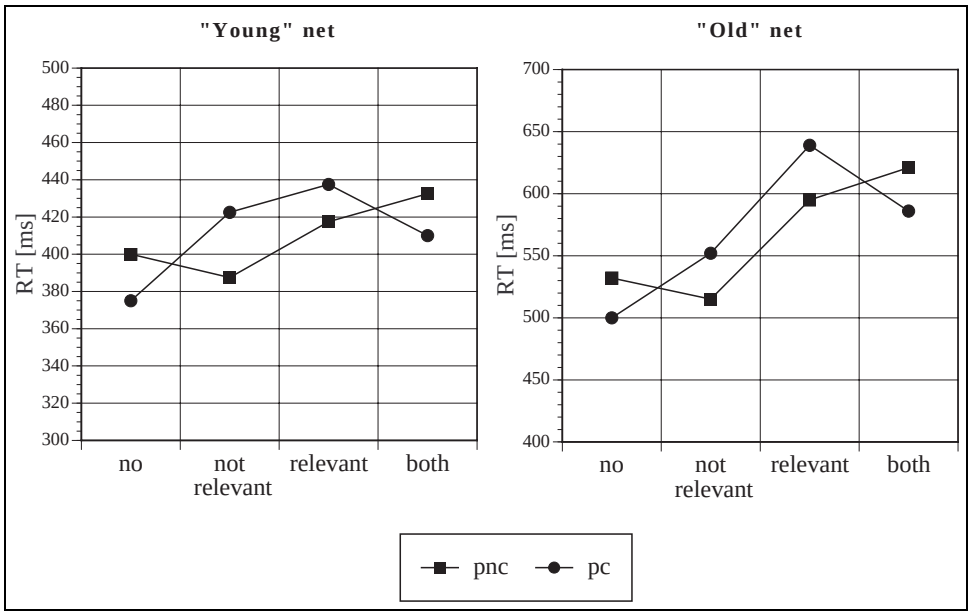


Figure 7: Simulated reacti on times for the single condition in the *Task Decay Experiment.* (See caption of fig. 5.)

It becomes obvious that the net learned to capture the general behavior of the human reaction times. What can also be seen is that the net operates worse the longer the reaction times (the data of the old subjects are less well modelled than the data of the young subjects). This is due to the fact that the memory for previous input decreases exponentially, so the net can make less use of the information that was presented in the previous trial.

4 Conclusion

The results of the simulation show that leaky integrator units can succesfully be applied to the neural networks that simulate time-dependend behavior. Following the argumentation of Kaplan et al. (1991) each leaky integrator unit could be interpreted as a cell assembly, thus allowing to train time-dependend networks with symbolic-like unit representations. They require temporally structured input, depend on prior experience, provide the possibility of competition between assemblies and their activation is controlled by leakage. Although the net used to learn the reaction times from the *Task Decay Experiment* was only feed-forward, the learning procedure is general enough to be also applied to recurrent networks, making it possible to use feedback of activation not only as a way of making a system time-dependend but to fulfill requirements based on physiological theory. Unlike time-delay networks that require to fix a certain maximum time-delay the learning procedure described here lets the leakage parameters adopt to an adequate value, providing more flexibility in the time domain.

In order to find out more about the potential value of using leaky integrator units in modelling time dependend bahavior, especially for time-series-analysis, clearly more simulations are needed to compare the current appoach with other existing methods. A problem here is the rather high time-complexity of the approach. If s denotes the number of time-steps needed for one training pattern and m denotes the number of connections of the architecture the time-complexity is $O(sm)$. With $s \approx 100$ this requires about 100 times more computing time than a standard feed-forward net. As for benchmarking results a large number of simulations is needed, this strongly suggests the use of parallel hardware.

References

Abeles, M. (1991). *Corticonics: Neural Circuits of the Cerebral Cortex*. Cambridge University Press, Cambridge.

Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in an combinatirocal environment. In Caudill, M. and Butler, C., editors, *IEEE First International Conference on Neural Networks*, volume 2, pages 609–618, New York. IEEE Press.

Anderson, J. A. (1991). Why, having so many neurons, do we have so few thoughts? In Hockley, W. E. and Lewandowsky, S., editors, *Relating Theory and Data: Essays on Human Memory in Honor of Bennet B. Murdock*, pages 477–507. Hillsdale, NJ: Erlbaum.

Anderson, J. A., Silerstein, J. W., Ritz, S. A., and Jones, R. S. (1977). Distinctive features, categorial perception, and probability learning: Some applications of a neural model. *Psychological Review*, 84:413–451.

Cohen, J. L., Dunbar, K., and McClelland, J. L. (1990). On the control of automatic processes: A parallel distributed processing account of the stroop effect. *Psychological Review*, 97:332–361.

Heathcote, A. (1998). Neuromorphic models of response time. *Australian Journal of Psychology*, 50:157–164.

Hebb, D. O. (1949). *The Organization of Behavior*. Wiley, New York.

Hertz, J. (1995). Computing with attractors. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 230–234. MIT Press, Cambridge, MA.

Kaplan, S., Sonntag, M., and Chown, E. (1991). Tracing recurrent activity in cognitive elements (TRACE): A model of temporal dynamics in a cell assembly. *Connection Science*, 3:179–206.

Liebscher, T. (2000). *Modellierung von Reaktionszeiten mit neuronalen Netzen aus leaky Leaky Integrator Units*. PhD thesis, University of Potsdam.

Mayr, U. and Liebscher, T. (2000). Is task-set disengagement an active process? evidence from cognitive aging. in preparation.

McClelland, J. L. (1979). On the time-relations of mental processes: An examination of systems of processes in cascade. *Psychological Review*, 86:287–330.

McClelland, J. L. and Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: Part 1. an account of basic findings. *Psychological Review*, 88:375–407.

Mozer, M. C. (1994). Neural net architectures for temporal sequence processing. In Weigend and Gershenfeld (1994).

Pearlmutter, B. A. (1995). Gradient calculations for dynamic recurrant neural networks: A survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228.

Pineda, F. J. (1987). Generalisation of back-propagation to recurrent neural networks. *Physical Review Letters*, 59:2229–2232.

Poddar, P. and Unnikrishnan, K. (1991). Memory neuron networks: A prolegomenon. Technical Report GMR-7493, Computer Science Department, General Motors Research Laboratories, Box 9055, Warren, MI, USA.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1996). *Numerical Recipies in C*. Cambridge University Press, New York.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning internal representations by error propagation. In Rumelhart et al. (1986b), chapter 8.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, editors (1986b). *Foundations*, volume 1 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, MA.

Weigend, A. S. and Gershenfeld, N. A., editors (1994). *Time Series Prediction*. Addison-Wesley, Redwood City.

Werbos, P. (1989). Maximizing long-term gas industry profits in two minutes in Lotus using neural network models. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:315–333.

Werbos, P. (1990). Back-propagation through time: What it does and how to do it. In *Proceedings of the IEEE*, volume 78, pages 1550–1560.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrant neural networks. *Neural Computation*, 1:270–280.

Genetic Algorithms and the Evolution of Neural Networks for Language Processing

Jaime J. Dávila
Hampshire College, School of Cognitive Science
Amherst, MA 01002
jdavila@hampshire.edu

Abstract

One approach used by researchers trying to develop computer systems capable of understanding natural languages is that of training a neural network (NN) for the task. For this type of approach, one of the key questions becomes how to best configure NN parameters such as topology, learning rates, training data, and other. How to choose values for these parameters is still an open question, especially since the effect these variables have on each other is not completely understood.

Genetic algorithms (GA) are particularly well suited for finding optimal combinations of parameters, since they make no assumption about the problem being solved. Different NN configurations are coded as genomes, which have a fitness function based on how well they can solve a particular task. Genomes are paired and recombined in the hope that the offspring of good solutions will be even better.

In this paper I present ways in which I have used Genetic Algorithms (GAs) to find which NN parameter values produce better performance for a particular natural language task. In addition to this, the system has been modified and studied in order to evaluate ways in which coding methods in the GA and the NN can affect performance. In the case of GA coding, an evaluation method based on schema theory is presented. This methodology can help determine optimal balances between different evolutionary operators such as crossover and mutation, based on the effect they have on the structures being processed by the GA.

In the case of NN coding, the effect of different ways of representing words and sentences at the output layer is examined with both binary and floating point schemes.

Keywords: Neural Networks, Genetic Algorithms, Natural Language Processing.

1 Introduction

One approach used by researchers trying to develop computer systems capable of understanding natural languages is that of training a neural network (NN) for the task. For this type of approach, one of the key questions becomes how to best configure NN parameters such as topology, learning rates, training data, and other. How to choose values for these parameters is still an open question, especially since the effect these variables have on each other is not completely understood.

Genetic algorithms (GA) are particularly well suited for finding optimal combinations of parameters, since they make no assumption about the problem being solved, and find solutions by combining exploitation of known good solutions with exploration of new solutions. GAs are modeled after the process of natural selection. We start with a randomly selected population of elements, each of which represents a possible solution to the problem being solved. Each of these elements, called a genome, is formed by a string of values (called genes) that code the free variables in the experiment. The fitness of each genome is determined by how well the solution it codes solves the problem at hand. From the point of view of a genetic algorithm trying to optimize NN configurations, particular values for these variables constitute particular schemata.

The rest of this paper is organized as follows; section 2 presents some of the ways in which other researchers have used GA to optimize NN. Section 3 presents the natural language task I

have been working with. Section 4 will present a method of coding NN topologies at the layer level of abstraction. In section 5 I present a mathematical model for evaluating a GA coding scheme's ability to efficiently process the genetic schemata in its population. Section 6 uses this model to make predictions for three different ways of coding NNs for the natural language task I present in section 3. Section 7 analizes results for these three coding schemes. Section 8 deals with different ways of representing sentences at the NN output layer. Section 9 then discusses the empirical results of using this different sentence coding scheme. Finally, section 10 describes possible future research and section 11 presents conclusions.

2 GA for NN Optimization

In recent years researchers have used genetic algorithm techniques to evolve neural network topologies. Although these researchers have had the same aim in mind (namely, the evolution of topologies that are better able to solve a particular problem), the approaches they used varied greatly.

For example, de Garis (1996) evolved NN by having a series of growth commands give instructions on how to grow connections among nodes. Each node in the network processed signals that told it how to extend its synapses. When two different synapses reached each other, a new node was formed. The genetic algorithm was responsible for evolving the sequence of growth commands that controlled how the network developed.

Fullmer and Miikkulainen (1991) developed a GA coding system where pieces of a genotype went unused, imitating biological DNA processing. Only information stored between a Start marker and an End marker was used to generate networks. The number of correctly configured Start-End markers defined how many hidden nodes the network would have. In addition, information between these Start-End markers defined how the nodes were connected to each other. The meaning conveyed by each position in the used part of the genome depended on its distance from its corresponding Start symbol.

For example, the genome shown in figure 1 would generate two nodes, one for string **S,a,1,b,5,a,-2,E** and another for string **S,b,0,a,3,E** , which wraps around the end of the genome. Node a had an initial activation of 1(because of substring **S,a,1**), is connected to node b with a weight of 5 (because of substring **b, 5**), and to itself with a weight of -2 (because of substring **a, -2**). Node b had an initial activation of 0 (because of substring **S,b,0**) and a connection to node a with a weight of 3(because of substring **a,3**).

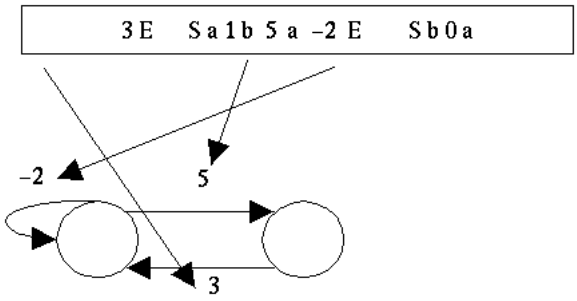


Figure 1: example of coding used by Fullmer & Miikulainen (Fullmer and Miikkulainen (1991)).

The network evolved by this process was used to control a virtual creature's movements in a square field, avoiding bad objects and coming into contact with good objects. The GA continued to run until a network that could solve the problem evolved. The number of generations needed until this network was found varied between 7 and 304 for objects that could be identified before hitting them, and between 15 and 414 generations when recognizing the object required traveling around it looking for a characteristic view.

Kitano (1994) used GA to evolve a sequence of graph generation rules, as opposed to directly coding network topology. Each genome defined a sequence of rules used to rewrite an element of the graph. When these rules were applied until only terminal symbols remained, the graph defined a connectivity matrix which was then used to configure a NN. For example, if we were developing a network with two nodes, a genome might code rules [S → AB][A → 01][B → 10]. When these three rules are applied we end up with a 2*2 matrix than defines the connectivity between the two nodes in the network.

To be sure, the above examples do not represent a complete list of researchers who have used GA to optimize NN. A more complete review can be found, for example, in Yao (1993).

3 A Natural Language Processing Task for a NN

This task was originally presented in Dávila (1999). As an overview, a network is asked to receive a sentence one word at a time, and to incrementally build a description of the sentence in its output nodes. For example, if the sentence "the boy ran in the park" is entered, the network should respond by indicating that the boy is a noun phrase, and it acts as the agent of the verb **ran** . The network should also indicate that "in the park" is a prepositional phrase modifying the verb **ran** .

Entering a word into the network amounts to activating a single node that represents the given word at the input layer, and at the same time activating those semantic nodes that reflect the meaning of the word being entered. For example, to enter the word **boy** , a node that represents that word is activated, as well as nodes that indicate that the word being entered is a proper noun, singular, concrete, and human. In addition, an ID node is set to a value that would allow the network to distinguish john from other words that might have the same semantic identity, such as **girl** . An example of such activation is shown in figure 2.

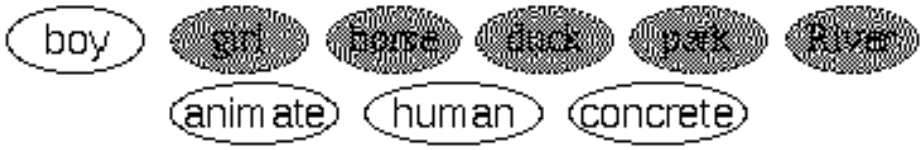


Figure 2: Input layer presentation of **boy** (not all input nodes are shown).

The language used in this research is composed of ten nouns: boy, girl, john, mary, horse, duck, car, boat, park, river. Available semantic nodes are: human, animal, or mechanical (three mutually exclusive nodes); animate or inanimate (represented by one node, active if the noun is animate); proper (active if true, inactive otherwise); and one ID node.

Verbs are entered in a similar way; the node representing that verb is activated, simultaneously with semantic nodes that convey the meaning of that verb. Semantic nodes available for verbs are: present or past tense (two mutually exclusive nodes), auxiliary verb, movement verb, sound producing verb, sound receiving verb, visual receiving verb, and a verb ID node used to distinguish verbs that would be identical otherwise (for example, ran and swam would be identical without this last node). In total, there are twelve main verbs (saw, swam, swimming, ran, runs, running, is, was, raced, floated, said, heard) and two auxiliary verbs (is, was). For example, figure 3 shows how the verb runs is entered in the network; the individual node for **runs** is activated, as well as semantic nodes representing a verb of movement and a present tense verb.

In addition to nouns and verbs, the language has three adjectives (fast, blue, red), one article (the), one adverb (fast), and three prepositions(with , in , after). Each of these is entered in the network by activating an individual node for the word, plus an additional node that indicates which type of word (adjective, article, adverb, or preposition) is being entered.

After each word is entered, the NN is expected to produce a representation of what it understands about the sentence up to that point. For example, after the network sees **the boy** (entered

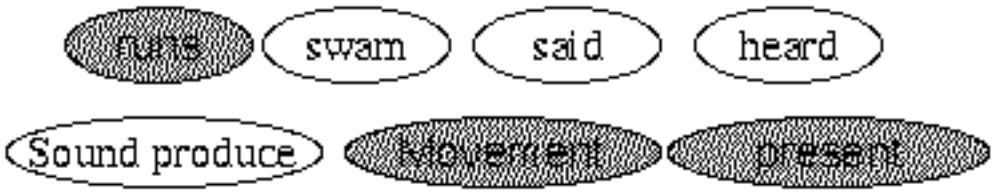


Figure 3: Input layer presentation of **runs** (not all input nodes are shown).

one word at a time; first **the** and then **boy**) it should indicate that it has detected a noun phrase that uses an article, and that the noun of this phrase is **boy. Boy** in this case is represented by activating output nodes that code *human , animate , concrete* , and an additional ID node that distinguishes **boy** from other words that otherwise would have identical semantics (such as **girl**). An example of such an activation is shown in figure 4.

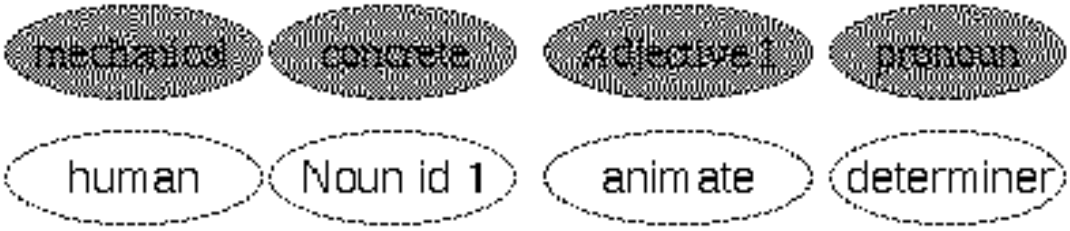


Figure 4: output layer presentation of **the boy** (not all output nodes are shown).

If the network, immediately after having been shown the boy at the input layer, is shown **runs** , it should respond by indicating that it has detected a verb phrase with **runs** as its verb. Indicating that the verb of this verb phrase is **runs** is done by activating semantic nodes for *verb of movement* and *present tense* . In addition, a node ID is activated so that **runs** can be differentiated from other verbs that would otherwise have identical semantics. At this point the network should also indicate that the first noun phrase detected is the agent of the first verb phrase detected. This is done by activating a *np1-agent-of-vp1* node in the output layer.

In the manner described above, then, the network should continue to indicate its understanding of the sentence being entered until an **end-of-sentence** marker is seen. The fitness of a network is determined by computing the sum of squared errors for all output nodes during the processing of all sentences in the language.

4 Genetic Definition of NN Topology

Each NN in this system has 75 hidden nodes between the input and output layers. These 75 nodes are divided into N hidden layers, where N is a number between 1 and 30. The exact number of hidden layers is determined by the first gene of the corresponding genome. This position stores a random floating point number, with a value between 0 and 1. To determine how many hidden layers a network has, the value of this gene is multiplied by 30, and rounded to the next highest integer. If the result of this rounding up is 31, the network uses 30 hidden layers.

The number of hidden nodes in each of these hidden layers is also determined by the network's corresponding genome. The genome has 30 genes used to code the relative worth of each of the possible hidden layers. Once the number of hidden layers is determined to be N using the process described above, the N layers with the highest relative worth are identified. The 75 available

hidden nodes are distributed among each of these N hidden layers according to each layer's worth relative to the sum of all N worth values.

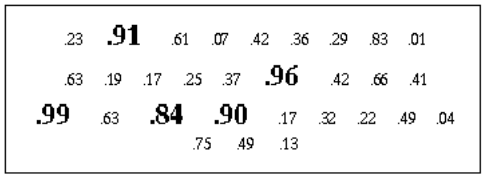


Figure 5: sample of genes 1-30.

For example, if a genome had genes 1-30 as illustrated in figure 5, and it had already been determined that it would have five hidden layers (as described above), the five layers to use are those indicated in bold. Since the sum of these five genes is 4.6, the first hidden layer would have (75*.91/4.6 =) 14 nodes. The other four hidden layers would be allocated hidden nodes in the same way.

The connections between layers are also determined by the network's genome. For each of the thirty possible layers, there is a gene that indicates where the layer takes its input from. Each of these genes stores a random floating point value between 0 and 1. To determine where each hidden layer takes its input from, its takes-its-input from gene value is multiplied by N+2 (where N is the number of hidden layers this network will have, as determined by the procedure outlined previously), and rounded to the nearest integer. The resulting number points to which layer this one takes its input from. We multiply by N+2 to allow a hidden layer to take its input from any of the N hidden layers, as well as either the input or the output layer. A value of 1 would mean the layer takes its input from the input layer. A value of N+2 would mean the layer takes its input form the output layer. For values between 2 and N+1, the layer would take its input from the layer with the (N-1)th highest relative worth.

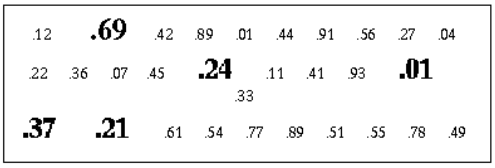


Figure 6: sample of genes 31-60.

For example, if the same genome used for the example above had genes 31-60 as illustrated in figure 6, we would look at the corresponding 5 takes-input-from genes, shown in bold in figure 6. Multiplying each of the selected genes by 6, we would obtain 4.14, 1.44, .06, 2.22, and 1.26. This would mean that hidden layer 1 would take its input from hidden layer 4, hidden layer 2 would take its input from hidden layer 1, hidden layer 3 would take its input from the input layer, hidden layer 4 would take its input from hidden layer 2, and hidden layer 2 would take its input from hidden layer 1.

Where each layer sends its output is determined in a similar way, using positions 61-90 of the genotype. Each of these genes stores a random floating point value between 0 and 1. To determine where each layer sends its output, its sends-output-to gene value is multiplied by N+1 and rounded to the nearest integer. The resulting number points to which other layer this one will send its output to. We multiply by N+1 to allow for hidden layers sending their output to any of the N hidden layers, as well as to the output layer. A value of N+1 would mean the layer sends its output to the output layer. For values between 1 and N, the layer sends its output to the layer with the Nth highest relative worth. No layer sends its output back to the input layer.

Results for the experiment described above are presented in table 1. The four topologies evolved by the GA system described above outperform commonly used NN topologies such as

Topology	Average	Best	Worst
Type I	90.21	92.09	88.97
Type II	95.99	97.62	84.91
Type III	87.59	88.84	86.20
Type IV	82.69	85.97	80.39
SRN	70.58	90.40	17.69
fully connected	72.95	90.41	17.79
FGS	71.85	90.40	33.29
N-P	72.95	90.41	17.79

Table 1: Percent of language correctly processed after training with 20% of complete language, for both evolved and commonly used topologies.

Simple Recurrent Networks (SRN), Frasconi-Gori-Soda networks, and Narendra-Parthasarathy networks. Although some of these commonly used topologies managed to outperform some evolved topologies in the best of cases, on average the evolved topologies performed better by more than 10%. In addition, previously used topologies demonstrate a higher sensitivity to initial conditions. The worst performance for previously used topologies is more than 45% lower than the worst performance for evolved topologies. Details about the characteristics of the evolved networks can be found in Dávila (1999).

5 Schema Disruption Computation

If we view the evolution of NN as a process with the main goal of defining connections between any two nodes, then we can determine their ability to combine building blocks by estimating how likely it is for evolutionary operations to disrupt connection definitions; the less likely it is for connection definitions to be disturbed, the easier it is for the algorithm to combine building blocks present in the current population.

An operation like crossover can disrupt a connection definition every time a crossover point is selected between two genes that, taken together, define a connection between nodes of a network. Therefore, how likely it is for crossover to cause this disruption can be estimated by the distance between genes that combine to define any particular connection. If a particular connection is defined by alleles in genes g_i and g_j , then the bigger the distance between g_i and g_j , the bigger the chance that the connection will be disrupted by a crossover operation. Taking a sum of the distance between genes that can define a connection we obtain a total disruption index (TDI) of

$$\sum_{k=0}^{C}\sum_{i=0}^{N}\sum_{j=0}^{N}(|i-j| * DC(k,i) * DC(k,j)) \tag{1}$$

where N is the number of genes, and DC(k, x) equals to a number between 0 and 1 which indicates what is the probability that gene x is involved in defining connection k. Notice that this number reflects a global probability of disruption for the complete network, as opposed to for any particular connection. This is different from what was originally presented in Dávila (2000), and is motivated by the fact that the more connections a network has, the more likely it is to suffer disruptions.

6 Making Predictions with the TDI Computation

Under the GA coding method presented in section 4, which from now I will call SYSTEM-A, the existence of a connection k between nodes i & j depends on the number of layers that the hidden nodes are divided into (gene 0), which layers contain nodes i & j (genes 1-30), where the layer with

node j takes its input from (a gene from among genes 31-60, depending on the values of j, gene 0, and the nodes/layer distribution determined by genes 1-30), and where the layer with node i sends its output to (a gene from among genes 61-90, again depending on the values of i, gene 0, and the nodes/layer distribution determined by genes 1-30).

As a comparison, the same type of network topology could be evolved by using the following coding scheme, which I will call SYSTEM-B. Each network still has 75 hidden nodes, but they are always divided into 30 hidden layers (that is, there is no gene used to determine how many hidden layers to use). The distribution of these 75 nodes into 30 layers is done based on 30 relative worth values in the genome. To determine how many nodes each layer has, the relative worth for a specific layer is divided by the sum of relative worth of all 30 layers, and then multiplied by 75.

The connections between layers are also determined by the network's genome. For each of the thirty layers, there is a gene that indicates where the layer takes its input from. Each of these genes stores a random floating point value between 0 and 1. To determine where each hidden layer takes its input from, its takes-its-input from gene value is multiplied by 32 and rounded to the nearest integer. The resulting number points to which layer this one takes its input from. A value of 1 would mean the layer takes its input from the input layer. A value of 32 would mean the layer takes its input form the output layer. For values between 2 and N+1, the layer would take its input from the (N-1)th hidden layer.

If we assume that DC(k,x) = 1 for all values of k and x, then TDI(SYSTEM-A) = TDI(SYSTEM-B). In reality, though, DC(k,x) does not always return 1, and in fact it tends to return smaller values under SYSTEM-B than under SYSTEM-A. Notice, for example, that under SYSTEM-A the set of gene sequences that would allow node n to be in layer L has a higher cardinality than under SYSTEM-B, given that SYSTEM-B always has 30 hidden layers, while under SYSTEM-A the number of hidden layers is determined by a gene. This will, in turn, affect which genes are involved in defining a connection between two given nodes, which affects DC(k,x). Under standard random distributions, considering actual values for DC(k,x) would give TDI(SYSTEM-A) > TDI(SYSTEM-B). What this would mean is that it is easier for a particularly good schema to be disrupted by crossover under SYSTEM-A than under SYSTEM-B.

Aside from the effect of DC(k,x), of course, the actual positioning of the genes and how they map into phenome characteristics also has an effect on the disruption caused by crossover operations. Take, for example, a system with the same types of genes as SYSTEM-B, but where the position of the genes has been altered. Instead of having 30 worth values followed by 30 takes-input-from values and then 30 sends-output-to values, SYSTEM-C arranges genes so that the worth, takes-input-from and sends-output-to genes for any one particular layer are in three consecutive positions (positions 3*L, 3*L+1, and 3*L+2, where L is the hidden layer number). Because a gene with a particular functionality will have the same effect on resulting phenomes regardless of its position in the genome, we can discard terms DC(k,i) and DC(k,j) in equation (1), and obtain

$$TDI'(.) = \sum_{k=0}^{C} \sum_{i=0}^{N} \sum_{j=0}^{N} (|i-j|) \tag{2}$$

Under this definition, TDI (SYSTEM-B) > TDI (SYSTEM-C), since SYSTEM-C minimizes the distance between the relative-worth and takes-input-from genes . This means that crossover is less likely to disrupt a useful schema under SYSTEM-C.

7 Empirical Results

To verify the effect these disruptions might have on solutions found by evolutionary computation, I performed the optimization of topologies for the natural language problem outlined previously under the three genome coding schemes discussed above. In order to better measure the effect of crossover, computations were performed with no mutation and population size of 21 elements. Network fitness values were computed by taking the sum of square errors for all output nodes throughout the presentation of a language of 413 sentences. Training is performed on 20% of this same language. In order to verify consistent performance, all evolved networks were validated by

performing 144 bootstrap validation runs per network (Weiss and Kulikowski (1991)). Evolutionary runs were repeated 48 times, and the graphs presented here, although taken from particular runs, are typical of results throughout all runs.

Figures 7 and 8 show the fitness for the best, average, and worst individuals in a run of 40 generations. Figure 7 plots the values for SYSTEM-A, while figure 8 shows the values for SYSTEM-C. It quickly becomes evident that average and even worst element evolve towards the best solution faster under SYSTEM-C. This is in accordance with the prediction made by TDI computations for these two coding schemes; the traits that make the best individual so good can be transfered to other members of the population more efficiently under SYSTEM-C.

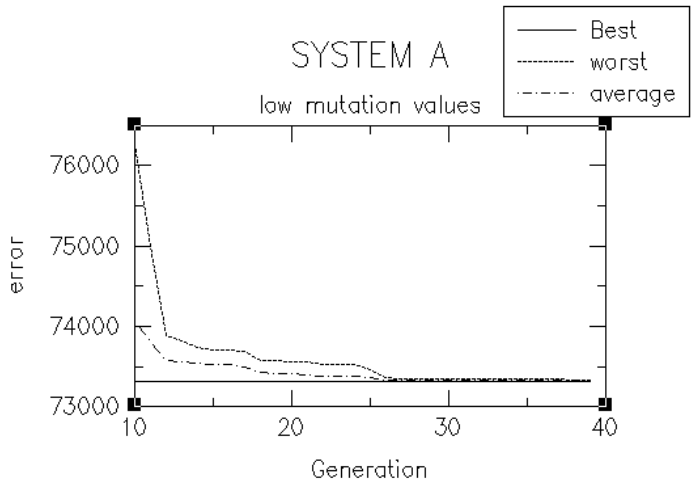


Figure 7: fitness values for SYSTEM-A

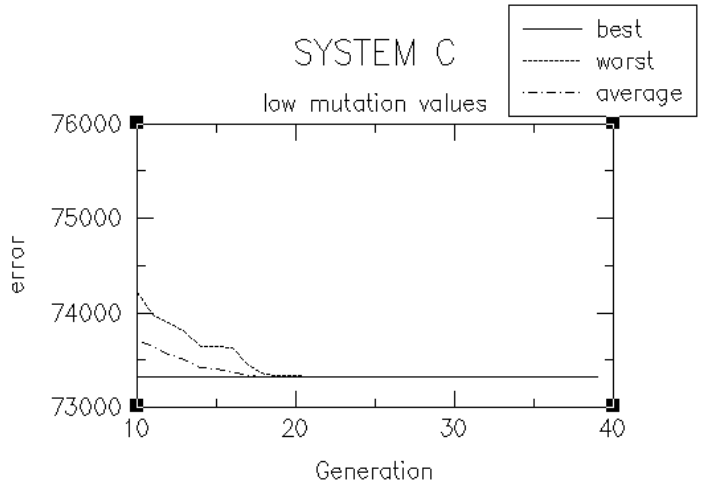


Figure 8: fitness values for SYSTEM-C

In order to verify that it is in fact the ability to efficiently process GA schemata that causes this performance difference, runs with SYSTEM-C were repeated with higher mutation values. A sample run of this type of experiment is plotted in figure 9. Notice how it now takes longer for the GA to process and disperse good schemata through the population.

It is important to notice that what is being computed, predicted, and verified here is simply the ability of a GA system to process the schemata that already exist in a population. These

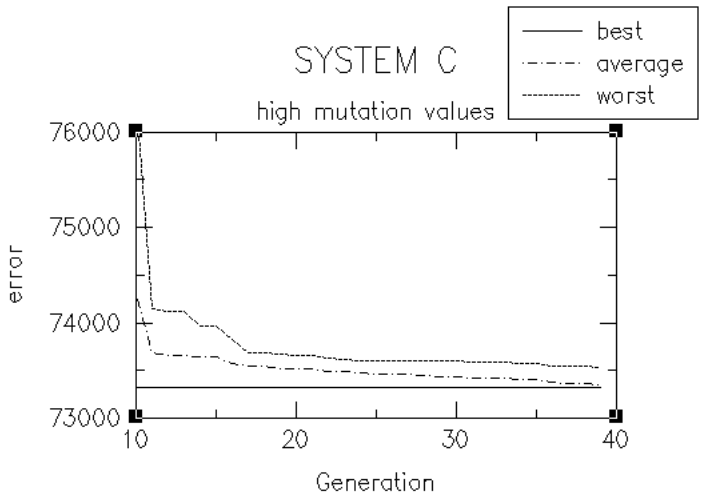


Figure 9: fitness values for SYSTEM-C under higher mutation values

computations do not talk directly about the ability of the system to discover new schemata. In fact, random mutation in some cases could generate an individual which is better than anything currently in the population, or anything that could be evolved from the population by schema recombination. An example of this is shown in figure 10. Notice, however, that although the best elements in this population are better than the ones obtained in previous experiments, the system is still slower to process the schemata. This is shown by the bigger difference between better and worse elements within this particular run.

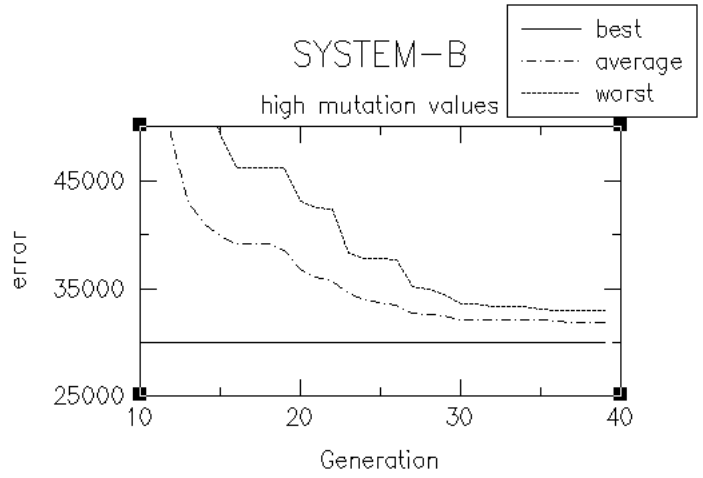


Figure 10: fitness values for SYSTEM-B under higher mutation values

8 Analog vs. Digital Sentence Representation

The method for sentence representation at the output layer used in these experiments has a number of limitations and problems. For example, a high number of output nodes are required to remain inactive (output of zero) for the output pattern to be considered correct. Because of this, both the evolutionary process and the (backpropagation) learning process find solutions where many nodes are simply disconnected from the rest of the network, ensuring that they never activate. Although this leads to errors for some of the patterns, these errors are small when considered within the

context of the complete language used.

Another issue introduced by this output coding scheme is the difficulty of expanding it to include more that the three noun phrases and two verb phrases used here. The number of nodes, and therefore connections, generated by such an expansion would make the system harder to train and slower to respond to inputs.

In order to avoid this type of problem, I have modified the way in which sentences are represented at the output layer. Instead of having binary (and mutually exclusive) nodes representing noun properties such as human, animal, mechanical and inanimate, these four nodes are substituted with a single type-of-noun node, with activations ranging between 0 and 1. Something similar can be done to describe verbs, substituting nodes for verb-of-movement, verb-of-sound-producing, verb-of-sound-receiving, etc., with a single type-of-verb node. Another change similar to this can be done with the NP1-MODIFIES-NP2, NP1-MODIFIES-NP3, NP1-MODIFIES-VP1, NP1-MODIFIES-VP2 nodes; they can be substituted with a single node that indicates which phrase this one phrase is modifying. By doing this we both drastically reduce the number of nodes that can correctly keep their output at zero, and the number of nodes that need to be added to the output layer when a new phrase is added to the sentence representation. In particular, the number of nodes needed to represent a noun prase is reduced from 3*NVP + NNP + 12(where NVP is the number of verb phrases and NNP is the number of noun phrases) to a constant 8. The number of nodes needed to represent a verb phrase is reduced from NNP + 6 to a constant 6.

9 empirical Results for Analog Representation.

After 40 generations, typical runs of the experiments described in section 8 converge to NN with errors of around 5.45 - 5.81%. This compares very favorably with results for digital systems, where the variability was somewhat higher. In fact, the only digital system capable of ever producing better results (Type II in table 1) has the biggest performance variability, and sometimes performs close to 10 percentage points below the lowest analog performer.

A sample plot of fitness values for analog systems is presented in figure 11.

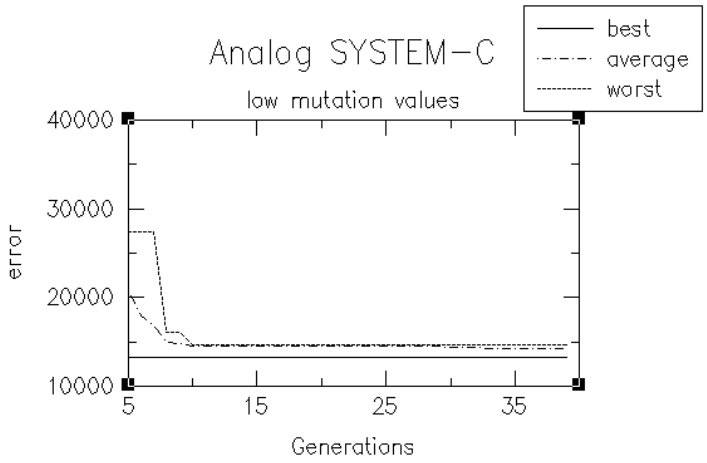


Figure 11: fitness values for analog SYSTEM-C, low mutation values

In terms of GA schemata disruption, figure 11 shows this coding taking ample advantage of the better elements in the population. Compare figure 11, for example, with figure 8; we note that the system moves towards good fitness values in the presence of a superior element much quicker than the digital version of SYSTEM-C. Equations (1) & (2) predict this, based on a smaller total number of connections C used (since there are fewer output nodes).

10 Future Work

In order to better understand the effect of different TDI values during the evolution of NN, analysis of a series of different coding schemes should be caried out. For example, the systems mentioned in section 1 could be examined, as well as others found in the GA/NN literature. By observing how different coding schemes appear to be able (or not) to process good GA schematta, a better understanding of TDI values can be obtained. In addition, the disruptive ability of different TDI values should be compared with that of several ranges of mutation rates. By doing so, TDI values can be better understood, allowing this methodology to better assist in the design of coding schemes and the balancing between different types of evolutionary operations.

Another aspect of TDI computation to be explored relates to its effect when a population contains elements with drastically different fitness values. For example, figure 10 shows fitness values for a run where there is a considerable initial difference between the best and worst elements. Although the graph shows that fitness values are starting to converge towards the 40th generation, the system seems to process the schemata of the best elements slowly. It would be important to determine if this is caused by the magnitude of the difference, or by other factors (evolutionary pressure, size of the population, etc.).

Finally, several question relating to the analog coding system presented in section 8 should also be explored further. In particular, the effect of using the same number of nodes, but with different coding schemes for nouns and verbs should be investigated. Furthermore, the possibility of having a GA evolve these codings should be explored.

11 Conclusion

This paper has presented a mathematical methodology for predicting the effectiveness of a GA in processing NN topology. The methodology is used to predict the effectiveness of several GA coding schemes. These predictions are shown to correspond with actual results.

In addition, different ways of representing sentences at a NN output later are presented. Advantages of using non-binary representations are discussed, both from the point of view of expanding sentences that can it can process and efficiency of GA used to evolve them.

References

de Garis, H. (1996). CAM-BRAIN: The Evolutionary Engineering of a Billion Neuron Artificial Brain by 2001. Which Grows/Evolves at Electronic Speeds Inside a Cellular Automata Machine (CAM), In *Lecture Notes in Computer Science - Towards Evolvable Hardware, Vol. 1062*, pp. 76-98. Springer Verlag.

Dávila, J. (1999) Exploring the Relationship Between Neural Network Topology and Optimal Training Set by Means of Genetic Algorithms. In *International Conference on Artificial Neural Networks and Genetic Algorithms*, pp. 307-311. Springer-Verlag.

Dávila, J. (2000) A New Metric for Evaluating Genetic Optimization of Neural Networks. In *Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks.* pp. 52-58. IEEE

Fullmer, B., Miikkulainen, R., (1991). Using marker- based genetic encoding of neural networks to evolve finite-state behavior. In *Proceedings of the first European Conference on Artificial Life*, pp.. 253-262.

Kitano, H., (1994). Designing Neural Networks using Genetic Algorithm with Graph Generation System. In *Complex Systems, 4.* pp. 461-476.

Weiss, S., Kulikowski, C. (1991). Computer Systems that Learn. Classification and Prediction Methods from Statistics. In *Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann.

Yao, X. (1993). A review of evolutionary artificial neural networks. In *International Journal of Intelligent Systems, 4.* pp. 203–222.

A Study and Improvement of the Genetic Algorithm in the CAM–Brain Machine

Yvan Saeys and Herwig Van Marck
Sail Port Western Europe
Centre for Evolutionary Language Engineering (CELE)
Sint-Krispijnstraat 7, B-8900 Ieper, BELGIUM
{Yvan.Saeys|Herwig.VanMarck}@sail.com

Abstract

This paper presents a study of the CAM–Brain Machine (CBM), a hardware tool which implements a cellular automata based neural network. The idea of this machine is to build a brain, consisting of up to 64,640 modules. Each of these modules implements a neural network with up to 1152 neurons.
The structure of these modules is not fixed, but evolves directly in hardware under the control of a built-in genetic algorithm that guides the evolution. The goal was to analyse this existing genetic algorithm in the CBM, discover some of its weaknesses and present a better alternative.

Keywords: Artificial Intelligence, Genetic Algorithms, Brain Building, Evolutionary Programming.

1 Introduction

In the domain of artificial intelligence, there has already been a lot of research on the way information is processed in the human brain. The challenge to mimic the human brain is the dream of numerous scientists and with the emergence of artificial neural networks a mathematical model was proposed, resulting in simple nets with limited capabilities.

As research continued the amount of neurons in these nets increased and their capabilities improved spectacularly. With increasing processor speed, simulation times of neural networks could be reduced and larger networks (networks with more neurons) could be tested. Still there's a big difference between simulations and really useful networks. Nowadays we realize that, in order to solve more realistic problems, increasing the amount of neurons is one of the things that has to be changed. One of the projects that tries to accomplish this is the CAM-Brain project, which initially aimed to implement one billion artificial neurons. To simulate such a large network, a hardware tool was designed in order to get a realistic, high-speed approach, the result being the CAM-Brain Machine (CBM). As a consequence of this implementation in hardware, some tradeoffs had to be made, restricting the neural network model and making it different from the traditional artificial neural networks.

The central concept in the CBM is a *module*. A module is based on a three-dimensional cellular automaton and represents a digital neural network. The cellular automaton can be thought of logically as a toroidal cube of 24×24×24 cells. Each of these cells is connected to his six neighbours and can be either a neuron, an axon or a dendrite. The number of neurons in one module is limited: a maximum of 1,152 neurons per module are supported, with fixed positions. Since all neuron positions are fixed on positions of the form $(3x-1, 2y-1, 2z)$ with $0 < x \leq 8, 0 < y \leq 12, 0 \leq z < 12$, the neuron is fixed at coördinate (2,1,0) in each block of 3×2×2 cells.
Figure 1 shows the alignment of the neurons in the module. Each grey cell represents a neuron. Since up to 64,640 modules can be combined to form one brain the maximal number of neurons in

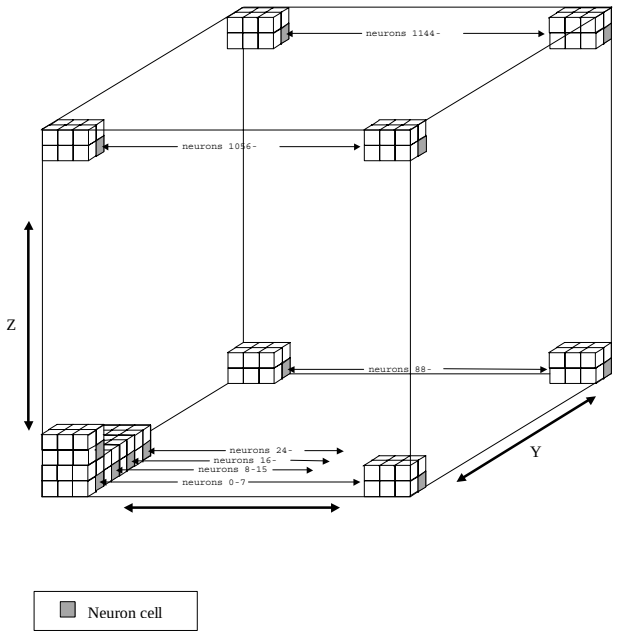


Figure 1: Position of the neurons.

one brain is 74,465,280. The cell configurations, as well as the connections between neighbouring cells are not fixed but can be "programmed". The way this is done in the CBM is by means of a genetic algorithm. This genetic algorithm is also implemented in the hardware, and we will explain it more detailed in a later section.
Our goal was to analyse this built-in genetic algorithm, discover some of its weaknesses and present a better alternative. Before we go into details, we start with a brief introduction about genetic algorithms, followed by an architectural overview of the CBM. That will present the necessary basic knowledge to understand the rest of this paper.

2 A summary of genetic algorithms

Genetic algorithms were invented in the early seventies by John Holland. These meta-heuristic search algorithms try to find a certain optimum for a set of parameters, based on a qualitative measure (the fitness value). Unlike other search heuristics, who operate on just one solution, genetic algorithms operate on a whole set of solutions, which is called the *population*. Each individual in the population is equal to a point in the search space and is called a *chromosome*. In fact these chromosomes are a concatenation of the parameters we're trying to optimize, hence the analogy with human genetics.

The process of searching an optimal solution is turned into an evolutionary process, analogous to Darwinism, where the individuals who are best adapted to their environment (i.e. having the highest fitness values) stand a better chance to survive. The best individuals get the chance to reproduce and in this way we get an evolutionary proces consisting of several *generations*.
The transition from one generation to another is done by means of the genetic operators. These are the common genetic operators.

- *Selection*: this operator simply selects which chromosomes are copied from one generation into the next.
- *Crossover*: starting from two parent chromosomes, this operator "builds" a new child chromosome. The intuitive idea behind this is that a better child can be produced when both

parents have good properties. In this way the child can combine good properties of both parents.

- *Mutation*: this operator alters one or more parameters in the chromosome, thus allowing a kind of diversity in the population. One can consider mutation as an operator introducing a certain amount of random search in the searching process.

3 Overview of the CAM–Brain Machine

The CAM–Brain Machine (CBM)[1] is a hardware tool which allows us to study a kind of artificial brain. The central idea is a three- dimensional automaton, that can be seen as a cube of 24×24×24 cells, each connected to their six nearest neighbours. These connections can be either input buds or outputs buds. Figure 2 shows the possible connections between two neighbouring cells.

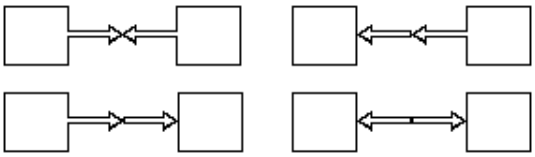


Figure 2: The connections between neighbouring cells.

Since the system of connections between the cells also applies to the cells on the edges, this results in a wrap-around mechanism, making the cube a toroidal one.

The neural network model used in the CBM is a digital model: the CoDi-model. CoDi stands for "Collect and Distribute" and applies to the way the information is processed in the modules. The model is digital since the signals between cells are digital, a restriction imposed by the hardware implementation. If a cell sends a 1-signal to another cell we say that the first cell *fires*. The 1-signal is called a *spike*, a terminology also used in the field of *Spiking Neural Networks*[2]. The processing of information can be subdivided in three consecutive phases:

1. gathering of information;
2. processing of information;
3. distribution of information.

A special type of cell was devoted to each of these phases.

- Dendrites: they take care of the collection of information. In the CoDi-model dendrites have up to five inputs and one output and work according to the mechanism of *spike-blocking*. This entails that a dendrite only fires if exactly one of his inputs receives a spike. If two or more spikes are received, the dendrite blocks the output (does not fire).
- Neurons: these cells are the processing cells of the machine. In the CoDi-model their number of inputs can vary from zero to six (although a neuron with only inputs or only outputs does not make a lot of sense). Additionally all inputs have a synaptic weight, analogous to the weight of inputs for neurons in a conventional artificial neural network (e.g. perceptrons). However, due to hardware restrictions these values are constrained to -1 or 1. Each neuron also has a 4-bit accumulator summing the input signals. Thus the accumulator can hold values in the interval $[-8, 7]$. If the threshold[3] exceeded or equalled, the neuron will fire. When

[1] The abbreviation CAM stands for *Cellular Automata Machine*.
[2] More information about Spiking or Pulsed Neural Networks can be found in Maass (1999).
[3] In the current CoDi-model this threshold is fixed and equals 2.

a neuron fires, it's accumulator is reset to 0, in all other cases the accumulator value remains unchanged. When the accumulator equals -8 and a negative weight is added (underflow) the accumulator is reset to 0 without firing.

- Axons: these cells take care of the distribution of information. In the CoDi-model each axon has exactly one input and five outputs[4]. If an axon receives a spike on his input, it will pass a the spike to all his outputs, if there is no spike nothing will be passed to the outputs.

- For the sake of completeness we also mention a fourth type of cells: empty cells. These cells are not used.

The complete configuration of a module can be seen as a collection of all the cell information, i.e. all the information about the connections, combined with the information about the cell types. Such a set of characteristics is called a *phenotype* in genetics. The phenotype simply describes the visible properties of an organism.

Genotype versus phenotype
When one decides to use genetic algorithms to guide the evolutionary process one has to find a suitable representation for the individuals in the population. One such representation of a module is to concatenate all characteristics of all cells in the module into one large chromosome. Another way of characterisation is to describe how the module is built. Such a building plan or blueprint is called a *genotype*. Using a genotype is in most cases a more compact way to encode information. A concatenation of all parameters can be seen as a very simple genotype, leaving out any compactness.

Consequences of a digital model
Since we are working with a digital model, we have to find a way to encode information into sequences of 1's and 0's (such a sequence is called a *spiketrain*). In order to do this we have to choose a certain interpretation of a spiketrain. Possible solutions are:

- deducing information based on frequencies, e.g. counting the number of spikes during a certain time window.

- deducing information based on the length of the period between two spikes.

The method chosen in the CBM is Spike Interval Coding Representation (SIIC) which uses a convolution filter to transfer the information in spiketrains to an analog, time-dependent signal. The inverse processing is done by deconvolution with the Hough Spiker algorithm. Both convolution and deconvolution are done by a hardware unit. In our opinion, this approach seems too simple to model real brain-like information processing, and better alternatives are certainly to be found, a very interesting topic for future research. More information about coding information in series of spikes can be found in Rieke et al. (1996).
With one module, however, we won't get far. Therefore there has to be a way to combine various modules to compose one large "brain". This is done by providing the module with various I/O points. Each module in the CBM has 192 I/O points: 188 inputs and (only) 4 outputs. By means of the I/O points modules can receive input from up to 188 other modules and send information to 4 other modules. However, the total number of modules in one brain can be up to 64,000.

Hardware architecture of the CBM
The whole architecture of the CBM is based around the Xilinx XC6246 FPGA (Field Programmable Gate Array). These FPGA's are reconfigurable hardware components (also known from the field of evolvable hardware) which allow a rapid reconfiguration of the internal logic, making them particularly convenient for an "evolving" neural module. A more in-depth treatment of the CBM can be found in Saeys (2000).

[4]In the CBM however an axon with 3 inputs and 3 outputs is perfectly possible, but this advantage is not used in the CoDi-model.

4 The built-in genetic algorithm

4.1 General view

Like in all genetic algorithms, we evolve a population of chromosomes and allow it to converge to the desired solution. In the CBM the chromosomes represent whole neural networks. Each chromosome describes the *genotype* of a whole neural module, resulting in a bitstring with length 91,008. In general, the GA in the CBM looks like this.

- Create an initial population of chromosomes. Each chromosome represents the genotype of a module.
- Repeat the following steps until the maximal number of iterations is reached:
  1. Calculate the fitness for all individuals in the population;
  2. Apply crossover to the best individuals. This way new individuals are born.
  3. Apply mutation to the new individuals.

4.2 Evaluation of a chromosome

The way the evaluation of each chromosome is done in the CBM is quite specific and consists of two consecutive phases: a growth phase and a signal phase.
The idea in the CBM is to "breed" a set of modules. To do this we actually have to build up a module from zero, step by step. When we start to build, the whole module consists only of blank cells. Then each cell is loaded with its genotype information. In the next step some cells are seeded as neurons. The other cells on neuron positions will later on become either axons or dendrites. After that, the neurons send *growth signals* to their neighbours, alternating between growth signals for axons and dendrites. An empty cell that receives a growth signal turns into an axon or dendrite, according to the particular growth signal. Each cell passes an incoming growth signal to his neighbouring cells. This process is repeated during 96 cycles, resulting in a neural network. During the growth phase the genotype bitstring (chromosome) is turned into a phenotype bitstring.

Whereas the growth phase can be considered as a preparatory phase, the signal phase effectively performs the fitness evaluation of a chromosome. During this phase the input signals are presented to the module and the signals are propagated through the whole neural network. The output signals are then compared to the target vectors, resulting in a fitness value that describes how much the actual output deviates from the desired output.

4.3 Genetic operators

Selection is done in a software module on the host computer. This host computer is connected to the CBM and also serves as an interface to the user. During selection the individuals are sorted according to their fitness and the best ones are kept for the next phase (crossover).
Crossover and mutation are implemented using bitmasks, which are layered on the chromosome bitstrings. To reproduce a child, the bits with a 0 in the mask are taken from one parent, while the other bits are taken from the other parent. In that way we get a child with characteristics of both parents. During mutation, the bits corresponding to 1's in the mask are flipped, the others are kept unchanged. To avoid having to load a new mask for each operation a simplified mechanism (due to hardware limitations) was implemented. The bitmasks are loaded only once at the beginning of the GA, and after that these masks are shifted after each operation. The idea behind this is that new masks are generated by shifting an old mask, thus providing a certain randomness. Unfortunately this mechanism just shifts in 0's on the right, whereas it might have been better to rotate the bits.

5 Shortcomings of the built-in genetic algorithm

The genetic algorithm in the CAM-Brain has some severe limitations, some of them due to hardware restrictions, others due to an inaccurate design.

A first problem is the limited coverage of the genotype. The genotypes lead to the construction of phenotypes that don't make use of the full architectural capabilities of the CBM: currently it is impossible to have another kind of cell than a neuron, an axon or a dendrite. Nevertheless cells like an axon with three inputs and three outputs have proven to be useful. Another problem, related to the building strategy of the genotype, is that it implies a certain structure: first the positioning of the neurons, then consecutive phases of growing axons and dendrites. This way axonic and dendritic trees are grown in the module. When an axon and a dendrite grow next to each other they automatically make contact. This way it is very hard to get a connection from one neuron to another without connecting with any neuron in between.

The idea of using a genotype is to represent encoded information in a compact way. Later on this information is used to build up a module, described by its phenotype. The problem in the CBM is that the length of the genotype is equal to the length of the phenotype, so the genotype is *not* more compact than the phenotype. Considering the length of the chromosomes only, there is no benefit for the GA in working with the genotype instead of the phenotype.
Furthermore, the genotype is very large. Each chromosome is a bitstring of length 91.008. One can easily see that a more compact representation of the genotype leads to a smaller search space. Thus, a GA operating on a smaller genotype will usually converge faster.

Another issue is the implementation of the genetic operators. The fundamental problem with the genetic operators is that they're not well–structured and don't take into account the way the module is structured. Since the bitmasks for crossover and mutation are just shifted, thus performing a random crossover and mutation, there is a high chance that we're just combining rubbish, whereas the task of the crossover operator is to combine structured pieces of useful genetic material. Normally the crossover operator has to create children that are a combination of characteristics of both parents, but by applying random crossover the algorithm clearly does not.

A last problem is that it is impossible to reuse parts of modules. One can save only a whole module and reuse it afterwards. The problem arises when two modules have parts in common. Since only whole modules can be used, such common parts have to be evolved over and over again each time a new module needs them.

6 Suggestions for improvement

In order to bypass the problems with the current GA we had to think of a new GA that offered the possibility to take advantage of the full capacity of the architecture, and additionally provides a way to reuse parts of modules. In Saeys (2000) we worked out three different methods based on a partition of the module in basic blocks. These basic blocks are used to provide a structural exchange of information, which takes into account the way the module is built. Each basic block consists of 3×2×2 cells (Figure 3).

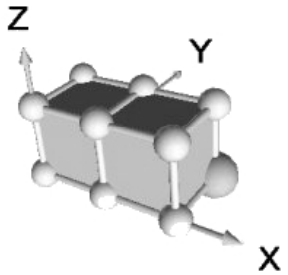


Figure 3: A basic block.

The partition of a module in basic blocks is shown in Figure 4. To keep it simple and clear, we only show a part of the cube with dimension 6, but we keep in mind that the real cube in the CBM has dimension 24. In Figure 4, the image on the left shows only the grids and the cells. The drawing in the middle of the same figure shows a partition of the cube in 3×2×2 basic blocks. We used the 3×2×2 basic blocks because the number of neurons in a module is limited, and in that way each basic block contains one neuron position. The image on the right in Figure 4 shows a view of the basic blocks in a (part of a) module.

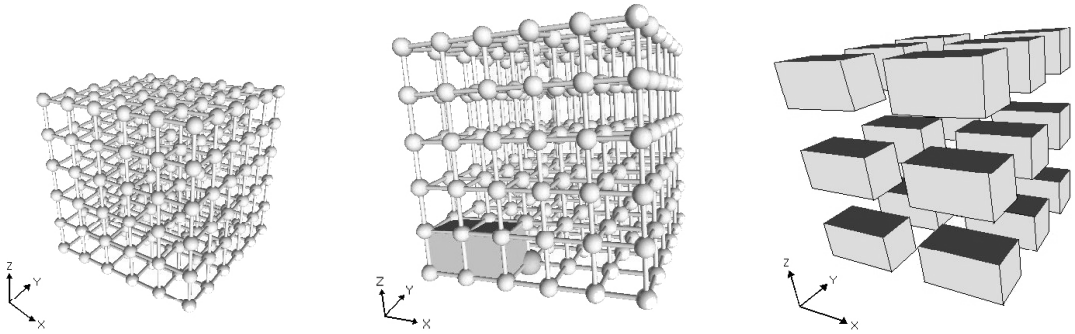


Figure 4: Partition of the module in basic blocks.

Furthermore we decided to separate the two phases in the evaluation of modules and drop the first phase (growth phase) and the genotype. Working directly with the phenotype allows us to express more module configurations, but leaves out the possibility to really build up a module. Nevertheless working with a genotype to build up a module is a good idea, as long as a powerful genotype can be used. The quest for a new genotype for the CBM remains a topic for future research.

By defining a new representation of the module (as a set of basic blocks) we also need to construct a customized crossover and mutation operator. These new operators will work on the basic block level and will impose a more structured genetic algorithm.

6.1 Crossover

Given two parent chromosomes (each built up as a set of basic blocks) the crossover operator has to create children that are a combination of characteristics of both parents. Herein the original algorithm failed. Our approach will present a way of exchanging subcomponents between the parents, resulting in two children.
These subcomponents can be compared with Tetris-like[5] building blocks. Each cube in such a component represents a basic block of 3×2×2 cells. Various components are stored in a library and provide different shapes (Figure 5).

The algorithm to exchange subcomponents works as follows.

1. Choose a random component from the library.
2. Choose a starting block in the mother and father module and map the chosen component onto the module. This way we select different basic blocks in each parent, resulting in two congruent shapes.
3. Exchange the two selected components.

[5]Tetris is a puzzle-like computer game where the player has to fit various different-sized blocks into a two-dimensional grid.

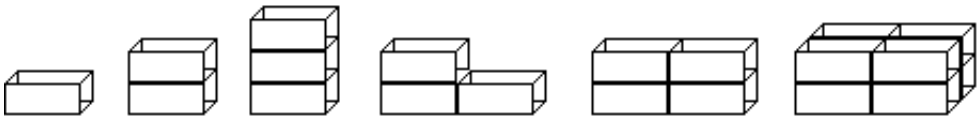


Figure 5: Library of components.

This is just a basic version of the crossover algorithm. More advanced options can be found in Saeys (2000).

6.2 Mutation

Mutation performs a kind of random search in the search space, and one has to be careful that by using mutation each point in the search space can be reached. Our mutation operator is a combination of the classical mutation and some additional functionality. The classical part picks a random basic block, and alters one of the characteristics of the block. An advantage of the use of basic blocks is to create an additional library of evolved components (each component consisting of some interconnected basic blocks). These components can be viewed as subnetworks performing a specific task e.g. a logical and-port or a random generator. By constructing such a library of useful components we introduce the possibility to swap in components during the mutation phase of the genetic algorithm, thus performing a reuse of components already evolved. This is the additional functionality of the mutation operator.

7 Experiments

Experiments with the original GA showed that the crossover operator didn't behave in a constructive way. The results of the algorithm didn't get worse when we left out the crossover operator, and even tended to be better with a higher mutation rate. From this we can conclude that the genetic algorithm isn't really working properly, but rather performs a random search.

The first tests with the new algorithm showed that we were now able to develop more complex circuits, e.g. an invertor with only one input. Such a circuit could not be evolved with the original GA while the new algorithm finds it easily. Such an invertor can be seen in Figure 6.

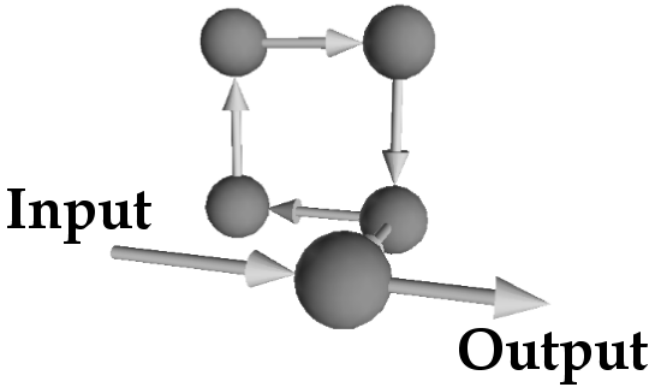


Figure 6: An invertor.

In this circuit all cells are initially firing, providing a constant series of 1-signals at the output. However, if a spike is presented at the input, the output dendrite receives two incoming signals and blocks, causing the output to produce a 0. With the original algorithm, the only way to get signals into the module is by passing a signal to one of the module inputs. As a consequence there can't be any firing signals present in cells other than the input cells in the first cycle of the signal phase. This is the basic reason why a NOT-port like the one in Figure 6 can not be evolved by the original algorithm.

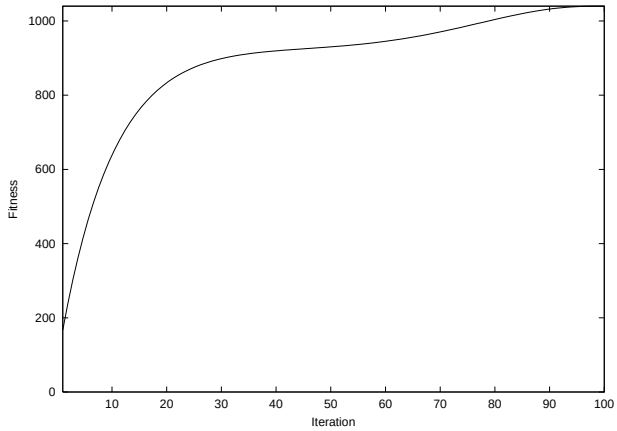


Figure 7: Evolution of an invertor.

The new algorithm finds the NOT-port easily, as can be seen in Figure 7. This simulation was done using a population of 100 individuals with 0.8 as crossover rate and 0.001 as mutation rate. In most cases the new algorithm converged faster than the original one, and achieved better solutions (solutions with a higher fitness value). Figure 8 shows the fitness for the evolution of a simple on-switch for both the old algorithm (original) and a basic version of the new algorithm. The optimal solution of this problem had fitness 2070. We used a population of 300 individuals with 0.8 as crossover rate and 0.001 as mutation rate.

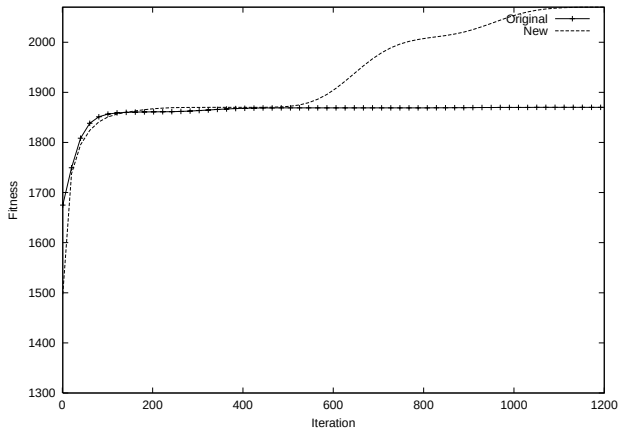


Figure 8: Comparison of the original and the new algorithm.

Figure 8 shows that the fitness values of the new algorithm start at a lower point than the one of the original algorithm. This can be explained by the fact that the modules in the original algorithm are really built up, so there's a lot of connectivity. The new algorithm doesn't use a

building phase but directly operates on the phenotypes. It's clear that this new algorithm will need some time to discover well-connected modules, but once this is done the algorithm not only converges faster but also achieves better solutions. In fact the new algorithm finds an optimal solution (this is a correct solution with a minimal delay between the input and output cells).

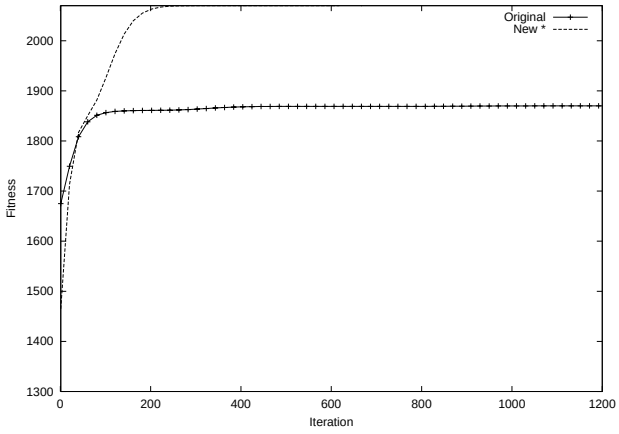


Figure 9: Comparison of the original algorithm and New.*

Another reason for the lower initial fitness values is the value of the phenotype bit that indicates whether a cell is firing or not. We already explained that in the original algorithm no other cells than module inputs could be firing during the first cycle of the signal phase. With the new algorithm every cell can fire initially, which can be useful (e.g. the NOT-circuit). However, in other cases it makes it harder to find a good solution since these fire bits generate some noise inside the module, thus disturbing the incoming signals. We worked around this difficulty by setting all fire bits in the initial population to 0. Later on these bits can still be altered by mutation. Figure 9 shows the original algorithm and the adapted new one (called New*). Again we see a considerable improvement in the fitness, and a very fast convergence to the optimal solution.

In the case of the full switch the difference in convergence speed becomes even more clear. Figure 10 shows the results for the original algorithm versus the new one.

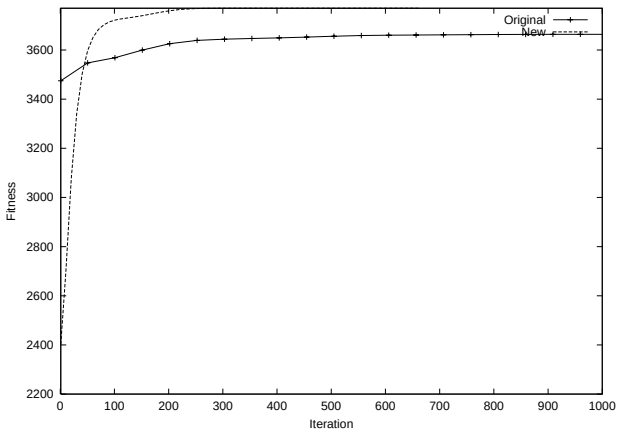


Figure 10: Comparison of the original and the new algorithm.

Using the CBM for optical character recognition
Several experiments were done to explore the capabilities of the CBM in the field of character recognition. We tried to evolve different modules that would be able to distinct specific characters from others. An example of this is the pattern for the character 'a' in Figure 11.

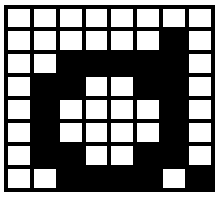


Figure 11: The character 'a' seen as a pattern of pixels.

The goal of this experiment was twofold. On one hand we wanted to see if a module was capable of evolving a very complex structure. On the other we tried to find out if this way of presenting patterns to the CBM is a good one. The character is drawn on a 8×8 canvas of pixels. In this way the pattern can be mapped onto one side of a module (which had a grid of 8×8 IO's). We then tried to evolve a module that could make the difference between an 'a' and several other characters. The results of these experiments were not really good. What happens is that we're in fact training a module which has to operate as a giant AND-gate. Since the character 'a' consists of 23 pixels the ideal module should only give an output signal if all of these 23 pixels are present. This way the module performs an AND-operation on 23 inputs. As a consequence the pattern is not recognized if we shift it one or more positions. This shows that this way of presenting patterns to the CBM is not a good idea.
Another problem is the complexity: an AND-gate with 23 inputs with the appropriate routing from specific inputs to one of the outputs might be too complex for the algorithm to discover. In order to enhance the CBM with more capabilities for pattern recognition we need to find a more suitable representation of the patterns, not just a simple transformation onto the inputs of the module, but a more generic way to decode the information into spiketrains. The coding of information into spiketrains, and especially how to use them in the CBM remains a topic for further research.

8 Conclusion

In this paper we presented the CAM-Brain Machine and the genetic algorithm used to let the brain develop its neural modules. We also showed that the built-in algorithm has several weaknesses, the most important being a limiting representation and a non-constructive crossover. To overcome those disadvantages we designed a new genetic algorithm, using basic blocks. The results of this new algorithm are very encouraging, and could lead to a more component-based view of brain building with the CBM.
It goes without saying that there's still a lot of work to do, but with this new genetic algorithm we provided a basic framework for future evolutionary algorithms for brain-like systems as the CBM.

References

Holland, John H. (1975) Adaptation in natural and artificial systems, MIT Press.

Korkin, M., de Garis, H., Gers, F., Hemmi, H. (1997) "CBM (CAM-Brain Machine)" a hardware tool which evolves a neural net module in a fraction of a second and runs a million neuron artificial brain in real time.

Saeys, Y. (2000) Studie en optimalisatie van het genetisch algoritme in de CAM-Brain neurale computer, Last year thesis (in Dutch), University of Ghent, Belgium.

Rieke, F., Warland D., De Ruyter Van Steveninck R. and Bialek W. (1996) Spikes : Exploring the Neural Code (Computational Neuroscience), MIT Press.

Maass, W., Bishop C.M. (1999) Pulsed Neural Networks, MIT Press.

Parsing Ambiguous Context-Free Languages by Dynamical Systems

Disambiguation and Phase Transitions in Neural Networks with Evidence from Event-Related Brain Potentials (ERP)

Peter beim Graben*, Thomas Liebscher and Douglas Saddy
Universität Potsdam,
P.O. Box 601553
Institute of Linguistics,
D - 14415 Potsdam
peter@ling.uni-potsdam.de

Abstract

In an ERP-study, we examined the processing of initially ambiguous German wh-questions where we observed an anteriorly distributed P600 in the number mismatch condition but no significant effect in voltage averages of the case condition. We developed an alternative approach for analyzing event-related brain potentials by means of symbolic dynamics and measures of complexity of sliding cylinder sets. By applying these methods to the ERP data we observed that the P600 ERP component corresponds to a large drop in cylinder entropy. On the other hand, we found an early and a late increase of the cylinder entropy for the case clash condition. In symbolic dynamics theory any string of finite or infinite length can be mapped onto a real number lying in the unit interval by a g-adic expansion. In the same manner, any bi-infinite sequence can be represented by a point in the unit square. By applying this technique cylinder sets correspond to rectangles in the unit square. On the other hand cylinder sets can be easily interpreted as states of a stack automaton. We applied two different g-adic expansions to stack and input words of a stack automaton. This leads to a map of the parser's states onto rectangles in the unit square. We developed an ambiguous context-free model grammar for the structures examined in our ERP experiment and processed them by a top-down parser obtaining a list of parsing states. This list has been translated into a sequence of rectangles lying in the unit square by the g-adic expansion algorithm. Finally, these patterns are trained with a neural network using an additional input unit as control parameter representing the strategies of the parser. We study the impact of the control parameter settings and observed phase transitions of the dynamics that can be detected by cylinder entropies. In order to provide an interface to ERP data we choose a measurement partition of the unit square. The symbolic dynamics of the modeled parsing trajectory shows an entropy drop for the number mismatch condition with respect to the measurement partition, while the entropy of the case clash condition increases at the critical parsing step recapitulating our ERP findings.

Keywords: Event-related brain potentials, Symbolic dynamics, Cylinder sets, Entropies, Turing-Machines, Stack automata, Gödel-Code, g-adic Expansions, Neural Networks, Context-free languages, Ambiguity Resolution, Parsing

1 Introduction

Phase transitions in the human brain are well known in connection with cognitive motor control (Kelso et al., 1986; Haken, 1996; Kelso et al., 1992; Engbert et al., 1997) and have been successfully

*also at Institute of Physics, Nonlinear Dynamics Group

modeled by nonlinear field theories of cortical activation and by synergetic computers (Jirsa and Haken, 1996; Haken, 1996). But recently, phase transitions have been observed also in human language processing by Raçzasek et al. (1999) using continuously varying prosodic parameters for the disambiguation of speech. Although there are some approaches to model language processing by recurrent neural networks (Elman, 1995) most efforts in this field have been done in developing symbol manipulating automata (Aho and Ullman, 1972; Mayer, 1986).

In order to bridge the gap between dynamical and connectionist models on one hand and symbolic computing on the other hand, Moore (1990, 1991b) proved the formal equivalence of Turing-machines and nonlinear dynamical systems using methods from symbolic dynamics. He demonstrated that any symbolic system can be mapped onto a piecewise linear map defined at the unit square by interpreting the symbols as integers from some g-adic number system. Strings of symbols will either be assigned to g-adic fractions or to points in the unit square by this algorithm.

Here we are going to use Moore's technique for mapping top-down automata that recognize context-free languages onto such dynamical systems. To capture the dynamics of nondeterministic parsers we decompose an ambiguous grammar into a set of nonambiguous grammars that can be processed by deterministic top-down automata. After representing these parsers by nonlinear dynamical systems, their numbers constitute the values of a control parameter. The maps together with the control parameter can then be used to train a simple feed-forward neural net representing a bifurcating dynamical system at the unit square. The processing of ambiguous structures, garden path interpretations and disambiguation can be modeled as phase transitions of this dynamical system elicited by changes of the system's control parameter.

Symbolic dynamics is also an appropriate tool to analyzing natural data. It has been successfully applied to physiological data like cardiorespiratory time series (Kurths et al., 1995; Schiek et al., 1998) and movement control (Scheffczyk et al., 1997; Engbert et al., 1997), but also to neuronal spike trains obtained by electrophysiological measurements (Rapp et al., 1994). Symbolic dynamics is able to tackle nonstationarity as has been shown by Schwarz et al. (1993) and, more recently, by Buchner and Zebrowski (1999). In the framework of symbolic dynamics phase transitions can be detected by using measures of complexity (Badii and Politi, 1997; Wackerbauer et al., 1994).

In a recent study we presented an approach that uses symbolic dynamics to analyze event-related brain potentials (ERP) (beim Graben et al., 2000b). We showed that the information theoretic notions of cylinder sets are appropriate tools in order to deal with ensembles of non-stationary data. The calculated measures of complexity are able to capture the coherency and disorder of ERP. Large entropy drops correspond with traditional ERP-voltage averages. But on the other hand, measures of complexity are able to detect phase transitions towards higher disorder that cannot be revealed by averaging voltages. Thus we have confirmed that ERP can be considered as indicators of phase transitions in the human brain as has been proposed by Başar (1983).

2 The ERP Experiment

2.1 Setup and Material

We performed a language processing experiment in which subjects were seated in front of a monitor for stimulus presentation. They were wired with the EEG-amplifier device by a 32 channel electrocap. The EEG-data have been recorded in 25 channels according to the international 10 - 20 system at a sampling rate of 250Hz with the ScanAmps/NeuroScan recording system against reference at the left mastoid bone. Four channels were used for measuring EOG artifacts. We report the data analysis of 16 subjects (15 female) aged 19 - 25. The subjects were volunteers, native speakers of German and not familiar with the purpose of the study.

The ERP experiment was performed using 30 structurally equivalent sentences for each condition and five conditions total. We report here the results of two experimental conditions: number-mismatch and case-mismatch against their corresponding control conditions. The sampling of ERP epochs started 200ms before the critical word occurred and finished 1000ms after. Thus, at time $t = 0$ms the critical words appeared. The sentences presented to the subjects were initially

ambiguous interrogative sentences. The point of disambiguation was either at the verb (number-mismatch) or at the second article (case-mismatch). The control sentences were not ambiguous at these points. Table 1 shows examples of the German sentences, an English paraphrase and the correct English translation. The critical words where ERP measurement took place are printed in italics. For details see Saddy et al. (1999); an introduction into language related brain activity can be found in Kutas and van Petten (1994); Friederici (1999).

Interrogative sentences in German are usually expected to be of subject-verb-object order (control conditions). But object-verb-subject order is also possible (conditions: number- and case-mismatch). The first two words in the sentences, "welche Frau", are ambiguous with respect to nominative or accusative case and are always the same. Subjects expect the sentences to be of the frequent subject-verb-object structure. There are two ways for disambiguating such sentences; either by the number-feature: subject-noun and verb must agree in number (singular or plural) or by the case-feature: the subject-noun has nominative case, whereas the object-noun bears accusative case. In the number-mismatch condition the disambiguating information comes at the verb that does not agree with the subject-noun in number: the subject noun "Frau" is singular, the verb "sahen" is plural. The sentence seems to be ungrammatical at that point and a reanalysis of the structure built so far must take place. In the case-mismatch condition the additional information is provided by the article "der" that is nominative case marked (an accusative case marked article would be "den"). As in the number-mismatch condition, the sentence must be reanalyzed.

Condition	Example	Paraphrase	Translation
number-mismatch	welche Frau *sahen* die Männer ?	which woman (accusative singular) saw (plural) the men (nominative plural) ?	which men saw the woman ?
control number	welche Frau *sah* den Mann ?	which woman (nominative singular) saw (singular) the man (accusative singular) ?	which woman saw the man ?
case-mismatch	welche Frau sah *der* Mann ?	which woman (accusative singular) saw (singular) the man (nominative singular) ?	which man saw the woman ?
control case	welche Frau sah *den* Mann ?	which woman (nominative singular) saw (singular) the man (accusative singular) ?	which woman saw the man ?

Table 1: Example sentences of the language processing ERP experiment. Critical words (where ERP measurements happen) are printed in italics.

2.2 Results of Voltage Averages

We report the results of the grand average analysis from all 16 subjects at the electrode Fz (fronto-central site). In the number-mismatch condition compared to its control condition we observed a positive peak 600ms after stimulus onset. This is the so-called P600 ERP component indicating reanalysis processes in the brain's language processing system (Friederici, 1999). This component appears to be highly significant by means of a running t-test ($p < 0.01$). In contrast, comparing the voltage grand averages of the case-mismatch condition with the corresponding control condition there is no significant event by means of the running t-test. We noticed only a very late effect after 800ms where the voltage averages of the case-mismatch condition vanished. This might be a late positivity with respect to the control condition. But it appeared to be barely significant ($p < 0.1$) (Saddy et al., 1999; beim Graben et al., 2000b).

3 Symbolic Dynamics

3.1 One-dimensional systems

A time discrete deterministic dynamical system is a pair (X, Φ_r), where X is the *state space* (sometimes called *phase space*) and the *flow* $\Phi_r : X \to X$ is a map that assigns to a state x_t at a certain time t its successor x_{t+1} at time $t+1$, occasionally depending on a control parameter r. A given state x_{t_0} for a certain time t_0 is called *initial condition* of the dynamics. The set of states $\{x_t | x_t = \Phi_r^t(x_{t_0}), t \in \mathbb{Z}\}$ generated by the flow from the initial condition x_{t_0} is called a *trajectory* of the dynamical system. The powers of the map Φ_r^t are recursively defined by iterating the map: $\Phi_r^t = \Phi_r \circ \Phi_r^{t-1}$.

A special class of dynamical systems is given by the unit interval $[0, 1]$ as their state space and by a nonlinear function Φ_r mapping the unit interval on itself. These systems are called one-dimensional dynamical systems (Collet and Eckmann, 1980). The most simple one-dimensional dynamics are defined by piecewise linear maps at the unit interval. In the following, we shall discuss the Bernoulli map as an example (Schuster, 1989).

The Bernoulli map lives in the unit interval $[0, 1]$ as its state space and the map is given by $\Phi(x) = 2x \bmod 1$, i.e. the map is provided by the linear function $y = f_0(x) = 2x$ at the subinterval $[0, 0.5]$ and by the linear function $y = f_1(x) = 2x - 1$ at the subinterval $]0.5, 1]$. Though being piecewise linear, the map is nonlinear at the whole state space $[0, 1]$ and exhibits the typical stretching and folding properties of chaotic dynamical systems. The Bernoulli map does not depend on a control parameter.

An intriguing description of the Bernoulli's map dynamics is obtained by a binary expansion of its states: any number in the unit interval can be represented by a binary fraction. Numbers smaller than 0.5 will start with their *most significant digit* '0' while numbers greater than 0.5 begin with the binary digit '1'. For an example, let us consider the initial condition $x_0 = 0.1101$. Its decimal expansion is given by

$$x_0 = \sum_{i=1}^{\infty} a_i \cdot 2^{-i} = 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 0.8125\,, \tag{1}$$

where a_i are the binary digits. What are the successors of x_0 lying at a trajectory of the system? In the decimal number system, we obtain $x_1 = \Phi(x_0) = 0.6250$, $x_2 = \Phi(x_1) = \Phi^2(x_0) = 0.2500$, $x_3 = \Phi(x_2) = \Phi^3(x_0) = 0.500$, $x_4 = \Phi(x_3) = \Phi^4(x_0) = 0.0$. Using the binary number system yields the trajectory $x_1 = \Phi(x_0) = 0.101$, $x_2 = \Phi(x_1) = 0.01$, $x_3 = \Phi(x_2) = 0.1$, $x_4 = \Phi(x_3) = 0.0$. It is easy to recognize that the Bernoulli map acts on the binary numbers as a shift to the left discarding the 2^0 position.[1]

The binary descriptions, e.g. '1101', of the states x together with the representation of the map Φ given by the left shift σ is called the *symbolic dynamics* of the Bernoulli map. The binary digits '0', '1' can be regarded as symbols from a finite alphabet A. States of the dynamical system will be mapped onto sequences $s = a_{i_1} a_{i_2} a_{i_3} \ldots$ of symbols stemming from A. Sequences of finite length will be called *words*. The set of all sequences of (one-sided) infinite length $s = a_{i_1} a_{i_2} a_{i_3} \ldots$ assumes the notation $A^{\mathbb{N}}$, where $\mathbb{N}$ is the set of positive integers. As noted above, the most significant digit a_{i_1} of a binary fraction decides whether the state is taken from the lower or from the upper half interval: $a_{i_1} =$ '0' means $x_0 \in [0, 0.5]$ whereas $a_{i_1} =$ '1' means $x_0 \in]0.5, 1]$. Hence, the most significant digit a_{i_1} *partitionates* the system's state space into disjunct subsets. In the same manner, the first two symbols $a_{i_1} a_{i_2}$ of a sequence decide where the initial condition x_0 and its first iterate $x_1 = \Phi(x_0)$ are situated. In the example discussed above, '11' means: $x_0 \in A_1$ and also $x_1 \in A_1$. The latter expression is equivalent to $\Phi(x_0) \in A_1$ and this can be reformulated to $x_0 \in \Phi^{-1}(A_1)$ where Φ^{-1} denotes the preimage of some set. This is always defined even if the map itself is not invertible as the Bernoulli map. By generalizing this expression, we obtain

$$s = a_{i_1} a_{i_2} a_{i_3} \ldots \Rightarrow x_0 \in \bigcap_{k=0} \Phi^{-k}(A_{i_{k+1}}) \tag{2}$$

[1] That is exactly what the definition of the function Φ says: multiply by 2. i.e. shift to the left and subsequently take the remainder after dividing by 1.

When we apply this construction to the Bernoulli map we find that the first two symbols describe a partition of the unit interval into quarters: '00' means $[0, 0.25]$, '01' means $]0.25, 05]$, and so on. In general, by fixing the first n binary digits of a sequence $s = a_{i_1}a_{i_2}a_{i_3}\ldots$ we refer to the interval $[0.a_{i_1}\ldots a_{i_n}00000\ldots, 0.a_{i_1}\ldots a_{i_n}11111\ldots]$. Symbolically we write

$$[a_{i_1},\ldots,a_{i_n}] \Rightarrow \begin{cases} 0.a_{i_1}\ldots a_{i_n}00000\ldots \\ \quad\ldots \\ 0.a_{i_1}\ldots a_{i_n}00001\ldots \\ 0.a_{i_1}\ldots a_{i_n}00010\ldots \\ 0.a_{i_1}\ldots a_{i_n}00011\ldots \\ \quad\ldots \\ 0.a_{i_1}\ldots a_{i_n}11111\ldots \end{cases} \tag{3}$$

and call the set of symbol strings described by the corresponding interval a *cylinder set*. A cylinder set of length n is therefore a set of all sequences coinciding in the first n letters of the alphabet.

3.2 General systems

Next, we shall introduce symbolic dynamics of general time discrete (or e.g. Poincaré map time discretized) dynamical systems depending on some control parameters $r \in \mathbb{R}^p$, whose state spaces are subsets or manifolds in some $\mathbb{R}^m$. A coarse grained description of the dynamics can be gained by a partition covering the state space of the dynamical system (Badii and Politi, 1997; Beck and Schlögl, 1993). Let $\{A_i | i = 1, 2, \ldots, I\}$ be a family of I pairwise disjunct subsets covering the whole state space X, i.e. $\bigcup_{i=1}^{I} A_i = X, \quad A_i \cap A_j = \emptyset, \quad i \neq j$. The index set $A = \{1, 2, \ldots, I\}$ of the partition can be interpreted as a (finite) alphabet of letters $a_i = i$. In the case of an invertible deterministic dynamics we are able to determine the system's past as well as its future by iterating the inverse flow Φ_r^{-1} and the flow Φ_r, respectively. By deciding which cell A_i of the partition is visited by a state x_t at time $t = \ldots -1, 0, 1, \ldots$ we assign a symbol $\ldots, a_{i_{-1}}, a_{i_0}, a_{i_1}, \ldots$ at each instance of time. Thus, we obtain a bi-infinite sequence of letters $s = \ldots a_{i_{-1}}a_{i_0}a_{i_1}a_{i_2}\ldots$ from A. The expression $A^{\mathbb{Z}}$ refers to the set of all these bi-infinite strings of symbols. Given a time discrete and invertible deterministic dynamics Φ_r we construct a map $\pi : X \to A^{\mathbb{Z}}$ that assigns initial conditions $x_0 \in X$ to bi-infinite symbol strings $s \in A^{\mathbb{Z}}$ by the rule $\pi(x_0) = s$, iff $x_t = \Phi_r^t(x_0) \in A_{i_t}$, $t \in \mathbb{Z}$. Thus, π maps initial conditions x_0 in the state space onto symbolic strings regarded as coarse grained trajectories starting at x_0. By doing so, the flow Φ_r will be mapped onto the left shift σ again. The shift is hence a map $\sigma : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ acting according $\sigma(\ldots a_{i_{-1}}\widehat{a_{i_0}}a_{i_1}a_{i_2}\ldots) = \ldots a_{i_{-1}}a_{i_0}\widehat{a_{i_1}}a_{i_2}\ldots$ where the hat denotes the current state under observation. The map π mediates between the quantitative states $x \in X \subset \mathbb{R}^m$ and the states of the symbolic dynamics $s \in A^{\mathbb{Z}}$ by

$$\pi(\Phi_r(x_t)) = \sigma(\pi(x_t))\,. \tag{4}$$

The map π might not be invertible. If π is invertible the partition is called generic and every string of symbols corresponds to exactly one initial condition generating it (Beck and Schlögl, 1993). Then Eq. (4) entails $\Phi_r(x_t) = \pi^{-1}(\sigma(\pi(x_t)))$, the maps Φ_r and σ are topologically conjugated, provided π being a homeomorphism.

Nevertheless, we can apply the map π^{-1} at subsets of $A^{\mathbb{Z}}$ namely at strings of finite length, looking for their preimages in X. In order to do this we generalize the notion of cylinder sets. Let $t \in \mathbb{Z}, n \in \mathbb{N}$ and $a_{i_1}, \ldots, a_{i_n} \in A$. The set

$$[a_{i_1},\ldots,a_{i_n}]_t = \{s \in A^{\mathbb{Z}} \,|\, s_{t+k-1} = a_{i_k}, \quad k = 1,\ldots,n\} \tag{5}$$

is called *n-cylinder* at time t. For further references see (Badii and Politi, 1997; Beck and Schlögl, 1993; beim Graben et al., 2000b; McMillan, 1953; Wackerbauer et al., 1994). Since cylinders are subsets of $A^{\mathbb{Z}}$ of infinite strings coinciding in a (discrete) time interval $\{t, t+1, \ldots, t+n-1\}$, we can determine their preimages under the deterministic dynamics Φ_r

$$\pi^{-1}([a_{i_1},\ldots,a_{i_n}]_t) = \bigcap_{k=0}^{n-1} \Phi^{-k}(A_{i_{k+1}})\,. \tag{6}$$

This is almost the same formula as Eq. (2).

As for the Bernoulli map any general symbolic dynamics can be mapped back onto a quantitative dynamical system by performing a g-adic expansion. In order to archive this the symbols of the alphabet A must be encoded by integers. If I is the cardinality of the alphabet we need I digits $0, 1, 2, \ldots I-1$ representing the letters $a_1, a_2, a_3 \ldots a_I$. The assignment $g(a_i) = i - 1$ is sometimes called *Gödel encoding*. Given a finite or bi-infinite symbolic sequence $s = \ldots a_{i_{-1}} \widehat{a_{i_0}} a_{i_1} a_{i_2} \ldots$ where the hat denotes the current state again, we expand the left-half sequence $s_- = \ldots a_{i_{-3}} a_{i_{-2}} a_{i_{-1}} a_{i_0}$ into the I-adic fraction $x = \ldots + g(a_{i_{-3}})I^{-4} + g(a_{i_{-2}})I^{-3} + g(a_{i_{-1}})I^{-2} + g(a_{i_0})I^{-1}$ and the right-half sequence $s_+ = a_{i_1} a_{i_2} a_{i_3} \ldots$ into the I-adic fraction $y = g(a_{i_1})I^{-1} + g(a_{i_2})I^{-2} + g(a_{i_3})I^{-3} \ldots$. The sequence s will so be mapped onto a pair (x, y)

$$x = \sum_{k=0}^{\infty} g(a_{i_{-k}}) I^{-k-1} \tag{7}$$

$$y = \sum_{k=1}^{\infty} g(a_{i_k}) I^{-k} \tag{8}$$

of real numbers lying in the unit square $[0, 1] \times [0, 1]$.

Furthermore, as cylinder sets of the Bernoulli map were represented by subintervals of the unit interval, we shall show that cylinder sets of an arbitrary symbolic dynamics (Eq. 5) are rectangles in the unit square. To this aim let us consider a cylinder of length $n + l + 1$ at time $-l$: $[a_{i_1}, \ldots, a_{i_{n+l+1}}]_{-l} = \{s \in A^{\mathbb{Z}} | s_{-l} = a_{i_1}, s_{-l+1} = a_{i_2}, \ldots s_0 = a_{i_{l+1}}, s_1 = a_{i_{l+2}}, \ldots, s_n = a_{i_{n+l+1}}\}$. The elements of this set can be split into two half-sequences: $s_- = \ldots s_{-l} s_{-l+1} \ldots s_0$ and $s_+ = s_1 s_2 \ldots, s_n \ldots$. According to Eq. (3) we obtain an interval of real numbers for the expansion of s_- and s_+, respectively. The lower bound of the expansion of s_- is

$$x_1 = \sum_{i=0}^{-l} g(s_i) I^{i-1} \,. \tag{9}$$

For the upper bound of the expansion of s_- we find

$$x_2 = x_1 + \sum_{i=l+2}^{\infty} (I-1) I^{-i} = x_1 + I^{-l-1} \tag{10}$$

where we made use of the limit of geometric series. Accordingly, we obtain the interval $[y_1, y_1 + I^{-n}]$ with $y_1 = \sum_{i=1}^{n} g(s_i) I^{-i}$ for the right-half sequence s_+. The cylinder set $[a_{i_1}, \ldots, a_{i_{n+l+1}}]_{-l}$ is therefore represented by the rectangle $[x_1, x_2] \times [y_1, y_2]$.

3.3 Symbolic Dynamics of Time Series

Let us consider an ensemble of time series $(x_i(t))$ of the dynamical system (X, Φ_r) obtained by a real valued observable h by $x_i(t) = h(y_i(t))$, where i $(1 \le i \le N)$ is the ensemble index and t is the (discrete) time index. N is the cardinality of the ensemble. The $y_i(t) \in X$ form an ensemble of trajectories in the state space. After choosing a partition $S_i, \quad i = 1, 2, \ldots, I$ of the set $H = h(X)$ leading to a partition of X (beim Graben et al., 2000b), we decide whether the values $x_i(t)$ belong to the sets S_j in order to assign a symbol $a_{i;k_t}$. Thus, the ensemble of time series will be mapped onto an ensemble of symbolic sequences

$$E = \{s_i | s_i \in A^L, 1 \le i \le N\} \tag{11}$$

where L is the length of the time series, i.e. the number of samples. Each string s_i is a sequence $a_{i;k_1} a_{i;k_2} \ldots a_{i;k_L}$ of L letters from the alphabet A. The cylinder sets (5) will be defined for measured data as $[a_{k_1}, \ldots, a_{k_n}]_t = \{s \in E \,|\, s_{t+l-1} = a_{k_l}, \quad l = 1, \ldots, n\}$.

If the time series are nonstationary we need a statistical formalism of transient behaviour of dynamical systems. This can be provided by a statistical description of initial conditions even if

the system is purely deterministic. Let us prepare an ensemble of initial conditions according to a probability measure μ_0. Now let the system evolve. What is the image of some set A under the action of the map Φ_r after time t? In the deterministic case the image $B = \Phi_r^t(A)$ must have the same probability as A. Therefore it holds $\mu_0(A) = \mu_t(\Phi_r^t(A))$ with μ_t as the probability measure after time t. This identity can be reformulated as

$$\mu_t(B) = \mu_0(\Phi_r^{-t}(B)) \tag{12}$$

for any measurable subset $B \subset X$ with $\Phi_r^{-t}(B) = (\Phi_r^t)^{-1}(B)$ as the preimage of B after time t. This is the Frobenius–Perron equation of the dynamics (Beck and Schlögl, 1993).

In the coarse-grained description provided by a symbolic dynamics measurable sets of states $B \subset X$ are given by cylinder sets due to Eq. (6). Let $t \in \mathbb{Z}, n \in \mathbb{N}$ and $a_{i_1}, \ldots, a_{i_n} \in A$ as before. The measure

$$p(a_{i_1}, \ldots, a_{i_n}|t) = \mu_t \left(\bigcap_{k=0}^{n-1} \Phi^{-k}(A_{i_{k+1}}) \right) \tag{13}$$

is called *word probability* of the n–word $(a_{i_1}, \ldots, a_{i_n}) \in A^n$ at time t. The distribution of all word probabilities for a given length n is called *word statistics*. For a measured ensemble E of time series the word probabilities will be estimated by the counting measures of the ensemble as the relative frequencies

$$\bar{p}(a_{k_1}, \ldots, a_{k_n}|t) = \frac{\#([a_{k_1}, \ldots, a_{k_n}]_t)}{N}\,, \tag{14}$$

where '$\#(\cdot)$' denotes the set theoretic cardinality function.

For detecting disorder-order phase transitions of the ensemble dynamics we use measures of complexity such as Shannon's entropy. The Shannon entropies (Shannon and Weaver, 1949) of order n at time t of the ensemble E will be accommodated as

$$H_n(t) = -\sum_{(a_{k_1}, \ldots, a_{k_n})} \bar{p}(a_{k_1}, \ldots, a_{k_n}|t) \times \log \bar{p}(a_{k_1}, \ldots, a_{k_n}|t)\,. \tag{15}$$

The quantities

$$H(t) = H_n(t)/n \tag{16}$$

measure the information per letter and are called *relative entropies*. Entropy is a measure of uncertainty of a given probability distribution. It reaches its maximum value +1 for uniformly distributed events. It takes its minimum 0 if there is only one certain event with probability 1.

4 Symbolic Dynamics of ERP-Data

We apply our techniques basing on symbolic dynamics on the ERP-data obtained by the language processing experiment discussed in section 2 (Saddy et al., 1999; beim Graben et al., 2000b). According to our discussion in section 3.3 a partition of the state space or a corresponding partition of the range of the observables that are generated by the dynamical system is needed in order to obtain a symbolic dynamics. We decided to use static encoding where the values of time series have been translated into symbols according to a certain threshold after we had transformed each voltage epoch to an uniformly distributed time series by a ranking procedure. Performing a binary static encoding according to the median ($F(x_{med}) = 0.5$) of each epoch separately as the decision point yields a symbolic sequence containing the same number of '0' ($r_t \leq 0.5$) and '1' ($r_t > 0.5$). Thus, ranking maximizes the entropy of each sequence.

Our concepts of running cylinder entropies (Eqs. 15, 16) are appropriate tools for quantifying patterns of the symbolic dynamics because they can be considered as a family of cylinder sets. Figure 1 presents the Shannon entropies calculated from the running cylinders of length $n = 1$ for the number-mismatch condition (solid line) and the control condition (dotted line) of the grand ensembles of all 16 subjects, respectively.

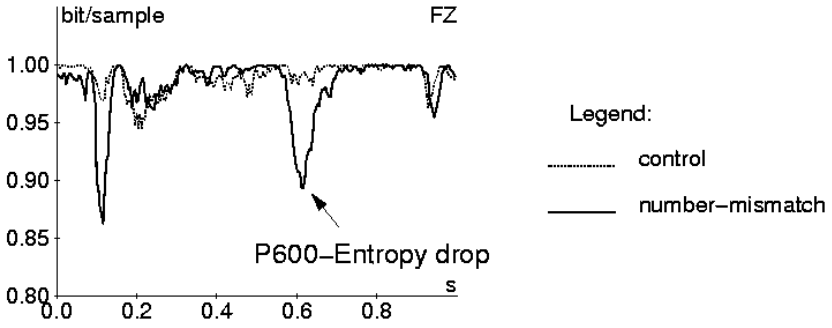


Figure 1: Running normalized Shannon entropies for the number-mismatch condition (solid lines) compared to the control condition (dotted) of the grand ensemble. Shannon entropy of 1-words ($n = 1$) in bit/sample against time (s), $t = 0$: stimulus onset time. The ungrammaticality effect, P600 (positive peak after 600ms), is marked by an arrow.

The entropy drop corresponding to the P600 ERP component comes out to be significant with $p < 10^{-23}$ by applying a running χ^2-test on the word statistics collected in a sliding 20ms window ($\chi^2 = 62.67$, $df = 2$). For details see beim Graben et al. (2000b).

In the same way, we computed the running cylinder entropy of the case-mismatch condition. Figure 2 shows the Shannon entropies calculated from the running cylinders of length $n = 1$ for the case-mismatch condition (solid line) and the control condition (dotted line) of the grand ensembles.

Although there were no significant effects in the voltage comparison, the symbolic dynamics reveals structures in the data without any counterpart in voltage averages. There is an early entropy increase about 360ms exhibiting significance $p < 0.05$ ($\chi^2 = 9.2$, $df = 2$) and a very late entropy increase at 880ms appearing significant with $p < 10^{-8}$ ($\chi^2 = 40.1$, $df = 2$).

5 Symbolic Dynamics of Automata

In order to describe Turing-machines as dynamical systems, Moore (1990, 1991b) introduced the concept of *generalized shifts*. Like the shifts discussed in section 3.2 the generalized shifts are maps defined on the set of bi-infinite symbolic strings $A^{\mathbb{Z}}$. Let $s = \ldots a_{i_{-1}} \widehat{a_{i_0}} a_{i_1} a_{i_2} \ldots$ be such a sequence where the hat denotes the current state. In Moore's construction the hat indicates the tape position of the head of a Turing-machine. A generalized shift can be characterized by a range of d symbols called *domain of dependence*. The generalized shift acts on this word w of length d by first replacing it by a further word w' of length d' and then by performing a shift k symbols to the left or to the right (Moore, 1990, 1991b; Badii and Politi, 1997; Politi and Badii, 1997). Moore has proven that these systems are computationally equivalent to any Turing-machine.

The g-adic expansion algorithm of symbolic sequences into points of the unit square (Eqs. 7, 8) can be applied to states of the generalized shift in quite the same way. It is therefore possible to map any Turing-machine onto a time discrete nonlinear dynamics living at the unit square. The corresponding map has been shown to be piecewise linear just as the Bernoulli map discussed in section 3.1 (Moore, 1990, 1991b; Politi and Badii, 1997; Badii and Politi, 1997).

In this contribution we are going to construct a map on the unit square that can be assigned to the behaviour of a stack automaton. Because formal languages accepted by stack automata belong to a lower class in the Chomsky-hierarchy than recursively enumerable languages accepted by Turing-machines (Hopcroft and Ullmann, 1979; Badii and Politi, 1997), any stack automaton

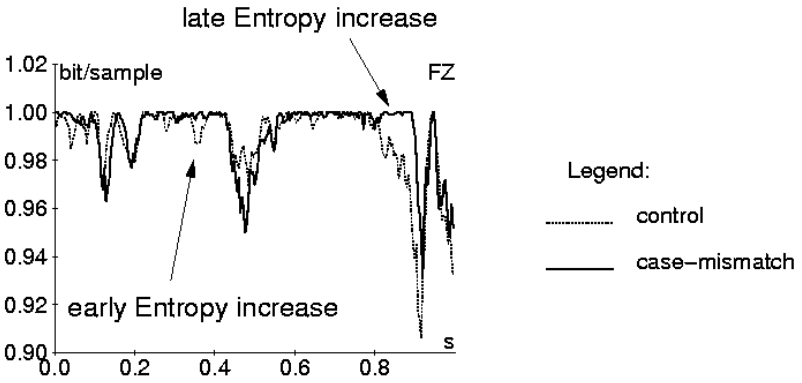


Figure 2: Running normalized Shannon entropies for the case-mismatch condition (solid lines) compared to the control condition (dotted) of the grand ensemble. Shannon entropy of 1–words ($n = 1$) in bit/sample against time (s), $t = 0$: stimulus onset time. The early and the late entropy increase are marked by arrows.

can be simulated by a Turing-machine. Hence, Moore's proof holds for stack automata too.[2]

Let us consider a context-free grammar $G = (N, T, S, P)$ where $N = \{S, N_1, N_2, \ldots, N_{n-1}\}$ is the set of n nonterminal symbols including the start symbol S, $T = \{t_1, t_2, \ldots, t_m\}$ is the set of m terminal symbols constituting the context-free language $\mathcal{L}(G) \subset T^*$ and $P \subset N \times (N \cup T)^*$ is a set of finite production rules $p_i : N_i \to \alpha_i$ with $\alpha_i \in (N \cup T)^*$. The grammar G is called *ambiguous* if there are at least two rules $p_i \neq p_j$ expanding the same nonterminal N_i. For the time being, we shall assume G to be nonambiguous. Then the language $\mathcal{L}(G)$ can be processed by a deterministic stack automaton. In the following we are going to establish a relation between simple deterministic *top-down parsers* and piecewise linear maps at the unit square. Such a parser possessing a single internal state q can be described by an *actual pair* (α, w) where $\alpha = \alpha_{i_0}\alpha_{i_1}\ldots\alpha_{i_{k-1}}$, $\alpha \in (N \cup T)^*$ is the content of the parser's stack while $w = w_{j_1}w_{j_2}\ldots w_{j_l}$, $w \in T^*$ is some finite word at the input tape (Aho and Ullman, 1972; Mayer, 1986). At each instance of time the automaton has only access to the top of the stack α_{i_0} and it "sees" the first symbol of the input tape w_{j_1}.

For the aim of our construction we shall reorder the content of the stack: $\alpha'_{i_{-k}} = \alpha_{i_k}$, obtaining the reversed sequence $\alpha' = \alpha'_{i_{-k+1}}\ldots\alpha'_{i_{-1}}\alpha'_{i_0}$. Next, we concatenate the transformed stack with the input band yielding a two-sided (but finite) string

$$s = \alpha'_{i_{-k+1}}\ldots\alpha'_{i_{-1}}\alpha'_{i_0}w_{j_1}w_{j_2}\ldots w_{j_l}\,. \tag{17}$$

Now, one could apply the g-adic expansion on the left- and right-half sequences s_- and s_+ after introducing a Gödel encoding of the set $V = N \cup T$ in order to map the actual pair to a point in the unit square. But this is not what we will do. To avoid gaps in the unit square we decided to use two different encodings: one of the set V for the stack and another of the terminals T for the input tape. Thus, we perform a $(n+m)$-adic expansion of the stack and a m-adic expansion for the input tape. This yields a partition of the unit square into rectangles of equal size $(n+m)^{-1} \times m^{-1}$ according to Eqs. (7, 8). The two g-adic expansions lead to a point (x, y) in the unit square with

[2]Moore has also proven that stack automata are equivalent to nondeterministic one-sided generalized shifts (Moore, 1991b,a), but we do not pursue this approach since we are interested in deterministic dynamical systems.

coordinates

$$x = \sum_{j=0}^{k-1} g(\alpha'_{i_{-j}})(m+n)^{-j-1} \tag{18}$$

$$y = \sum_{j=1}^{l} g(w_{i_j})m^{-j}\,. \tag{19}$$

By this construction the most significant digits α'_{i_0} and w_{i_1} determine the rectangle where the point (x, y) will be found. We obtained therefore a coarse grained description of the actual pairs represented by states of a quantitative dynamical system.

To proceed further, we need some consideration of the *empty word* ϵ. In formal language theory the set A^* of all strings of words of finite length formed by letters of the alphabet A constitutes a semi-group with respect to the concatenation "$\cdot$" of words: $u \cdot (v \cdot w) = (u \cdot v) \cdot w$ for all $u, v, w \in A^*$ and $w \cdot \epsilon = \epsilon \cdot w = w$. The empty word ϵ is the neutral element of this semi-group. In order to complete our construction we have to determine the image of ϵ in the unit interval when we consider one-sided sequences $s \in A^{\mathbb{N}}$. This can be done by using Eq. (6) which states the relation between cylinder sets of length n and their preimages in the phase space of a dynamical system. Decomposing the left hand side of Eq. (6) into concatenation and the right hand side into intersection factors leads to

$$\pi^{-1}([(a_{i_1}, \ldots, a_{i_m}) \cdot (a_{i_{m+1}}, \ldots, a_{i_n})]_1) = \left(\bigcap_{k=0}^{m-1} \Phi^{-k}(A_{i_{k+1}})\right) \cap \left(\bigcap_{k=m}^{n-1} \Phi^{-k}(A_{i_{k+1}})\right) \tag{20}$$

or in shorthand notation $\pi^{-1}(u \cdot v) = \pi^{-1}(u) \cap \pi^{-1}(v)$ for words u, v regarded as cylinder sets. Hence, π^{-1} is a semi-group homomorphism mapping the concatenation of words onto the intersection of sets. It is well known from set theory that the neutral element with respect to set intersection is a base set. In our case, this is just the unit interval $[0, 1]$ which has to be identified with the empty word. Correspondingly, the goal of the top-down parser, the actual pair (ϵ, ϵ) is the whole unit square. A further consequence of this reasoning is that the content of the stack as well as the content of the input tape might be considered as one-sided infinite strings that are committed to finite words at the very beginning. Due to this interpretation we allow uncertainty about forthcoming and already processed input respectively. Now, we are able to extend the symbol sequence from Eq. (17) towards plus and minus infinity:

$$s = \ldots \alpha'_{i_{-k+1}} \ldots \alpha'_{i_{-1}} \alpha'_{i_0} w_{j_1} w_{j_2} \ldots w_{j_l} \ldots\,. \tag{21}$$

The actual pair (α, w) of the parser comes out to be a set of all bi-infinite sequences s coinciding in the symbols $\alpha'_{i_{-k+1}} \ldots \alpha'_{i_{-1}} \alpha'_{i_0}$ at the left-half side and coinciding in the symbols $w_{j_1} w_{j_2} \ldots w_{j_l}$ at the right-half side, that is a cylinder set. Using the g-adic expansions (Eqs. 18, 19) of the parser's state, the actual pair will be mapped onto a rectangle in the unit square.

Our remaining job is establishing the parser's dynamics by a generalized shift and finally by a piecewise linear map at the unit square. Since the top down parser has only access to the most significant symbols of stack and input tape it can be seen as a map $\tau : (\alpha_{i_0}, w_{i_1}) \mapsto (\bar{\alpha}, w'_{i_1})$ where $\bar{\alpha}$ is a string consisting of terminals and nonterminals and w_{i_1} is the beginning of the input tape after the operation of the parser. For building this map we must take into account three different cases:

1. The first item on the stack is a nonterminal $\alpha_{i_0} = N_i$. The parser has to look into the list of productions whether it can find a rule expanding N_i into a sequence $\bar{\alpha}_i$. If there is no such production the parse ends in a nonaccepting state (N_i, w_{i_1}).

2. The first item on the stack is a terminal $\alpha_{i_0} = t_i$. The parser has to compare this symbol with the first item in the input tape w_{i_1}. If they are equal both symbols will be canceled from stack and from the input word (this operation is called *attach*). The parser reaches the next state (α_{i_1}, w_{i_2}). If they are not equal the parse ends in a nonaccepting state (α_{i_0}, w_{i_1}).

3. Stack and input contain the empty word (ϵ, ϵ). This is the only accepting state of the top-down parser.

It follows from these operations that the domain of definition of the parser's dynamics are the pairs of most significant symbols (α_{i_0}, w_{i_1}) corresponding to the rectangles constituting the partition of the unit square. The images of these rectangles are the same rectangles for the nonaccepting states (the map is constant at these domains). If the most significant digit of the stack is a nonterminal that can be expanded by some production, the image of the corresponding rectangle is a subset of that rectangle described by the most significant digit of the expanded string. Due to the expansion the image rectangle will be contracted with respect to the x-axis (denoting the stack). If the most significant digits of stack and input are terminals and if there are identical the rectangle will be extended in x- as well as in y-direction and it will be shifted in parallel to become a subset of that rectangle describing the most significant digits of stack and input after cancellation of the previous symbols. This construction will be explained in the next section by an example.

From these constrains we can construct the map assigning points (x, y) to points $(x', y') = \Phi((x, y))$ in the unit square. The map is defined to be piecewise linear at the rectangles provided by the partition describing the most significant digits of the actual pairs of the parser. At each rectangle the map assumes the form

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} \lambda_x & 0 \\ 0 & \lambda_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{22}$$

where $(a_x, a_y)^T$ is a local translation vector whereas λ_x, λ_y are local stretching $(\lambda > 1)$ and contraction $(\lambda < 1)$ factors, respectively. These coefficients might differ from rectangle to rectangle.

Since actual pairs of the parser will be mapped onto rectangles and since these are cylinder sets, we can employ the technique of running cylinder entropies developed for the analysis of event-related brain potentials (see subsection 3.3) also to parsing models. In order to do this we need a second partition of the unit square, called a *measurement partition* (Wackerbauer et al., 1994) for capturing the parsing rectangles. When the grid of this partition is rather course grained, small rectangles will slip through the meshes not contributing to the cylinder entropy. On the other hand, when the partition is fine grained it will probably cover larger rectangles yielding a high amount of entropy.

6 Modeling Language Processing

6.1 The Model

In this section we shall develop a model of the ERP experiment discussed in sections 2.2 and 4. We suggest an ambiguous context-free "toy" grammar[3] G consisting of the productions

$$S \rightarrow W v_{sg} N_{sg} \tag{23}$$
$$S \rightarrow W v_{pl} N_{pl} \tag{24}$$
$$N_{sg} \rightarrow d_{nom-sg} n_{nom-sg} \tag{25}$$
$$N_{sg} \rightarrow d_{acc-sg} n_{acc-sg} \tag{26}$$
$$N_{pl} \rightarrow d_{nom-pl} n_{nom-pl}\,. \tag{27}$$

In this grammar $S \in N$ denotes the start symbol, W denotes the ambiguous initial wh-phrase "welche Frau", v_{sg} denotes the singular verb "sah", v_{pl} denotes the plural verb "sahen", d_{nom-sg} denotes the nominative case marked singular article "der", n_{nom-sg} denotes the singular noun "Mann", d_{acc-sg} denotes the accusative case marked singular article "den", n_{acc-sg} denotes the singular noun "Mann", d_{nom-pl} denotes the nominative case marked plural article "die" and

[3]We do not claim that the human language processor is a context-free top-down parser, of course. The model subsequently discussed should be regarded as an academic exercise. However, due to Moore's proof also the human parser is representable by a piecewise linear map at the unit square provided it can be emulated by some Turing-machine.

n_{nom-pl} denotes the plural noun "Männer". These are the terminals. Finally, N_{sg} and N_{pl} stand for the nonterminals singular noun phrase and plural noun phrase, respectively.

The control condition of the ERP experiment is given by expanding the rules (23) and (26) yielding the string $Wv_{sg}d_{acc-sg}n_{acc-sg}$ ("welche Frau sah den Mann"). The number-mismatch condition results from expanding the rules (24) and (27): $Wv_{pl}d_{nom-sg}n_{nom-sg}$ ("welche Frau sahen die Männer") and the case-mismatch condition follows from the rules (23) and (25): $Wv_{sg}d_{nom-sg}n_{nom-sg}$ ("welche Frau sah der Mann").

For implementing the nonlinear parsers dynamics we need nonambiguous grammars. The grammar G will therefore be split into three nonambiguous parts: a grammar G_0 for the case-mismatch:

$$S \rightarrow Wv_{sg}N_{sg} \tag{28}$$
$$N_{sg} \rightarrow d_{nom-sg}n_{nom-sg}\,, \tag{29}$$

a grammar G_1 for the control condition, the default grammar:

$$S \rightarrow Wv_{sg}N_{sg} \tag{30}$$
$$N_{sg} \rightarrow d_{acc-sg}n_{acc-sg}\,, \tag{31}$$

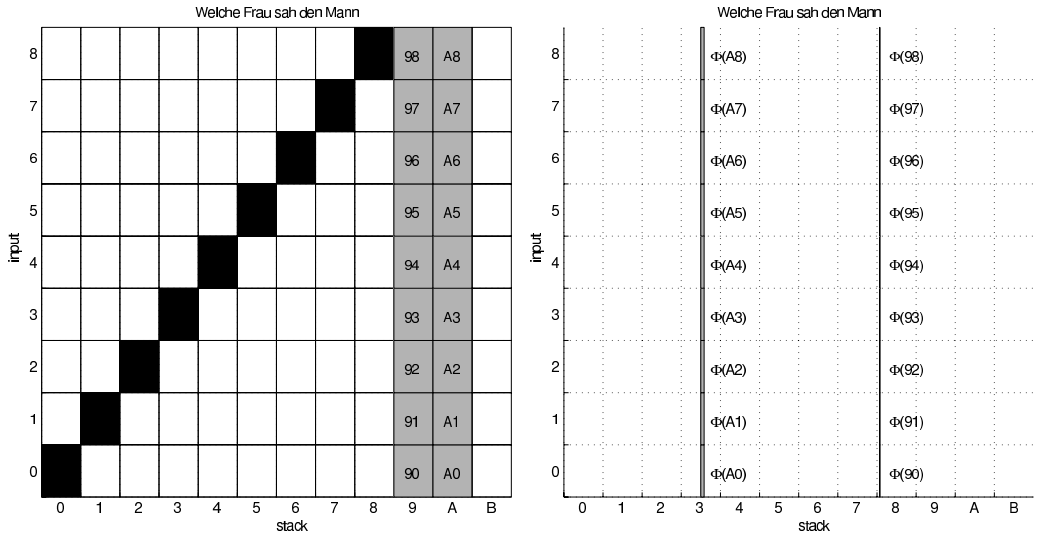
and eventually a grammar G_2 for the number-mismatch:

$$S \rightarrow Wv_{pl}N_{pl} \tag{32}$$
$$N_{pl} \rightarrow d_{nom-pl}n_{nom-pl}\,. \tag{33}$$

For each of these grammars we construct a top-down parser. The parser operating on G_1 is described by a map with stack alphabet $N(G_1) = \{S, N_{sg}\}$ and with terminal alphabet $T(G_1) = \{W, v_{pl}, d_{nom-pl}, n_{nom-pl}\}$. But for the sake of convenience we allow nonterminals and terminals of the original grammar G for all three parsers. This will result only in an additional number of nonaccepting states.

The next step is the Gödel encoding of N and T. We use the following arbitrary code: $v_{sg} = 0, v_{pl} = 1, d_{nom-sg} = 2, d_{acc-sg} = 3, d_{nom-pl} = 4, n_{nom-sg} = 5, n_{acc-sg} = 6, n_{nom-pl} = 7, W = 8, S = 9, N_{sg} = \mathrm{A}, \mathrm{N_{pl}} = \mathrm{B}$ taking a subset of hexadecimal digits. In this list the digits 0 – 9 encode terminal symbols and 9 – B (=12) encode nonterminals. Due to Eqs. (18, 19) we need a 12-adic number system for encoding words on the stack and a 9-adic system for encoding items in the input. We therefore partitionate the unit square into 108 rectangles that can be addressed by their most significant digits (α_{i_0}, w_{i_1}). The parser becomes initialized with the actual pair $(9, w)$ where w stands for an arbitrary input word. Let us consider the state $(9, 8)$ for example. The corresponding rectangle will be mapped by the first iteration to the rectangle $(80\mathrm{A}, 8)$ that is contained in the cell $(8, 8)$. Hence, the expansion of the start symbols results in a contraction of the rectangle along the x-axis and in a shift into the cell $(8, 8)$. The site and extension along the y-axis remains the same. Now let us consider the evolution of the rectangle $(8, 8)$ itself. By the attach operation this state will be mapped onto (ϵ, ϵ) i.e. the whole unit square. All rectangles lying in the quasi-diagonal (t, t) expand similarly: they will be stretched along the x- as well as along the y-axis and eventually shifted into the origin of the unit square. And for a last example let us look at the state $(0, 1)$. There is neither any rule nor the possibility to attach. The rectangle will thus be mapped onto itself. By this construction we obtain a piecewise linear map defined at the cells of the partition that is constant at 81 rectangles, fully expanding at 9 rectangles and partial contracting at 18 rectangles. Figures 3 (a) and (b) present the domains and images of this map.

After constructing three maps describing the top-down parsers operating with the nonambiguous grammars G_0, G_1, and G_2 we merge these maps into a single dynamical system by introducing a monotonic function of the grammar's index as control parameter. Let $r = i/2$ be the value of the control parameter dependent on the grammar index i. Then, $\Phi_r : [0, 1]^2 \rightarrow [0, 1]^2$ describes a *bifurcating* dynamical system living at the unit square. For the value $r = 0.5$ the system behaves as the parser of the default grammar G_1. The values $r = 0$ and $r = 1$ correspond to the case-mismatch and the number-mismatch settings, respectively.



(a) (b)

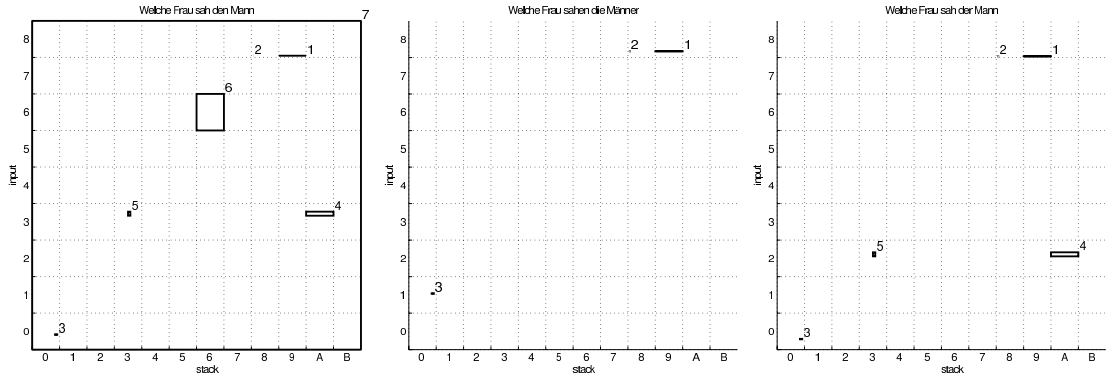
Figure 3: Domains and images of the piecewise linear map describing the top-down parser of the default grammar G_1. (a) Domains are the rectangles given by the partition of the unit square. White domains: constant map, black domains: the map is expanding to the full unit square (attach), gray domains: the map is contracting along x. Labels: pairs of most significant digits (xy). (b) Images of the contracting domains (gray), labeled by $\Phi(xy)$.

In order to make the system continuously dependent on the control parameter r we designed a feed-forward neural network consisting of three input units: two units encoding a point in the unit square and one unit providing the control parameter, 100 hidden units, and two output units encoding the image of the input point. We generated $N = 1404$ randomly distributed points in the unit square and determine their images using the maps of the parsers assigned to the grammars G_0, G_1, and G_2. The network has been trained with these pairs of points together with the three values of the control parameter using a learning rate of 0.005.

6.2 The Simulations

In this last subsection we shall present the results of our simulations. We determined the rectangles corresponding to the initial states of the parser $(9, w_i)$ where w_i are the input words $w_0 = 8025$, $w_1 = 8036$, and $w_2 = 8147$ accepted by the grammars G_0, G_1, and G_2. Then, we created $N = 1000$ points scattered randomly within these rectangles as initial conditions of the dynamics. For each value $r = 0, 0.5, 1$ of the control parameter we computed the iterates of the maps using the trained network. The support of the cloud of points in the unit square provides a new rectangle at each iterate; these rectangles are plotted in Figure 4. It shows the correct parse of the string $w_1 = 8036$ and the two incorrect parses ("garden paths") of $w_0 = 8025$ and $w_2 = 8147$ with the control parameter setting of the default grammar G_1 as trajectories of rectangles through the unit square. Because only the string $w_1 = 8036$ is accepted by the parser associated to grammar G_1 its trajectory ends in the unit square. The strings w_0, w_2 are attracted by fixpoints in the domains of the constant map.

The garden path readings of the number-mismatch and the case-mismatch condition are obtained by a "wrong" setting of the system's control parameter. When we compute the trajectories starting at the initial conditions w_0, w_2 with the right settings $r = 0$ and $r = 1$, respectively, we gain correct parses ending in the accepting state (ϵ, ϵ). Thus, we have proven that a nonlinear



(a) (b) (c)

Figure 4: Trajectories of rectangles obtained by iterating the initial conditions w_1, w_2 and w_3 through the unit square according to the default grammar G_1. (a) correct parse, (b) number-mismatch condition, (c) case-mismatch condition. The numbers indicate the iterates. The last rectangle is a stable fixpoint attractor of the dynamics.

dynamical system dependent on a control parameter is able to process ambiguous grammars deterministically according to the setting of its control parameter. Local ungrammaticalities and garden path interpretations will occur when the parameter is not tuned appropriately. This is only possible if the system bifurcates between two parameter settings, that is, there is a phase transition in the dynamics of the system. An detailed discussion of the bifurcation behaviour of the system and the issues of reanalysis and repair will be addressed in beim Graben et al. (2000a).

For illustrating this claim, we finally present the cylinder entropies (Eq. 15) of the parsing dynamics obtained by a further, appropriately chosen, partition of the unit square. Figure 5 shows the time course of the cylinder entropies of the correct parse compared to the number-mismatch condition. The measurement partition has been chosen to show an entropy drop when the parser encounters the verb.

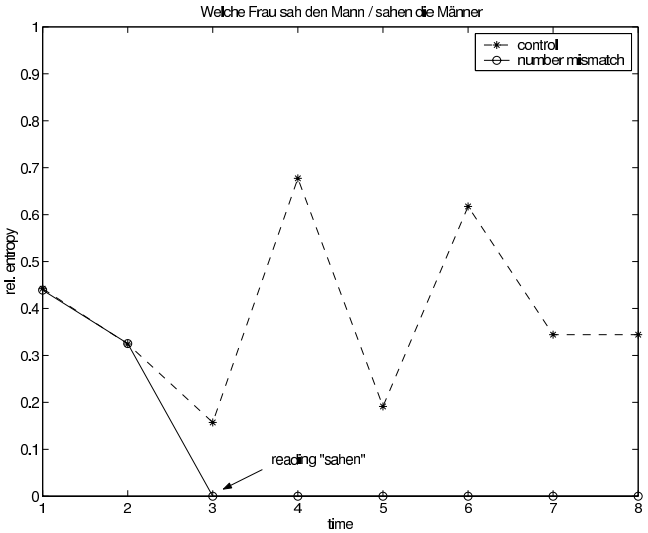


Figure 5: Shannon cylinder entropies of the simulated parses. Solid: number-mismatch condition, dashed: control condition.

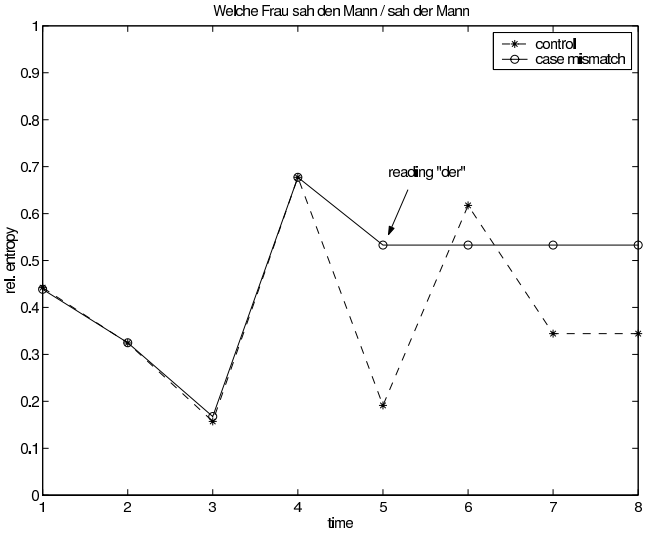


Figure 6: Shannon cylinder entropies of the simulated parses. Solid: case-mismatch condition, dashed: control condition.

Figure 6 demonstrates the cylinder entropies of the simulated parses of the correct parse compared with the case-mismatch condition. Here, there is an entropy increase in the case-condition when the parser fetches the second article. Using this measurement partition we can recapitulate the findings of the ERP experiment (see Figs. 1, 2).

7 Acknowledgements

We gratefully acknowledge an inspiring seminar on deterministic parsing of Peter Staudacher. This work has been supported by the Deutsche Forschungsgemeinschaft within the scientists group on conflicting rules in cognitive systems.

References

Aho, A. V. and Ullman, J. D. (1972). *The Theory of Parsing, Translation and Compiling*, volume I: Parsing. Prentice Hall, Englewood Cliffs (NJ).

Başar, E. (1983). EEG and synergetics of neural populations. In Başar, E., Flohr, H., Haken, H., and Mandell, A. J., editors, *Synergetics of the Brain*, volume 23 of *Springer Series in Synergetics*, pp. 183 – 200, Berlin. Springer.

Badii, R. and Politi, A. (1997). *Complexity. Hierarchical Structures and Scaling in Physics*, volume 6 of *Cambridge Nonlinear Science Series*. Cambridge University Press, Cambridge (UK).

Beck, C. and Schlögl, F. (1993). *Thermodynamics of Chaotic Systems. An Introduction*, volume 4 of *Cambridge Nonlinear Science Series*. Cambridge University Press, Cambridge (UK). Reprinted 1997.

beim Graben, P., Liebscher, T., and Saddy, J. D. (2000a). Language processing by nonlinear dynamical systems: Bifurcations, reanalysis and repair. In preparation.

beim Graben, P., Saddy, J. D., Schlesewsky, M., and Kurths, J. (2000b). Symbolic dynamics of event–related brain potentials. *Phys. Rev. E*, 62(4):5518-5541.

Buchner, T. and Zebrowski, J. J. (1999). Symbolic dynamics analysis of nonstationary data from a model of a magnetic system with solitons. *Phys. Rev. E*, 60(4):3973 – 3981.

Collet, P. and Eckmann, J.-P. (1980). *Iterated Maps on the Interval as Dynamical Systems*. Birkhauser, Boston.

Elman, J. L. (1995). Language as a dynamical system. In Port and van Gelder (1995), pp. 195 – 223.

Engbert, R., Scheffczyk, C., Krampe, R. T., Rosenblum, M., Kurths, J., and Kliegl, R. (1997). Tempo–induced transitions in polyrhythmic hand movements. *Phys. Rev. E*, 56(5):5823 – 5833.

Friederici, A. D. (1999). The neurobiology of language comprehension. In Friederici, A. D., editor, *Language Comprehension: A Biological Perspective*, pp. 265 – 304. Springer, Berlin, 2nd edition.

Haken, H. (1996). *Principles of Brain Functioning*, volume 67 of *Springer Series in Synergetics*. Springer, Berlin.

Hopcroft, J. E. and Ullmann, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison–Wesley, Menlo Park, California.

Jirsa, V. K. and Haken, H. (1996). Field theory of electromagnetic brain activity. *Phys. Rev. Lett.*, 77(5):960 – 963.

Kantz, H., Kurths, J., and Mayer-Kress, G., editors (1998). *Nonlinear Analysis of Physiological Data*. Springer, Berlin.

Kelso, J. A. S., Bressler, S. L., Buchanan, S., DeGuzman, G. C., Ding, M., Fuchs, A., and Holroyd, T. (1992). A phase transition in human brain and behavior. *Phys. Lett. A*, 196:134 – 144.

Kelso, J. A. S., Scholz, J. P., and Schöner, G. (1986). Nonequilibrium phase transitions in coordinated biological motion: critical fluctuations. *Phys. Lett. A*, 118(6):276 – 284.

Kurths, J., Voss, A., Witt, A., Saparin, P., Kleiner, H. J., and Wessel, N. (1995). Quantitative analysis of heart rate variability. *Chaos*, 5:88 – 94.

Kutas, M. and van Petten, C. K. (1994). Psycholinguistics electrified. event–related brain potential investigations. In Gernsbacher, M. A., editor, *Handbook of Psycholinguistics*, pp. 83 – 133. Academic Press, San Diego.

Mayer, O. (1986). *Syntaxanalyse*. Reihe Informatik. Bibliographisches Institut, Zürich, 3rd edition.

McMillan, B. (1953). The basic theorems of information theory. *Ann. Math. Statist.*, 24:196 – 219.

Moore, C. (1990). Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.*, 64(20):2354 – 2357.

Moore, C. (1991a). Generalized one-sided shifts and maps of the interval. *Nonlinearity*, 4:727 – 745.

Moore, C. (1991b). Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4:199 – 230.

Politi, A. and Badii, R. (1997). Dynamical 'strangeness' at the edge of chaos. *J. Phys. A*, 30:L627 – L633.

Port, R. F. and van Gelder, T., editors (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge (MA).

Raçzasek, J., Tuller, B., Shapiro, L. P., Case, P., and Kelso, S. (1999). Categorization of ambiguous sentences as a function of a changing prosodic parameter: A dynamical approach. *Journal of Psycholinguistic Research*, 28(4):367 – 393.

Rapp, P. E., Zimmerman, I. D., Vining, E. P., Cohen, N., Albano, A. M., and Jiménez-Montano, M. A. (1994). The algorithmic complexity of neural spike trains increases during focal seizures. *The Journal of Neuroscience*, 14(8):4731 – 4739.

Saddy, J. D., beim Graben, P., and Schlesewsky, M. (1999). Cortical dynamics of language processes. In Baragni, S., editor, *Proc. European Conference on Cognitive Science (ECCS)*, pp. 323 – 332, Siena (Italy). University Siena, Multimedia & Communication Laboratory, University Siena.

Scheffczyk, C., Zaikin, A., Rosenblum, M., Engbert, R., Krampe, R., and Kurths, J. (1997). Modeling qualitative changes in bimanual movements. *Int. J. Bifurcation Chaos*, 7:1441 – 1450.

Schiek, M., Drepper, F. R., Engbert, R., Abel, H. H., and Suder, K. (1998). Cardiorespiratory synchronization. In Kantz et al. (1998), pp. 191 – 213.

Schuster, H. G. (1989). *Deterministic Chaos*. VCH, Weinheim.

Schwarz, U., Benz, A. O., Kurths, J., and Witt, A. (1993). Analysis of solar spike events by means of symbolic dynamics methods. *Astronomy and Astrophysics*, 277:215 – 224.

Shannon, C. E. and Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana. Reprint 1963.

Wackerbauer, R., Witt, A., Atmanspacher, H., Kurths, J., and Scheingraber, H. (1994). A comparative classification of complexity measures. *Chaos, Solitons & Fractals*, 4(1):133 – 173.

Simulation of Emotions of Agents in Virtual Environments using Neural Networks

Aard-Jan van Kesteren, Rieks op den Akker,
Mannes Poel and Anton Nijholt

Parlevink Research Group,
University of Twente,
P.O. Box 217, 7500 AE Enschede
the Netherlands
{kesteren|infrieks|mpoel|anijholt}@cs.utwente.nl

Abstract

A distributed architecture for a system simulating the emotional state of an agent acting in a virtual environment is presented. The system is an implementation of an event-appraisal model of emotional behaviour and uses neural networks to learn how the emotional state should be influenced by the occurrence of environmental and internal stimuli. A part of the modular system is domain-independent. The system can easily be adapted for handling different events that influence the emotional state. A first prototype and a testbed for this architecture are presented.

1 INTRODUCTION

This paper presents results of ongoing research on human-computer interaction with intelligent conversational agents in virtual environments. It reports work in a project aiming at the design and agent-oriented implementation of a multi-user, multi-agent and multi-modal interactive environment [Nijholt 1999].
Our interest in emotions and our objective to simulate emotional behaviour finds its primary motivation in the hypothesis that avatars that somehow show emotions in the way they behave are more *believable* than agents that lack these human qualities.
There is another motive for incorporating emotions in the design of synthetic agents; often mentioned by AI-researchers. A. Damasio came with neurological evidence for the importance of emotions in human decision processes [Damasio 1994]. The interaction between emotions and the rational processes that underlie the making of decisions could possibly explain why humans are so good in making decisions in a context of uncertainty. Answering the question whether machines can have emotions, M. Minsky even stated*: "The question is not whether intelligent machines can have any emotions, but whether machines can be intelligent without any emotions"* [Minsky 1986]. Hence the idea to give synthetic agents emotions in order to make it possible that they perform whatever tasks they have in a more intelligent way. Whether there is some sense in this depends on the notion of emotion one has in mind and on the analysis of the aspects involved in the process of decision-making.

To model and to simulate all of the relevant aspects of emotions in a comprehensive system is a project for years, so it shouldn't come as a surprise that every group that's conducting research on the topic of emotional agents has made it's own decisions as to what aspects are built into their system, and what is left out for (hopefully) future development. Most groups [Reilly & Bates 1992, El-Nasr & Yen 1988, Velásquez 1997] have taken a wide range of topics related to emotions (e.g. emotion, facial expressions and emotional behaviour) into account for their emotional systems. Until now we have only looked at the topic "*simulation of emotions"*, as we think it's best to focus on one topic at a time.
We think of an emotional agent as an agent that has among other things (e.g. believes, desires and intentions) an *emotional state*. This emotional state can be altered by *stimuli* from the environment or by *stimuli* from internal elements of an agent (e.g. decisions, future expectations, memory, etc.). "Simulation of emotions" consists of the elements that *appraise* the stimuli and the processes that take care of the *dynamics* of the emotional state. In principle, every kind of behaviour and internal process of an agent can make use of the emotional state. For instance, the emotional state can be used to steer the facial expressions and the decision process. Consequently, our agents don't show emotions yet. A user can only get an impression of what the agent "feels", by looking at a representation of the emotional state.

For our model of emotions, we have two basic concerns. We want our model to resemble the emotional processes, as they exist within the human brain, as closely as possible and consequently we don't want to diverge too much from the leading theories of emotions. On the other hand we also want to develop a computational approach, which can be easily used to implement an emotional agent.

Our research is very much ongoing and some of the elements presented in this paper haven't been properly tested yet. This paper will give an overview of how we dealt with the problem of emotions until now and what our approach for the future will be. Furthermore an architecture for the simulation of emotions will be introduced.
First the relevant models of emotions will be discussed shortly. After that we will globally describe what our approach is, followed by a description of the environment that we used as a testbed. Subsequently our *distributed architecture* will be introduced, by firstly explaining the architecture as a whole and secondly by discussing the various subparts in more detail. A description of a first prototype will be given next to further clarify the architecture and to present some first test results. The paper will be concluded by a comparison with another model and some conclusive remarks.

2 THEORIES OF EMOTION

In this section, two theories on emotion will be introduced. How we used these theories for our model will be explained in the next section.

Event appraisal models[1] have as assumption, that it's the *subjective, cognitive interpretation of events* that explains the experienced emotions.
Event appraisal models mainly focus on the question how events are appraised and in which direction (and with what velocity) the emotional state is likely to shift. We think of this as the *emotional meaning* of an event. Usually event appraisal models don't model the *dynamics of the emotional state*: if an event is appraised, what will then become the new emotional state? For an AI-model of emotion this question is as important as a natural appraisal of events.

One of the leading event appraisal models is the model of Ortony, Clore and Collins (The OCC Model) [Ortony, Clore & Collins 1988]. The model only looks at *emotion types*. Each type contains a large number of emotional states, which may differ in intensity (e.g. the emotional states: worried, scared and terrified belong to the emotion type fear), but may also differ because of other reasons, such as the cause of the emotion (e.g. the emotional state heartache differs from other emotional states belonging to the emotion type distress). By focusing on emotion types instead of emotions, the whole field of emotions gets more comprehensible.
In order to appraise a particular event, there are three different aspects that can be focused on. Every aspect has a different set of emotion types attached to it. We will elaborate a bit on the different aspects by means of an example. Imagine that someone hears that his sister-in-law has just killed one of his children. If that someone focuses on the *consequences of the event*, he'll probably experience distress. If he focuses on the *action of the agent* (the sister-in-law), he'll probably experience reproach. And if he looks at the *aspects of an object* (aspects of the sister-in-law) he'll probably experience hate.
Associated with each aspect is a different kind of knowledge representation. If one focuses on the consequences of an event, *goals* are of importance. If one focuses on the action of an agent, *standards* are of importance and if one focuses on the action of an agent, *attitudes* are relevant.

A strong point of the OCC model, which makes it particularly useful for designing an AI-model of emotion, is that the model includes a complete set of variables that influence the intensity of the emotions. The variables can be global, which means that the variable influences all of the emotion types, or local, which means that the variable influences only some of the emotion types. We will shortly introduce the (in our opinion) three most important local variables. The variable *desirability* is important if one focuses on the consequences of an event. The variable *praiseworthiness* is important if one focuses on the actions of an agent and the variable *appealingness* is important if one focuses on the aspects of an object.

Besides the event appraisal models, there are more theories of emotions we found interesting and useful. One of those is Frijda's notion of emotion [Frijda 1986]. According to him, emotion is *action readiness change*, which refers to the inner disposition (and the absence) for performing actions and achieving relational change, as it exists at a particular moment. The experience of emotion largely consists of experienced action readiness or unreadiness: an impulse to flee, strike or embrace; or lack of impulse, apathy, listlessness. So, emotion is not the same as the feeling of emotion. A consequence of this is, that an explicit representation of emotions is unlikely to exist.

[1] For instance [Ortony, Clore & Collins 1988] and [Roseman, Jose & Spindel 1990]

3 OUR APPROACH

Our research goal is to develop a human like agent that shows natural emotional behaviour. Yet, as we didn't want to start too ambitiously we are firstly focusing on simple agents in a simulated environment. The agents and the environment shall be introduced in the next section.

Our model is very much an implementation of the more theoretical OCC model. First of all, only *events* can alter the emotional state. Furthermore, the representation of emotion consists of the same *emotion types* the OCC model distinguishes, the three *aspects* of events also play a central role in our model and the same *variables* (the properties of events that are important for emotions) as in the OCC model will be used. Furthermore, the agents will contain a representation of *goals*, *standards* and *attitudes*.

The current state of research on emotions is, that a lot of work has been done on philosophical questions like: "What are emotions?", "What influences emotions?", "Which emotions can be distinguished?" and "What is the connection between cognition and emotion?", but no complete quantitative models of emotions (besides some AI-models) have been developed yet. Apparently it's very difficult to describe the emotional behaviour in a complete and precise manner. On the other hand we noticed that it's relatively easy to imagine in practical situations what would be natural behaviour.
Therefore we chose an approach in which the system has to *learn* about emotional processes and rules from examples provided by a trainer. Most other research groups have used a completely opposite approach. In their systems the designer has to define the emotional processes and rules, from which natural emotional behaviour has to *originate*. We'll use the common terms *top-down approach* for our approach and *bottom-up approach* for the other approach (Figure 1 gives a schematic representation of the two concepts).

For our approach trainingsdata is needed, which will be obtained through annotation. We realize that annotation is a time-consuming and tedious task and therefore we defined the architecture in such a way, that a minimal amount of trainingsdata is required.
Another advantage of a top-down approach above a bottom-up approach is, that it's more flexible; the same system can be used for more applications, which is for instance practical if one wants to design agents with different personalities. Using a different trainingsset for each personality is an easy way to do this. Of course the same effect may be obtained in a bottom-up approach by changing some of the parameters, but we strongly doubt if parameters can be defined in such a way that it offers the same flexibility as a top-down approach does.

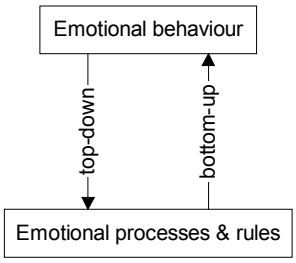


Figure 1 A top-down approach vs. a bottom up approach

As we want to design a system that only needs a small amount of trainingsdata, it's especially important that the system is able to generalize well. This is one of the proven qualities of *neural networks*, and accordingly we will use those for our system. For the emotion domain, it isn't very important that the system works very accurately in a quantitative way (the chosen numerical representation of emotions is a rough estimate anyway), as long as it performs well in a qualitative way and reasonably well in a quantitative way. Therefore relatively small neural networks can be used, which should enhance the generalization capacities of the system even more.
As Frijda's theory of emotions clearly suggests, an explicit representation of emotions is unlikely to exist within the human brain. We do not only think that an explicit representation of emotions is unlikely from a cognitive point of view, but we also think that a wrongly chosen representation might have a strong negative effect on the capacities of the system to show a wide range of emotional phenomena. By using *recurrent neural networks*, the system can learn an optimal, implicit representation of emotions itself, which we see as a major advantage of a connectionist approach.

3.1 THE ENVIRONMENT AND THE AGENTS

In order to develop and test a system that simulates emotions, an environment is needed, which is inhabited by agents that can have emotions. As we want to model a broad range of emotions, the environment and the agents must satisfy a number of requirements.

As we make use of the OCC model as basis for our AI-model of emotion, it's important that the agents have an explicit or implicit representation of *goals*, *standards* and *attitudes*[1]. Agents should also be able to translate observations into terms of these three concepts. In practice this last requirement has the following consequences:

- Agents should be able to see if an event satisfies a particular goal or has a positive or negative effect on the probability that a particular goal will be satisfied (important for the emotion types *joy* and *distress*).
- Agents must be able to reason about the *future* and should be able to change their *expectations* of the future as a consequence of events. Expectations about goals are important for emotion types such as *hope* and *fear*.
- Agents must have some kind of memory about previous expectations and should be able to compare new events to these previous expectations (important for the emotion types *relief*, *fears-confirmed*, *satisfaction* and *disappointment*).
- The same kind of reasoning the agents must be able to do for themselves, they must be able to do for other agents also (important for emotion types *happy-for*, *resentment*, *gloating* and *pity*).
- Agents must be able to compare actions of themselves or of other agents to their standards (important for emotion types *pride*, *shame*, *admiration* and *reproach*).
- Finally, agents must be able to compare aspects of objects or agents to their attitudes (important for emotion types *love* and *hate*).

The future must be partly *uncertain* for the agents. If it's precisely known what's going to happen in the future, emotions such as hope and fear cannot exist. A way to guarantee this is by making the environment *nondeterministic*. Another important property of the environment is, that it contains *multiple agents*, as some emotions depend on actions of other agents or on the consequences of events for other agents.
As the user has to annotate natural emotional behaviour, it's important that the user is able to put himself in the position of an agent. A user can only do this, if the behaviour of the agent is *believable* enough and if the user has the exact same knowledge as the agent. A final requirement of the environment is, that the situations are *complex* enough, such that interesting emotional behaviour can arise.
We set out to define and implement an environment, which satisfies all the previously stated constraints. For the domain we were inspired by [Inoue, Kawabata & Kobayashi 1996]. In Figure 2 a picture of the domain can be seen.

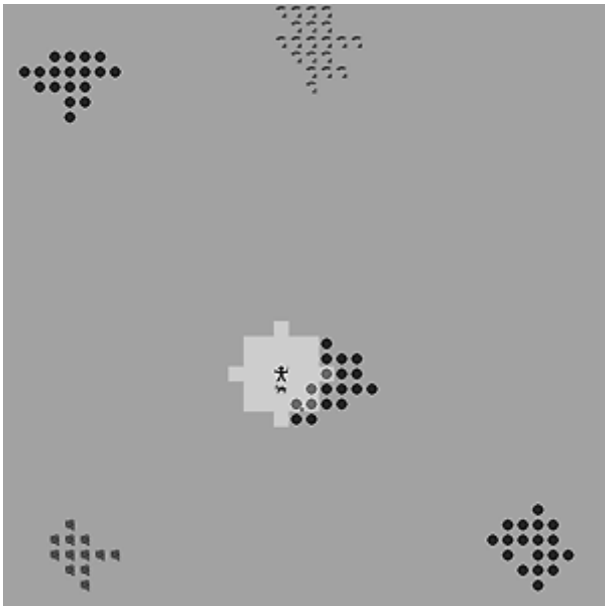


Figure 2 Picture of the environment

[1] We refer to [Reilly & Bates 1992] for a good example of an implementation of these concepts.

The domain is a gridworld containing grass, water pools that can be dry or contain water, apple trees that can have apples growing and rocks, with possibly herbs growing on it. The status of the trees, water pools and rocks constantly changes in a nondeterministic way.
Multiple agents inhabit the gridworld. One of the agents can be seen in the middle of the picture. An agent can only see a small part of the world (the light part). Currently the agent sees one predator (directly beneath the agent) and one tree with an apple (directly south-east of the predator). An agent only knows where the visible agents and predators are and he knows the location of all the trees, rocks and water pools, but the status is only known of the visible trees, rocks and water pools.

A trainer can only see the things an agent can see; in the light part, a trainer can see everything, just like an agent, and in the dark part, a trainer sees only the locations of the trees, rocks and water pools, as this is also knowledge that an agent possesses. Because of this, it's easier for a trainer to imagine what an agent feels. A trainer also knows how hungry, thirsty and healthy an agent is, which events have occurred lately and what his current action is.

Agents need food and water, which can be supplied by apple trees and water pools. There are predators that can be dangerous for the agents. An attack by a predator affects the health of an agent. An agent can regain health by eating an herb.
To a large extent, the behaviour of a predator is random. The only exception is when a predator is in the neighbourhood of food, water or an herb. Then it will stay there because it knows that agents will come there sooner or later. This means that the presence of a predator is an indication that food, water or an herb may be nearby. On the other hand the presence of food, water or an herb is an indication that a predator may be nearby.
Agents are able to make decisions and have all the previously described capacities. There is a social order between agents and they can choose (dependent on their character) either to follow the leader of a group or to go their own way. Social grouping is a result of common concerns. An agent knows how thirsty, hungry and healthy the agents in his group are.

4 THE SYSTEM

4.1 THE ARCHITECTURE

In this section, the architecture for our system will be introduced. First a global overview of the system will be given and after that the various concepts and subparts will be discussed in more detail. For the system we have chosen a *distributed architecture*, as depicted in Figure 3. The motivations for this particular architecture will be given after a brief introduction.

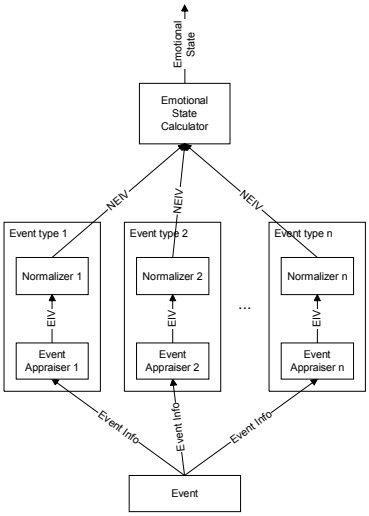


Figure 3 Schematic representation of the architecture

From a functional perspective, the task of the system is to convert an event with particular properties into a new emotional state. In order to convert an event to a new emotional state, two phases are distinguished.
The first phase is to appraise *the emotional meaning* of the event. This is the task of an *Event Appraiser*. Each event belongs to *maximally one* event type (e.g. the event: An agent sees an apple at a particular location while he has a particular desire for food belongs to the "Apple_spotted" event type). For each relevant event type, one event appraiser is defined. The content of the *event information* depends on the type of the event and can vary greatly from one event type to another. To sum up, when an event occurs, first the event type of the event has to be established. Then it's exactly known which Event Appraiser has to be used and what kind of information is needed for appraisal.
In the second phase, the *Emotional State Calculator* (the ESC) uses a numerical representation of the emotional meaning of the event (firstly stored in an *EIV* and secondly stored in an NEIV; both will be explained in detail later on) to calculate the new emotional state. A history of previous emotional states and emotional events is *implicitly* stored within the ESC, which is implemented by a recurrent neural network. The *Normalizers* form an interface between the Event Appraisers and the ESC. The task of a Normalizer is to normalize the data in an EIV, such that the ESC can treat all of the Event Appraiser in a *uniform* way. The Normalizers will also be implemented by neural networks.
This approach has a few qualities:

- **Conformity with emotion theory**. As mentioned before, emotion theorists are inclined to see the appraisal of events as a separate problem. In this architecture, the appraisal of events is explicitly a subpart of the system.
- **Conformity with human intuition**. In our first annotation attempts, we noticed that if we had to predict a new emotional state, given an event and the old emotional state, that we were inclined to first look only at the event and imagine what kind of emotional consequences this event might have and only after that we looked at the old emotional state and predicted a new emotional state. Therefore the way the system works conforms at least to our intuition and maybe also to the intuition of most people. This is an important property, as this means that the annotation task (which will be explained in the section on training) will be relatively easy.
- **Trainability**. The different subparts can be kept relatively small, as a result of which the various neural networks can be trained more easily and with a smaller amount of trainingsdata.
- **Possibility of incremental development**. This approach allows that events are added to the system one-by-one. This is a big advantage from a software engineering point of view.
- **Scalability**. If the domain gets more complicated (meaning more types of events), the amount of neural networks within the system will increase, but the complexity of the neural networks will remain the same. This means that the scalability of the system shouldn't be a problem, if the amount of types of events increases. Whether the system is scalable in other dimensions also (e.g. the amount of modelled emotions) is one of the open research questions remaining.
- **Reusability of the ESC**. The ESC is domain-independent and therefore can be reused for different agents in different domains.

4.1.1 EMOTION IMPULSE VECTORS

An *Emotion Impulse Vector* (an EIV) is a data structure especially suited for storing the *emotional meaning* of an event. The structure of an EIV has been chosen as follows:

$$EIV = <ei_1, ei_2, ..., ei_n>$$

in which every emotion type e_i has a corresponding *emotion impulse* ei_i. The emotion impulse ei_i indicates how the emotion e_i is likely to change. For instance a high positive value for ei_i means that the EIV probably will have a large positive effect on the emotion e_i.
A *Normalized Emotion Impulse Vector* (an NEIV) is a normalized version of an EIV. How the normalization functions, will be explained later on.

4.1.2 EVENT APPRAISAL

The task of an Event Appraiser is to assess the emotional meaning of an event to construct an EIV. Every event and domain has its own properties and therefore an Event Appraiser should be constructed separately for every event type. This way *a priori knowledge* about the event and domain can be used in an optimal way. A consequence of this is, that an Event Appraiser can be implemented by a neural network, but can just as well be implemented by a rule-based system, a function or something else.

Defining or training an Event Appraiser that can immediately communicate with the ESC in such a way that the behaviour of the total system is precisely as required, would mean a full understanding of the ESC and would at least

make defining or training the Event Appraiser a very difficult task. Therefore we introduced the Normalizers. Their task is to convert an EIV to an NEIV such that the behaviour of the total system works as required. As the Normalizers only have to do scaling operations, and therefore only have to learn linear or near linear functions, the Normalizers can be simple feedforward networks with only a few hidden neurons.
A nice property of using Normalizers is that the Normalizers can easily correct quantitative errors of the Event Appraisers and the ESC. Therefore it's not very important that the Event Appraisers and the ESC work well in a quantitative way, just as long as they work well in a qualitative way.

The inputs of an Event Appraiser are the variables defined in the OCC model or properties of an event that can be used to calculate the variables. The current *action* of the agent is a special kind of property, which can be important to appraise an event. For instance, an agent is bound to appraise the event of spotting a predator differently if he's attacking than if he's fleeing.

4.1.3 THE EMOTIONAL STATE CALCULATOR

The task of ESC is to calculate a new emotional state given a NEIV. Although the ESC is one neural network, conceptually it can be seen as two different parts. The first part, the core of the ESC, is a recurrent network and models all of the emotional behaviour. The state of the recurrent network can be seen as an *implicit emotional state*. We will implement the implicit emotional state with an Elman network [Elman 1990].
The second part of the ESC is a feed forward network, which has the implicit emotional state as input. This part, the Emotional state monitor, has as task to translate the implicit emotional state, which is very hard to understand and use to influence the behaviour, into an *explicit emotional state*. The structure of the explicit emotional state has been chosen as follows:

Explicit emotional state = $<e_1, e_2, \ldots, e_n>$

in which e_i denotes the intensity of an emotion type. Every e_i refers to a different emotion type.
A schematic representation of the ESC is depicted in Figure 4.

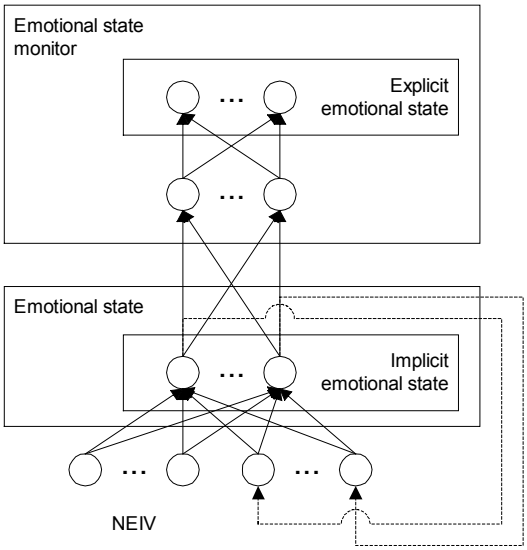


Figure 4 Schematic representation of the Emotional State Calculator. A dashed line represents a time delay of 1.

4.1.4 DECAY

The *decay* of an emotional state is a topic that has received special attention in most other AI-models of emotion and is also a topic mentioned in [Ortony, Clore & Collins 1988] as being a specific point of concern for a computational model of emotion. In our system the decay is modeled as an event type with a corresponding Event Appraiser and Normalizer. An event of this type occurs by definition after a fixed interval of time and sees to it that the emotional state eventually returns to a state of rest. A property that determines the appraisal of this event is among other thing the current (explicit or implicit) emotional state.

4.2 TRAINING THE SYSTEM

The system is built out of three types of building blocks. For every type of building block a different training strategy has to be used. The order in which the building blocks have to be trained or defined is depicted in Figure 5.

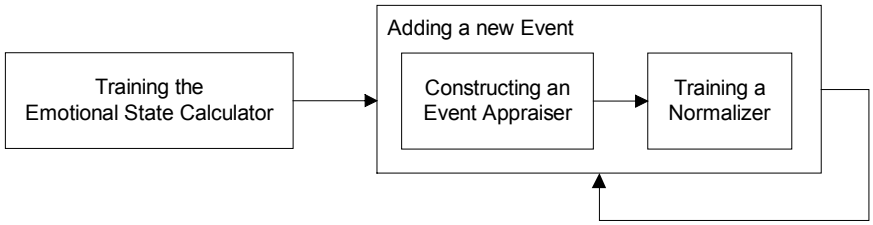


Figure 5 Schematic overview of the trainingsprocess

First the Emotional State Calculator has to be trained. This is the most difficult part of the system to train, as gathering trainingsdata is difficult. A *temporal sequence* of (NEIV, explicit emotional state) pairs is needed to train the Elman network. The NEIVs can be generated randomly, but the emotional states have to be annotated by hand.
There are two problems with this approach. First of all a reasonable amount of trainingsdata is needed and secondly the needed data is of an abstract nature, as a result of which annotating even one sample isn't an easy task. But we feel that the benefits of this tedious work are large enough to be profitable. If the Emotional State Calculator has been trained, adding events to the system is easy and an Emotional State Calculator is domain-independent and therefore only has to be trained once. The trainingsset doesn't have to be perfect, as the learning algorithm should be able to filter out the errors. Therefore the trainer can annotate quickly without worrying too much about errors, which makes the annotation task a bit easier. Also, it shouldn't be overlooked that a significant part of the difficulties of training the ESC are inherent to the complexity of the problem of emotions. A nice property of our system is that a major part of the complexity of emotions is modelled within a domain-independent part of the architecture, so that the same problems don't have to be solved over and over again for every domain and agent.

After the ESC works satisfactory, event types can be added one-by-one to the system. For a new event type, an Event Appraiser has to be trained or defined firstly. Again it's only important that an Event Appraiser works well in a qualitative way, which makes defining or training an Event Appraiser relatively easy.

From a procedural perspective, training a Normalizer is a bit less straightforward than training the ESC or an Event Appraiser. (EIV, NEIV) pairs are needed to train a Normalizer. The user cannot be expected to annotate NEIVs, because this would mean among other things that the user has to know exactly how the ESC works, which is not a realistic requirement. Therefore we propose the following approach.
Instead of (EIV, NEIV) pairs, (Event Information, Explicit emotional state, State of the ESC) triples have to be gathered. This can be done within the real domain. Every time an event of the relevant event type occurs, the user has to annotate the new explicit emotional state. The event information can be extracted from the event and the current state of the ESC can be copied easily. After the triples have been gathered, the triples have to be converted to the (EIV, NEIV) pairs. It is easy to convert the event information to an EIV, as the Event Appraiser is already trained or defined. An NEIV can be found by using the inverse of the ESC. If this inverse is known, the explicit emotional state (the output of the ESC) and the state of the ESC are enough information to calculate an NEIV. Of course the inverse is not known, but using an *iterative improvement algorithm* (e.g. a simulated annealer), can solve this final problem. This algorithm has to find the NEIV that explains the explicit emotional state the best. If needed, a second criterion can be that the NEIV has to resemble the EIV as much as possible.
Using this approach makes training a Normalizer easy for a user. Annotating an explicit emotional state is easy and as the Normalizers are usually small networks, only a small amount of trainingsdata is needed.

4.3 A FIRST PROTOTYPE

A first prototype of the system has been implemented. In this prototype the various subparts were kept as simple as possible, such that we could examine whether the subparts are able to *cooperate* and whether the system as a whole is capable of simulating the required behaviour. One of the consequences of this is, that the ESC has not been implemented by the proposed Elman network yet.

In the first version only two emotion types are modelled, joy and distress, and seven event types. The two emotion types joy and distress were considered as one emotion type, the joy/distress emotion type. If the intensity of this emotion type is negative, the agent experiences distress and if the intensity is positive, the agent experiences joy. So, an EIV, an NEIV and the explicit emotional state, all consist of only one value.

The ESC has been implemented as a simple linear function (and therefore the inverse is known). The previous emotional state is one of the variables of the function. Although this simple function suits us well in this phase of the research, we are convinced that such a simple approach won't be sufficient in the future and that the described approach with an Elman network is a more powerful and a cognitively more credible approach.

The Event Appraisers were kept simple too. We will clarify this part of the architecture a bit more, by focusing on the "New_predator_spotted" event type in detail. The events belonging to this event type occur when the agent sees a predator for the first time.
As mentioned before, the variables from the OCC model are the only factors that influence the intensity of an emotion type. Although more than one variable is defined in the OCC model, that influence the emotion types joy and distress, only the variable *desirability* determines the appraisal value in this version. The event information of the "New_predator_spotted" event type consists of two properties: The current health of the agent (*health*) and the amount of predators the agent has seen previously this turn (*#seen_previously)*. If the health of the agent is low, the desirability of the event should be lower, than if the health is high, as being near a predator while health is low is very dangerous. The property #seen_previously should have a negative effect on the desirability of the event, as for instance seeing a third predator is less desirable than seeing a second predator. Taking all of this into account, the EIV for this event was defined as follows:

$$EIV = \langle desirability(health, \#seen\_previously) \rangle$$
$$= \langle 1 * health - 20 * \#seen\_previously - 100 \rangle$$

Only qualitative and no quantitative arguments and no were taken into account while drawing up the formula and as a consequence the 1, 20 and 100 in the formula could for instance just as well have been 1.5, 30 and 0.

The Normalizer is a simple feed forward network, with one hidden layer, consisting of only 2 neurons. Only 20 trainingsamples were used to train the Normalizer. The other Event Appraisers and Normalizers were constructed in a similar way.

How to test the quality of the system is a hard question and a research project by itself. This version was tested in an informal way. We checked whether the emotional behaviour shown by the system corresponds to our ideas of emotions, which we also used for annotation. This was the case and therefore we can conclude that the architecture looks promising, but a lot more experiments need to be conducted before definite conclusions can be made.

5 COMPARISON WITH OTHER MODELS

Before defining our own models we studied other models. [Pfeifer 1988] gives a nice overview of all of the AI-models of emotion until 1988. Since then, more interesting models were put forward. [Reilly & Bates 1992] also based their model on the OCC model. In [El-Nasr & Yen 1988] fuzzy logic was used, a very useful technique for the emotion domain we think. A system which we want to discuss more thoroughly in this section is Cathexis [Velásquez 1997], as it resembles our model in a couple of ways; it also has a distributed architecture and special care has been taken to allow for maximum flexibility.

Cathexis is composed of a number of *"proto-specialist" agents* [Minsky 1986], each representing a different emotion type. Within each proto-specialist, different *sensors* are monitoring both *external and internal stimuli* for the existence of the appropriate conditions that would elicit the emotion type represented by that particular proto-specialist. Input from these sensors either increases or decreases the *intensity* of the emotion type. Parallel to this process, there's always a *decay process* going on, which sees to it that the intensity of the emotion type eventually returns to a state of rest. The

way the sensors function and how the sensors and the decay function affect the intensity can be completely defined by a user. Not only the sensors can influence the intensity of an emotion, but also other proto-specialists agents can, by providing either *inhibitory* or e*xcitatory* input. A schematic overview of the system is depicted in Figure 6.
Both Cathexis and our approach have distributed architectures. The difference is, that in Cathexis the emotions are scalable and in our architecture the complexity of the domain is scalable. The advantage of making the emotions scalable is, that it makes it easier to implement a system within a simple domain. But, as the domain gets more complex, the approach of Cathexis will experience scalability problems we think, as all the proto-specialists have to perform increasingly complex task then, while no provisions have been taken to deal with this complexity.

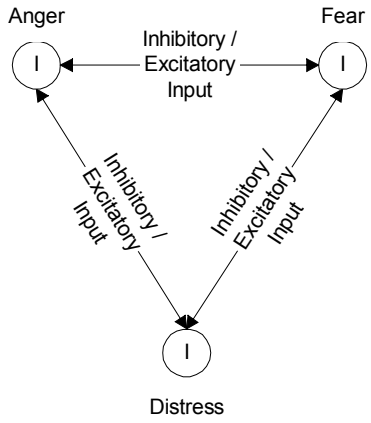


Figure 6 The "proto-specialist" agents for anger, fear and distress. (I: Intensity)

In our approach it's difficult to train the ESC for a system with multiple emotion types. But after that part has been done, adding more event types is easy.
The Cathexis system allows for a large degree of flexibility. Nevertheless we don't think the framework is powerful enough to model complex temporal emotional phenomena and phenomena in which more than two emotions participate. Our approach does have these possibilities and therefore is more powerful than Cathexis.
For Cathexis a bottom-up approach is used to make a system suitable for a particular domain. We think that it will be very difficult to precisely define all the functions and parameters in such a way that the system performs well in the case of a complex environment in which a lot of types of events can occur. In our architecture the generalization capabilities of neural networks should make it easier to handle complex environments and as a top-down approach is used, a full understanding of emotional processes and the environment is not needed; the trainer only has to know in practical cases, what the right emotional behaviour is.

6 CONCLUSION

An emotion theory based architecture for simulating emotions has been presented in this paper, which can be used to develop emotional agents acting within complex environments. A first prototype indicated that the various subparts of the distributed architecture are able to cooperate well and that the system is able to show natural emotional behaviour.

In this first implementation, the Emotional State Calculator (the ESC) was kept very simple. For the eventual system we have proposed a recurrent network of which the state of the network can be seen as an implicit emotional state. A foreseen problem of our approach is training the recurrent network, as it's difficult to obtain sufficient trainingsdata. But we feel that this problem is largely inherent to the complexity of emotions and that a nice property of our architecture is that a major part of this complexity is concentrated within a domain-independent part of the architecture.

In the future we want to expand our prototype with more emotions and event types. Furthermore we think that more research is needed on the ESC.

REFERENCES

Damasio, A.R. (1994). *Descartes' Error: Emotion, Reason, and the Human Brain*. New York: G.P. Putnam.

Elman, J.L. (1990). Finding structure in time. *Cognitive Science*, 14, pp. 179-211.

El-Nasr, M.S., & Yen, J. (1988). Agents, Emotional Intelligence and Fuzzy Logic. *Proceedings of the IEEE NAFIPS '98*, Fl.

Frijda, N.H. (1986). *The Emotions*. New York: Cambridge University Press.

Inoue, K., Kawabata, K., & Kobayashi, H. (1996). On a Decision Making System with Emotion. *IEEE International Workshop on Robot and Human Communication*, Tokyo, Japan, pp. 461-465.

Minsky, M. (1986). *The Society of Mind*. New York: Simon and Schuster.

Nijholt, A. (1999). The Twente Virtual Theatre Environment: Agents and Interactions. *Proceedings of the fifteenth Twente Workshop on Language Technology*, pp. 147-165.

Ortony, A., Clore, G.L., & Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge, UK: Cambridge University Press.

Pfeifer, R. (1988). Artifical Intelligence Models of Emotion. In V. Hamilton, G.H. Bower, & N.H. Frijda (Eds.), *Cognitive Perspectives on Emotion and Motivation*, Netherlands: Kluwer, pp. 287-320.

Reilly, W., & Bates, J. (1992). *Building emotional Agents*, Pittsburgh, PA: Carnegie Mellon University, Technical Rep. CMU-CS-92.143.

Roseman, I.J., Jose, P.E., & Spindel, M.S. (1990). Appraisal of Emotion-Eliciting Events: Testing a Theory of Discrete Emotions. *Journal of Personality and Social Psychology*, 59(5), pp. 899-915.

Velásquez, J. (1997). Modeling Emotions and Other Motivations in Synthetic Agents. In: *Proceedings of the AAAI Conference 1997,* Providence, RI, pp. 10-15.

Human Cognitive Processes in Speech Segmentation and Word Recognition

Régine Kolinsky, Vincent Goetry and Monique Radeau
Université Libre de Bruxelles
Fonds National de la Recherche Scientifique, Belgium

José Morais
Université Libre de Bruxelles

Abstract

Speech is a highly variable stimulus and there are no obvious boundaries in the signal to separate words. How then do human listeners know where words begin and end? In psycholinguistics, two major types of solutions to this segmentation problem have been proposed. According to the first type of solution, lexical segmentation is a by-product of the normal processes of word recognition. Yet, this view fails to explain how infants or language learners are able to progressively segment meaningful units from a continuous signal in the absence of or with restricted lexical knowledge. Nevertheless the infant or language learner must achieve segmentation of the input in some way in order to begin compiling a lexicon. Hence, it was proposed that segmentation and recognition processes are best achieved by a prelexical component that exploits units or cues indicating on what basis lexical access should be initiated. Experimental data on human monolingual and bilingual adults are presented to examine this notion. In particular, data from lexical decision, speech shadowing and target monitoring tasks as well as analyses of laboratory-induced perception errors are discussed.

Keywords: word segmentation; prosodic markers; processing levels; cognitive strategies.

A major question for psycholinguistics concerns the segmentation of spoken words. Given that speech is a highly variable stimulus and that there are no obvious boundaries in the signal to separate words, how do human listeners know where words begin and end?

One possibility is that listeners do not need to know this. Several models assume that word boundary information arises from the normal processes of word recognition. According to the first models devoted to speech recognition, listeners would identify words in a strictly sequential order. For example the *Cohort I* model (Marslen-Wilson & Welsh, 1978) insists upon onset alignment so that only stretches of speech corresponding to word onsets can generate the initial lexical candidates. In this model, the unfolding speech signal generates on-line a phase of lexical activation or *word initial cohort* (e.g., in French "spaghetti" will generate "silence", "savon", "stop", "scabreux", "spa", "spatule", "spaghetti" etc.), followed by a phase of deactivation in which the cohort members ("silence", "savon", "stop", "scabreux", etc.) that no longer match the gradually incoming input ("sp…") are eliminated. The result of these processes is the isolation of a unique candidate that is completely matched with the input and should correspond to the word to recognize.

Although the assumptions of the Cohort model are generally considered as being too simple and probably incorrect (e.g., Frauenfelder, 1991; Mattys, 1997), they make the model attractive by allowing it to predict the precise moment at which any particular word can be recognized in a given lexicon. This recognition point was assumed to correspond to the uniqueness point (*UP*) or the point where the word becomes unique with respect to the other words in the lexicon (e.g., the UP of "spaghetti" in French and English is at "g"). It is thus computationally inexpensive, because, in theory, only the speech before the UP must be processed to make word recognition possible. For example, only the first four phonemes of "spaghetti" should be decoded to achieve recognition, since it is the only word starting with the sequence "spag". In addition, early recognition would allow the word offset to be anticipated, and consequently

listeners could predict the onset of the next word. This model seems to solve the problem of word segmentation in continuous speech by turning it a by-product of the word recognition process itself.

If spoken word recognition were a sequential, "left-to-right" time-shadowing process, there should be a high correlation between the time that it takes to recognize a word and the position of its UP: words with early UP should be recognized faster than words with late UP. Sequential effects of this type have been observed in various tasks, for English as well as for French words (see e.g. Mattys' 1997 review). For example, they are observed in "shadowing", in which participants have to repeat back the incoming stimuli as soon and as accurately as possible (e.g., Radeau & Morais, 1990) as well as in gender classification, in which participants must decide whether the word they hear is masculine or feminine (cf. Radeau & van Berkum, 1996). In the last task, we observed that participants responded before the end of the words in 20% of the early UP items and in 10% of the late UP ones (Radeau, Mousty & Bertelson, 1989). However, words are not always recognized before their end (e.g., Bard, Shillcock, & Altmann, 1988; Grosjean, 1985; Tabossi, Burani, & Scott, 1995). In addition, UP effects are sensitive to speech rate (Radeau, Morais, Mousty & Bertelson, 2000). As a matter of fact, when the presentation of French words was speeded up from 3.6 to 5.6 syllables per second through either faster human delivery or computerized compression, UP effects (faster response latencies and higher accuracy for early over late UP words) disappeared in both shadowing and gender decision. In contrast, the effect increased at slower rate (2.2 syllables per second). The fast rate is close to that typical of conversational continuous speech (Malécot, Jonhston & Kiziar, 1972). Thus, the UP phenomenon might arise from artificial, laboratory-induced slow speech rates (e.g., Goodman & Huttenlocher, 1988, used a rate of 3.6 syllables per second, which is considerably slower than the natural one). In other words, UP effects seem to be strategic: a speech rate slower than normal would provide listeners with extended processing time and hence enable them to use powerful strategies such as time-shadowing decision unusable at faster rates (but see Mattys, 1997).

As regards word segmentation, strictly sequential models like Cohort I have been called into question by vocabulary analyses showing that most polysyllabic words contain one or several embedded words (e.g., Luce, 1986; McQueen, Cutler, Briscoe & Norris, 1995). For example, in English the word "carbone" contains "car" and the word "cardinal" contains "car" and "card". According to a sequential view, words like "card", "carbone" and "cardinal" remain in the cohort of "car" until a pause is heard at its offset. Since no such reliable cues indicate word boundaries in connected speech, "car" may need a few phonemes past its offset to be disambiguated. Since short words are the most frequent in English, recognition would seldom be possible before word offset or before the onset of the following word.

The problem of embeddedness thus requires the decision about word identity to be delayed until sufficient information is available. Delayed commitment has been introduced in subsequent models of speech recognition such as *TRACE* (McClelland & Elman, 1986) and *Shortlist* (Norris, 1994) through a multiple, exhaustive, alignment process. These models do not restrict lexical search to lexical candidates that are aligned with particular points of segmentation. In TRACE, an interactive activation network similar to the one developed by McClelland and Rumelhart (1981) for written words, many lexical candidates beginning at different points of the signal are considered in parallel. This is made possible by the alignment of a complete copy of the lexical network with each point in the input where a word might begin. The dynamics of interactive activation, in particular the inhibitory links between competing words, allows TRACE to converge on a single lexical entry (Frauenfelder & Peeters, 1990). However, there must be a very large number of such inhibitory links if the vocabulary is not trivially small, and this large number is repeated in each copy of the network, making the model computationally expensive and its architecture highly implausible. In addition, it fails to account for some important experimental results (see e.g. Norris, 1994, for review). More recent models like *Shortlist* have therefore suggested to separate the process of generating candidate words from the process of competition, which means that the problem of duplicating lexical networks disappears. Thanks to severe matching restrictions between the input and the candidates, the number of words allowed to enter the competition is kept relatively small in Shortlist compared with TRACE. As a result of multiple alignment, it nevertheless remains considerably larger than in the purely sequential Cohort I model. As noticed by Mattys (1997), models that postulate delayed commitment implicitly attribute a greater importance to accuracy than to speed in word recognition, since recognition is delayed until the entire input is segmented without ambiguity. The price for this is increased memory storage space: information has to be kept active until full disambiguation is achieved.

Like in Cohort, lexical segmentation in TRACE and Shortlist is a by-product of word recognition, although in these models it stems more specifically from direct lexical competition, i.e., from the lateral inhibition implemented by the inhibitory connections between nodes (including lexical hypotheses). If the speech signal may be segmented with no need for a separate, specific process of word boundary localization, why would we need such a process? The main problem is that all these models fail to explain how infants or language learners are able to progressively segment meaningful units from a continuous signal in the absence of or with restricted lexical knowledge. Nevertheless the infant or language learner must achieve segmentation of the input in some way in order to begin compiling a lexicon. Thus, it seems reasonable to hypothesize that recognition involves a separate procedure of segmentation of the speech input.

It was indeed proposed that both segmentation and recognition processes are best achieved by a prelexical component that exploits units or cues indicating on what basis lexical access should be initiated. Within this view, speech segmentation is a separate process that precedes the listener's attempts to recognize words. Prelexical segmentation procedures are attractive because they may provide a unified solution to the segmentation problem of both infant and adult listeners. These prelexical segmentation procedures are not incompatible with lexical competition. Rather, they can constrain the alignment process and/or reduce the number of lexical candidates (see McQueen, Norris & Cutler, 1994; Norris McQueen & Cutler, 1995; Vroomen & de Gelder, 1997).

The idea that the recognizer is provided with a separate, prelexical, segmentation component has prompted the search for a *universal segmentation unit*. Several candidates have been considered, ranging from temporally defined templates (e.g., Klatt, 1980) to abstract linguistic units. Among these proposals, the most influential in the field of psycholinguistics was that speech recognition is assisted by a syllabic segmentation procedure (e.g., Mehler, 1981). Yet, one serious problem is to establish how well human data tap into the issues these data are intended to clarify. The search for the perceptual building blocks making up the intermediate speech representations illustrates this problem quite well.

Among the experimental tasks used to study the interface between acoustic-phonetic and lexical processing, the monitoring task is one of the best known. Participants are required to detect as quickly as possible a specified sublexical property, called the target. The assumption is that response times provide direct evidence for a temporal processing sequence. Savin and Bever (1970), for instance, obtained faster responses for syllabic than for phonemic targets and concluded that syllables are the basic units of speech perception. However, this result may not reflect the precedence of syllable identification in the process of word perception. There are alternative explanations[1], among which the precedence of higher-level units in conscious access (e.g., Foss & Swinney, 1973; Marcel, 1983; Morais, 1985). Indeed, "a phonemic representation could be formed as a necessary stage before the word level is reached and yet be much slower to trigger an instrumental response (since this is a very unfamiliar task) and/or much less accessible (or not directly accessible at all) to conscious awareness." (Treisman, 1979, p. 310). The involvement of conscious awareness in the monitoring task is supported by the fact that illiterate adults, who do not possess conscious phonemic representations (they are unable to say, for example, how many "sounds" there are in "cat", cf. Morais, Cary, Alegria, & Bertelson, 1979), are very poor at detecting phonemes (Morais, Bertelson, Cary, & Alegria, 1986). They are far better at detecting syllabic targets, although they miss them more often than literates do (Morais et al, 1986). Furthermore, it is worth noting that the monitoring technique has never been used with phonetic features, despite their potential relevance in speech perception, probably because phonetic features are not of immediate conscious access, and most (literate) people would be unable to do the task at all without prior learning.

In a new version of the monitoring paradigm, called the *fragment detection* task, the relation between the phonological structure of the target (e.g., "pa" or "pal") and the phonological structure of the carrier word (e.g., "pa.lace", or "pal.mier")[2] was manipulated (Mehler, Dommergues, Frauenfelder & Segui, 1981). Response times of French listeners were found to be longer when there was no exact

[1] Other alternative explanations are the effects of the relative distance between the target and the carrier word in the process of target-word matching (McNeill & Lindig, 1973) and the adaptation of participants' behavior to the complex phonological relations that exist between targets, carrier-words and distractors used in the experiment (Norris & Cutler, 1988).

[2] The dot indicates the syllabification point.

correspondence between the target and the initial syllable of the carrier word than when there was such a correspondence. For example, "pa" was detected faster in "pa.lace" than in "pal.mier", while "pal" was detected faster in "pal.mier" than in "pa.lace". This *syllabic effect* was taken as evidence for a syllabification procedure that would operate during the process of word recognition: "the syllable is probably the output of the segmenting device operating upon the acoustic signal. The syllable is then used to access the lexicon" (p. 342).

Soon after this study, a set of cross-linguistic fragment detection experiments showed that while French listeners syllabified both French and English words, English listeners did not show a syllabic effect with either French or English materials (Cutler, Mehler, Norris & Segui, 1983, 1986). This cross-linguistic difference was interpreted as a consequence of the specific phonological structures of the two languages. French listeners segment the speech stream into syllables because their native language displays clear and little diversified syllabic structures. English listeners do not use such a strategy because English presents both widespread ambisyllabicity, i.e., consonants belonging to two syllables at once (see Kahn, 1980; Kager, 1989) and a larger variety of syllabic structures than French (see Goldman, Content & Frauenfelder, 1996). Thus, this study introduced a major conceptual change in theories of speech segmentation: the focus of research shifted from the search for the universal perceptual building block to the notion of *language-specific segmentation strategies*.

More specifically, Cutler and colleagues proposed that listeners apply a *universal rhythmic solution to the word-boundary problem, by exploiting whatever rhythmic structure characterizing their native language*. While the syllable would be the basis of rhythmic structure in French, English would be better characterized by its typical stress pattern (Cutler & Norris, 1988). This view is reminiscent of the typological work of linguists like Pike (1945) and Abercrombie (1967), who classified languages into rhythmic classes. According to this view, one should distinguish between stress-timed languages (like English and Dutch), which display regular inter-stress intervals, and syllable-timed languages (like French), in which syllables, rather than only stressed vowels, recur at regular intervals of time establishing temporal organization. This way to cluster languages, though criticized as too simplistic (e.g., Bertettino, 1989; Dauer, 1983; Nespor, 1990) and not fitting actual temporal regularities[3], is useful. The impression of different rhythmic types may be the by-product of language-specific phonological properties. In particular, syllable complexity (Bertinetto, 1981; Dauer, 1983) seems to have reliable acoustic/phonetic correlates, like the vowel/consonant temporal ratio (Ramus, Nespor & Mehler, 1999).

The rhythmic segmentation hypothesis implies that if a language presents a rhythmic structure based on some phonological construct other than the syllabic or stress pattern, this construct should be used in segmentation. This has been illustrated with Japanese, which belongs to a third category of "mora-timed" languages[4] (Ladefoged, 1975; see also Port, Dalby & O'Dell, 1987; Ramus & Mehler, 1999; Ramus et al., 1999). As a matter of fact Japanese listeners detect mora targets as easily when these mismatch as when they match the carrier word syllabic structure (e.g., "ta" in "tan.shi" vs. "ta.ni.shi"). In addition, they often miss the CVC[5] target ("tan") in CV carriers ("ta.ni.shi"), presumably because this matching requires an internal segmentation of the second mora ("ni") (Otake, Hatano, Cutler & Mehler, 1993).

Unfortunately, the fragment detection task might speak only indirectly to the role of the syllable in on-line segmentation of continuous speech. Indeed, the segmentation problem is essentially solved when listeners are presented with isolated words and targets matching the onsets of these carriers (Frauenfelder & Content, 1999). In addition, the task may tap post- rather than prelexical representations (Kolinsky, 1998; see also Frauenfelder & Content, 1999; Meunier, Content & Frauenfelder, 1997). This is suggested by the fact that literacy-induced metaphonological or orthographic representations affect the fragment detection patterns. For example, the "moraic" effect observed in Japanese (Otake et al., 1993) could result from the application of an orthographic strategy based on the written kana characters that coincide regularly with the

[3] Linguistic analyses have constantly failed to provide empirical evidence for the notion of basically different temporal regularities in "stress-timed" and "syllable-timed" languages (e.g., Delattre, 1966; Flechter, 1991; Nakatani, O'Connor & Aston, 1981; Roach 1982; Wenk & Wioland, 1982; Williams & Hiller, 1989).

[4] The mora is a unit that can be roughly described as intermediate between the syllable and the phoneme

[5] Throughout the paper, "C" stands for consonant and "V" for vowel

morae structure (Kolinsky, 1998),[6] and there is recent evidence supporting this interpretation. Testing Japanese children of various levels of kana literacy, Inagaki, Otake and Hatano (2000) observed that the mora-based fragment-detection pattern was strongly associated to kana reading level. This led them to conclude that, as children acquire kana literacy, the Japanese segmentation unit changes from a mixture of syllable- and mora-based to a strictly mora-based one.

One might of course accept the notion that literacy affects perceptual speech segmentation processes. Yet, a more conservative view, compatible with other experimental evidence (see discussion in Kolinsky, 1998, and in Morais & Kolinsky, 1994), would be that the fragment detection task taps later, post-lexical representations. The idea that literacy-dependent knowledge does not influence perceptual processes, but only post-perceptual ones, is plausible. According to our stage-processing approach (Morais & Kolinsky, 1994; Morais, Kolinsky, Ventura, & Cluytens, 1997; Kolinsky, 1998), spoken word recognition involves two broad stages, perceptual and post-perceptual. The input to the first stage is the acoustic representation of the speech signal. Operations of segmentation are carried out at this stage. These operations are modular in the Fodor's (1983) sense of modularity. They are influenced by the infant's linguistic experience and might depend, therefore, on specific properties of the listener's native language, but not on conscious, attentional, literacy-related knowledge. The output of the perceptual stage may however be re-elaborated by literacy-related knowledge at a second, post-perceptual, stage, the output of which is conscious recognition.

Stages of phonological representation and processing have been mainly ignored as a result of the influence of highly interactive models like TRACE (McClelland & Elman, 1986). In the framework of such models, if there were an influence of literacy-dependent knowledge, this should pervade the whole recognition process, including the earliest perceptual operations. In contrast, models that are partially autonomous and partially interactive, like ours, may easily propose that phonological representations are independent from literacy-related knowledge at the perceptual stage and modulated by it at a post-perceptual stage. As we will argue, a dissociation of perceptual and post-perceptual processes in terms of the presence or absence of literacy effects on spoken word recognition is suggested by the comparison of illiterate and literate Portuguese listeners (see other evidence and further discussion e.g. in Morais & Kolinsky, 1995; Morais & Kolinsky, in press a; b). In addition, it is worth noting that when the child begins to acquire literacy, his/her basic processes of speech perception have been established long ago and their reorganization under the influence of a still growing, necessarily imperfect and unstable, body of orthographic knowledge would introduce undesirable sources of error in the speech system (cf. Ventura, Kolinsky, Brito-Mendes & Morais, in press). According to our view, the paradigm of fragment monitoring has been unable to provide unequivocal evidence of perceptual codes, and thus no definite conclusion can be taken from fragment detection data concerning the idea that French listeners apply a syllabic segmentation procedure to the continuous speech input in everyday contexts (see also discussion in Dupoux & Mehler, 1982).

Yet, much effort has been devoted in the last ten years to provide psycholinguists with stronger methodological tools. Our group introduced several techniques aimed at examining early, perceptual representations. Inferences about perceptual codes, in speech as well as in other domains, may be made using indirect evidence (e.g., Marcel, 1983; Treisman, 1979). Thus, regarding speech perception, one should use experimental situations where no intentional retrieval of word constituents is required, i.e., where both the stimuli and the responses are at the word level. But how can we study word constituents without asking the listener to do anything with these constituents? We will show how this is possible by studying the influence of phonological overlap between successive words on the listeners' ability to repeat spoken words as well as by examining the occurrence of laboratory-induced speech perception errors, namely auditory word blends.

We begin with the word blends. In the early seventies, several researchers (e.g., Studdert-Kennedy & Shankweiler, 1970) observed that listeners presented with two different but simultaneous syllables, one at each ear, make fewer identification errors if the syllables share voicing (e.g., "da - ba", which are both voiced) or place of articulation (e.g., "da - ta", which are both dental) than if they share neither (e.g., the

[6] Regarding the CVC targets, the effect observed by Otake et al. (1993) may also result from phonetic/phonotactic mismatch between targets and carriers (Nakamura, Kolinsky, Spagnoletti & Morais, 1998).

double-contrast pair including the dental voiced “da” and the labial unvoiced “pa”). The fact that acoustically and phonetically closer, and thus apparently more confusable, stimuli produce fewer errors is counter-intuitive. Analyses of the participants’ errors for the double-contrast pairs revealed that most of the time all the component features of the input pair were present but recombined incorrectly in their response. As a typical example of such *phonetic feature blendings* , the input pair “da - pa” may be heard as “ta - ba”. It must be noted that if features can be wrongly combined, they must have been separately registered as independent entities at some earlier processing stage (Treisman & Paterson, 1984). Additional evidence suggests that phonetic feature blends arise at a level where acoustic information is evaluated by reference to phonetic categories (Cutting, 1976; Day, 1968; Halwes, 1969; Pisoni & McNabb, 1974; Tartter & Blumstein, 1981). Furthermore, we did not observe any difference in phonetic feature blending rates between Portuguese literate and illiterate adults (Morais, Castro & Kolinsky, 1991; Morais Castro, Scliar-Cabral, Kolinsky & Content, 1987), which supports the view that these blends arise quite early in the processing route.

Our group developed a systematic generalization of the phonetic feature blending situation that allowed us to compare directly the extent to which different speech properties are represented separately in the intermediate representations of speech (see Kolinsky, 1992). Two pseudowords (i.e. nonwords consistent with the phonotactics of the participants’ language) are presented simultaneously, each to one ear. The two stimuli (e.g., /kiʒu-bɔtɔ̃/) contain all the information necessary for the illusory perception of e.g. a French word (e.g., “bijou” – jewel -, /biʒu/, or “coton” - cotton - /kɔtɔ̃/). This information is distributed between the two stimuli, so that the combination, at some level of processing, of parts of information of one input representation with temporally contiguous parts of information from the other input representation creates the illusion. The distribution of information between the two dichotic stimuli is manipulated experimentally in order to test the relevance of different word constituents for spoken word recognition. Sets of pseudoword pairs were chosen so that all the corresponding trials would share the same resulting word illusion. The pairs within each of these sets differ in terms of the particular property whose possible “migration” from one stimulus representation to the other stimulus representation is under examination. For instance, for the illusory word “bijou” /biʒu/ or “coton” /kɔtɔ/, previously specified as the to-be-detected target, the stimulus pair is /piʒu-gɔtɔ/ for testing the voicing of the initial consonant, /giʒu-pɔtɔ/ for the place of articulation of the initial consonant, /kiʒu-bɔtɔ̃/ for the initial consonant, /bɔʒu-kitɔ/ for the first vowel, and /bitɔ̃-kɔʒu/ for the syllables. We call these trials “experimental” because they contain all the information necessary for the occurrence of a false detection of the target. Indeed, the most interesting situation is when the target is absent, but the participant still reports having heard it. However, such false alarms may occur as a consequence of simple misperception rather than of blend. Thus, in addition to the experimental trials, in which all the target properties are present but distributed between the two dichotic stimuli, we use corresponding “control” trials which differ from the experimental trials in only one stimulus of the pair. The stimulus that is perceptually closer to the target is the same in both the experimental and the control trials but, in the control trials, the other stimulus lacks the critical linguistic property that would allow the blend to occur. For example, given the target “bijou” (/biʒu/) in the initial consonant condition, the pseudoword /kiʒu/ is present in both the experimental and the control trial, and the pseudoword /dɔtɔ̃/, which lacks the labial feature, replaces in the control trial the pseudoword /bɔtɔ̃/ used in the experimental trial.

Several experiments conducted in French (see review in Kolinsky & Morais, 1996) clearly support a dominant role of the syllable. In addition, the lack of migrations of two-segment strings involved in a baseline condition in which the part of the speech signal that has to migrate in order to produce an illusory target does not correspond to any obvious phonological unit or feature (e.g., the pair /bɔtu-kiʒɔ̃/ for the target “bijou”, /biʒu/) allows us to rule out an interpretation of the syllable predominance as being a trivial length effect (Kolinsky, Morais & Cluytens, 1995). Pure coarticulatory effects, based on the fact that coarticulation is weaker between than within syllables (Liberman, Cooper, Shankweiler & Studdert-Kennedy, 1967), could account for the French results considered in isolation, but not for all the data collected with this experimental technique. Indeed, very different results were observed by using a

Portuguese version of our detection task. For both European and Brazilian Portuguese listeners, the initial consonant was found to be the property that blends the most (Kolinsky & Morais, 1993), and there is no reason to believe that coarticulatory influences are less strong in Portuguese than in French. Subsequent testing of Portuguese-speaking illiterate adults, again from both Portugal and Brazil, yielded the same pattern of results as the study on literates (Morais &, Kolinsky, 1994). Thus, at least for Portuguese, consonants have psychological reality at the perceptual level of processing, since their role can be demonstrated in a population that is unable to represent them consciously.

The French word blends results may be considered as congruent with Mehler et al.'s (1981) arguments for syllabic segmentation. We also have disclosed syllabic effects for French using another useful technique, namely *form-based priming*. When listeners receive pairs of stimuli and have to react only to the second one (called the target), a phonological overlap between the target (e.g., "blank") and the preceding "prime" (e.g., "plank") affects the time they take e.g. to repeat the target (shadowing) or to decide whether it is a word or not (lexical decision) in comparison to a condition where prime and target are unrelated. In particular, final phonological overlap reduces shadowing and lexical decision latencies (e.g., Radeau, Morais & Segui, 1995; see reviews in Slowiaczek, McQueen, Soltano & Lynch, 2000; Zwitserlood, 1996). As argued by Radeau and collaborators, pending some methodological controls, final overlap facilitation may disclose phonological representations in prelexical speech processing. First, it is not observed when either the prime or the target are presented visually in what is called cross-modal priming (see Dumay & Radeau, 1997; Radeau et al., 1995; Radeau, Segui & Morais, 1994). Thus, final overlap facilitation seems to arise during the activation of the input phonological lexicon and may be attributed to processes which are specific to speech. Second, Radeau, Besson, Fonteneau and Castro (1998) showed that final overlap facilitation is associated with electrophysiological changes at an early negative component, situated at about 400 msec (the "N400" component). This confirms the hypothesis that final overlap facilitation arises early in the processing routine. Using the shadowing situation and a delayed non-lexical decision task, Dumay, Benraïss, Barriol, Colin, Radeau and Besson (in press) observed syllabic effects on behavioral reaction times and on the N400 component, respectively. However, these effects as well as other syllabic priming effects reported in the literature (see Radeau et al.'s review) may alternatively be accounted for by the number of shared phonemes, i.e., by the quantity of overlapping phonological information rather than by syllabic structure *per se*. Empirical verification of this hypothesis is in progress. Evidence for syllables as segmentation units in French has also been provided through other techniques, for example word-spotting experiments in which listeners are asked to detect any real word embedded in nonsense disyllabic strings (McQueen, 1996). There seems to be a processing cost (on both reaction times and errors) for onset misalignment when French participants spot CVC words (e.g., "lac") embedded in nonsense carrier strings ("zu.glac" vs. "zun.lac"; cf. Dumay, Banel, Frauenfelder & Content, 1998).

Syllables may be less appropriate to the parsing of other languages like English or Dutch, which display unclear and diversified syllabic structures. An essential property of these two stress-based languages is the metrical distinction between strong syllables, which have full vowels, and weak syllables, which have reduced vowels (usually a schwa). Since most words display word-initial strong syllables in both English (Cutler & Carter, 1987; Cutler & McQueen, 1995) and Dutch (Vroomen & de Gelder, 1995), native listeners of such languages may exploit metrical patterns in locating word boundaries (Cutler, 1994; Cutler & Norris, 1988). More precisely, they would postulate a word boundary and start a lexical access attempt at every strong syllable.

In English, the use of such a "Metrical Segmentation Strategy" (*MSS*) was mainly supported by word-spotting data. For example, the MSS predicts better and faster detection of the word MINT when the second syllable of the carrier string is metrically weak (e.g., in MINtesh, /mɪntəʃ/)[7] than when it is metrically strong (e.g., in MINTAYVE, /mɪnteɪf/). Indeed, the second syllable would trigger segmentation only when it is strong, i.e. only in MINTAYVE, thus requiring assembly of the speech material across a segmentation point for successful detection. This is exactly what has been observed (e.g., Cutler & Norris, 1988; McQueen et al., 1994).

The MSS has been considered to be similar to the syllabic strategy observed in French, since "both stress in English and the syllable in French are the basis or rhythmic structure in their respective language"

[7] Upper case signals the strong syllable, lower case the weak one.

(Cutler, Dahan & van Donselaar, 1997, p.147). However, as noticed by Goetry and Kolinsky (in press), there is actually an important conceptual difference between the underlying assumptions of the MSS and the original conception of word segmentation. As a matter of fact, the *focus of research shifted from uncovering the nature and size of the prelexical unit(s) towards investigating the cue(s) that may best indicate where word boundaries are likely to occur in the speech stream.* Indeed, the fragment detection technique was aimed at uncovering the *classification* representations that are computed from the auditory input to contact the lexical representations (e.g., Frauenfelder & Tyler, 1987). Under this view, a prelexical representation of the signal is constructed as a sequence of specific units (e.g., syllables). On the contrary, the word-spotting task was aimed at testing the MSS, which is a segmentation device that does not investigate classification processes. Since the role of the MSS is to indicate *where* in the speech stream lexical access must be initiated, its heuristic is compatible with models of speech perception involving classification as well as with models involving no prelexical unit.

To illustrate this, it is worth noting that Cutler and colleagues examined the English alternative to the syllable hypothesis, which would consist in classifying the speech input into feet[8]. (Cutler & Norris, 1988). Yet, they did not observe the corresponding effects: for English speakers fragment detection was not faster when the target, for example "GAR", corresponded exactly to a foot, as in "GARGOYLE", which includes two feet, than when it was smaller than a foot, as in "GARgle", which constitutes one foot (but see Echols et al., 1997). The word-spotting task was then designed "to put on a test, in a way that directly measures speech recognition processes, the hypothesis that segmentation for lexical access occurs at strong syllables" (p. 114). Coherently, distributional analyses performed on the English and Dutch vocabularies, together with computer simulations showing the contribution of metrical cues in predicting the accurateness of recognition (Cutler, Norris & McQueen, 1996; Norris et al., 1995), provided support to the idea that the false alarm rate of the MSS is low in comparison with a segmentation procedure that considers each phoneme or syllable to be a potential word onset location.

Despite the different nature and focus of the fragment detection and word-spotting tasks, the syllabic segmentation strategy and the MSS were repeatedly considered as equivalent (although language-specific) solutions applied to the word boundary problem (Cutler, Mehler, Norris & Segui, 1992; Cutler et al., 1997). It was argued that even if the fragment detection task does not directly address the issue of segmentation, the classification of the speech input into any set of units is logically entailed by a segmentation process at the boundaries of these units (Norris & Cutler, 1985; Cutler & Norris, 1988). However, in French a pure syllabic alignment strategy would be misguided by potential resyllabification phenomena resulting from phonological processes applying across word boundaries. Indeed, in French, because of frequent phenomena like *elision*, *enchaînement* and *liaison*, syllabic boundaries do not always coincide with word boundaries (e.g., Dejean de la Batie & Bradley, 1995). Elision occurs when the dropping of phonemes ("le été" leading to "l'été") induce resyllabification of the string ("l'é.té"). Resyllabification also occurs when a final-word consonant that is always realized becomes the onset of the syllable of a following word beginning with a vowel (the phenomenon of "enchaînement", e.g. "par ici" syllabified "pa.ri.ci"). Liaison is a particular instance of resyllabification when the first word ends in a normally silent consonant, like the /t/ of the French word "petit". This consonant surfaces when it is followed by a vowel-initial word (like in "petit air"), resulting in the resyllabification of the surfaced latent consonant over word boundaries ("pe.ti.tair"). If syllables were used as alignment points in the speech signal, there would be frequent misalignments between syllable and word boundaries. Hence, a syllable alignment strategy appears to be implausible.

Consequently, the syllabic segmentation procedure proposed for French is unable to account for the successfulness of speech recognition, and is obviously less efficient than the MSS proposed for English and Dutch (see further discussion in Goetry & Kolinsky, in press). Since there is no reason to believe that French is intrinsically harder to parse than English or Dutch, one should question the notion that segmentation cues are necessarily isomorphic to classification units (Kolinsky, 1998).

Segmentation in French might be helped by cues other than syllable boundaries, whatever the classification format. Since most polysyllabic French words bear stress on their last syllable (Wenk &

[8] In English the foot is a rhythmic unit that contains a strong syllable plus, optionally, one or more following weak syllables.

Wioland, 1982), one may hypothesize that stress cues provide a powerful, deterministic cue to word-boundary location in this language (Cutler, 1990; Cutler et al., 1997; Vaissière, 1983, 1991). At the acoustic level, stress contrasts are characterized mainly by a perceivable syllabic lengthening of the stressed syllable in comparison to the unstressed ones (Flechter, 1991; Rossi, 1972; Tranel, 1987; Vaissière, 1983, 1991; Wenk & Wioland, 1982). Consequently, the French basic rhythmic unit is an iambic foot displaying a short-long durational pattern. This language may thus be better characterized as "trailer-timed" rather than "syllable-timed" (Wenk & Wioland, 1982).

A growing body of empirical evidence supports the idea that French listeners postulate a word-boundary after stressed (i.e., lengthened) syllables, thus treating them as word-offset indicators. Rietveld (1980) showed that French listeners used mainly durational contrasts in discriminating between the two interpretations of phonologically ambiguous sequences like "couplet" vs. "couple est" (verse vs. pair is). This was first observed in production, since these sequences were pronounced respectively with a trochaic (long-short: /kuːplɛ/)[9] or iambic (short-long: /kuplɛː/) pattern, and next in a perception experiment in which the durational contrasts provided the best predictor of the listeners' choice between the two semantic interpretations of the sequences. Converging data were reported by Content, Dumay and Frauenfelder (2000) and by Banel and Bacri (1994). In the last study, French listeners had to report the number of perceived words in ambiguous strings that displayed either iambic or trochaic durational patterns. Strings like /kɔrbo/ were perceived as a single disyllabic word (meaning crow) or as two monosyllabic words (/kɔr/, -body-, and /bo/, -beautiful-), depending on the durational pattern (iambic or trochaic, respectively). Further evidence for the use of stress cues in speech segmentation in French comes from word-spotting tasks involving both the detection of real words (Banel & Bacri, 1997) and the recognition of "words" belonging to a recently acquired artificial language (Banel, Frauenfelder & Perruchet, 1998).

However, it should be noted that the stimuli used in many of the former studies were obtained by artificial concatenation of syllables, and displayed durational differences reproducing the ratio observed in words pronounced in isolation (e.g., 520 milliseconds between short and long syllables). Such temporal contrasts are unlikely to occur in connected speech where temporal patterns are far less contrasted (Klatt, 1975). As previously illustrated with the effect of speech rate on UP effects (Radeau et al., 2000), one should wonder whether the strategies observed with artificially manipulated speech materials are actually used by the listeners to recognize words in continuous, fluent speech.

To this respect, the laboratory-induced juncture misperception procedure designed by Cutler and Butterfield (1992) may provide a more "ecological" situation. In this situation, listeners are required to recognize spoken sequences on the basis of partial acoustic cues, for example when sequences are presented at individual speech perception threshold. The recognition errors made by the participants, especially those related to word-boundary localization, provide information as regards the cues used in speech segmentation. Juncture misperceptions are significantly related to the sentences metrical structure in both English and Dutch. Indeed, listeners erroneously insert word boundaries mainly before strong syllables and delete word boundaries mainly before weak syllables (e.g., "conDUCT asCENTS upHILL" perceived as "the DOCtor SENDS her BILL", Cutler & Butterfield, 1992; see Vroomen, van Zon & de Gelder, 1996 for Dutch). The metrical effect on juncture misperceptions does not stem from the acoustic saliency of strong syllables over weak ones in barely audible speech (cf. Kearns, 1994), since in English similar metrical effects are observed in spontaneous misperceptions of normal speech (Cutler & Butterfield, 1992). Juncture misperceptions thus provide strong evidence supporting the use of the MSS in English.

In Dutch, however, we do not know of any study on spontaneous misperceptions of natural speech. Moreover, in a Dutch adaptation of the word-spotting task (e.g., detection of "MELK" in either /mɛlkəs/ or /mɛlkos/), metrical effects were observed on correct responses but not on reaction times (Vroomen et al., 1996). This raises the possibility that some factors induce Dutch listeners to exploit slightly different cues for speech segmentation.

One difference between English and Dutch is that the relationship between metrical structure and vowel quality, whereby strong syllables contain full vowels and weak syllables contain reduced vowels, is much more pervasive in English than in Dutch. The weak syllables of many Dutch words (e.g., "sigar",

[9] The symbol ː indicates vowel lengthening.

"kobra") contain unstressed unreduced vowels (Quené, 1993; Quené & Smith, 1992; Quené & Koster, 1998; Vroomen & de Gelder, 1995), while the syllables of the corresponding English words ("cigar", "cobra") nearly always display reduced vowels (Cutler & van Donselaar, in press; see also Fear, Cutler & Butterfield, 1995; Koster & Cutler, 1997). Consequently, vowel quality is a reliable predictor of the metrical status of the syllable in English but not in Dutch. Since Dutch word-initial syllables often bear primary stress (e.g., Vroomen & de Gelder, 1997; Vroomen, Tuomainen & de Gelder, 1998; van der Hulst, 1984), the most plausible acoustic features indicating likely word boundaries in Dutch are those related to primary stress, such as longer duration, higher intensity and flatter amplitude (e.g., Sluijter, 1995).

The idea that in Dutch word boundaries are betted signaled by the degree of the syllable stress than by the metrical distinction between strong and weak syllables has been recently supported by word-spotting data (Vroomen & de Gelder, submitted). When required to spot trochaic disyllabic words like "KRAter" (crater, /ˈkratər/[10]) in trisyllabic nonsense strings, Dutch listeners show faster reaction times when the initial syllable of the words is realised as primary stress (e.g., in /ˌpɔˈkratər/) than when it is secondary stressed (e.g., /ˈpɔˌkratər/), even after the acoustic differences between the two sets of trisyllabic strings are factored out of the decision latencies. This is at odds with the MSS, which attributes the same segmentation power to any syllable containing a full vowel, and thus would predict equally quick segmentation of "krater" in the two strings.

In English also, stress is related to word boundaries. As a matter of fact, English statistical analyses show not only that most frequent words display initial stress, but also that nearly half of the polysyllabic words include at least two strong syllables (Mattys, 2000). Thus, a *Stress Based Segmentation strategy* (*SBS*, Vroomen & de Gelder, submitted) would improve the recognizer's accuracy in comparison with MSS, which often postulates at least one erroneous word boundary. The role of stress in English has been supported by experimental evidence as well. For example, it has been shown with the word blends paradigm (Kolinsky & Morais, 1996) that the probability of misperceiving the vowel of a secondary stressed syllable is lower in a word than in a pseudoword, whereas there is no such lexical effect with the vowels of primary stressed syllables (Mattys & Samuel, 1997). This suggests that primary stressed syllables are processed with less assistance from the lexicon than secondary stressed syllables, which is consistent with the hypothesis that primary stress plays a critical role in English word recognition (see also Mattys, 2000; Mattys & Samuel, 2000).

Taken together, the studies investigating the role of stress in speech segmentation suggest that this cue is relevant for both languages displaying left-headed, trochaic, rhythmic structures (like English or Dutch) and languages presenting right-headed, iambic rhythmic patterns (like French). To further investigate this hypothesis, we ran a cross-linguistic study of juncture misperceptions in French and Dutch listeners using rhythmically matched parts of sentences presented at individual speech perception threshold (Goetry, Van de Velde & Kolinsky, in progress; see Goetry & Kolinsky, in press). In each language, the sentences included identical disyllabic strings (e.g., /livrɛ/ in French), which contained two words but displayed either a trochaic rhythmic pattern (e.g., /liːvr#ɛ/ in "j'ai vu que les livres, essentiels à notre époque, se vendaient très mal" – I saw that the books, essential in our epoch, were sold very badly -) or an iambic rhythmic pattern (e.g., /li#vrɛː#/ in "j'ai vu que les lits vrais, sans cesse vantés, sont plus confortables que ces paillasses" – I saw that the true beds, unceasingly praised, were more comfortable than those pallets -).

On the basis of the contrasted rhythmic structures of French and Dutch, which result from the fact that stressed syllables indicate word-offsets in French but word-onsets in Dutch, we expected French and Dutch listeners to display opposite juncture misperceptions of the matched sequences. That is, French listeners should more often (correctly) segment the sequences displaying a trochaic pattern than the sequences displaying an iambic pattern, while Dutch listeners should produce the reverse difference. The results showed the expected opposite relationship between rhythmic patterns and juncture misperception rates for the two sequence types across the two groups of listeners. Moreover, French listeners presented with the Dutch material displayed segmentation patterns which were closer to those observed for the French participants tested in their native language than to those observed for the Dutch listeners presented

[10] The symbols ˈ and ˌ indicate primary and secondary stress, respectively.

with the same Dutch material. Thus, the differences observed between the French and Dutch listeners cannot be attributed to some acoustical mismatch between the two material sets.

Thus, in a "trailer-language" like French (cf. Wenk & Wioland, 1982), disyllabic sequences are interpreted as a single word when they display an iambic pattern but as two monosyllabic words when they display a trochaic pattern. By contrast, in a "leader-timed" language like Dutch stress cues would indicate word onsets, thus leading to perceive a trochaic sequence as a single lexical item but an iambic sequence as two monosyllabic words. These cross-linguistic results support and extend the idea that stress cues assist speech segmentation in an opposite way in French and Dutch, by showing that these opposite prosodic effects hold true for continuous and natural speech processing.

Nonetheless, studies on monolinguals provide us with an incomplete picture of speech segmentation. As a matter of fact, most of the world's population has been estimated to use more than one language in everyday interactions (Grosjean, 1982; Harris & Nelson, 1992). One may thus wonder which processes underlie speech recognition in bilinguals who know two languages that display contrasted rhythmic regularities, like French and English or French and Dutch.

According to Cutler and collaborators, bilingual listeners would behave functionally as monolinguals in perceiving speech, since they would develop only one rhythmic segmentation strategy. In other words, rhythmic segmentation procedures would be mutually exclusive. This claim is based on the fact that, in the tasks initially used to assess syllabic segmentation in French listeners (i.e., fragment detection) and metrical segmentation in English listeners (i.e., word-spotting), native French-English bilinguals seemed to separate into two subgroups of which only one showed the (syllabic or metrical) effect previously observed in monolinguals (Cutler, Mehler, Norris & Segui, 1992).

This view implicitly assumes that, in one of their two languages, bilinguals are able to successfully segment the speech stream without exploiting the cues used by monolingual speakers. While this assumption might perhaps account for the limited listening comprehension abilities usually displayed by unbalanced bilinguals in their non-dominant or second language, it cannot account for the (nearly) monolingual-like proficiency levels of balanced bilinguals in their two languages. Indeed, if bilinguals reach a monolingual-like proficiency without the help of explicit segmentation cues, there is no reason to believe that these cues play any important role in speech segmentation in monolinguals, either. In addition, as we argued elsewhere (Goetry & Kolinsky, in press), the evidence on bilinguals' segmentation procedures is far from conclusive. First, it should be noted that in the Cutler et al.'s (1992) study, different bilingual participants were examined with the experimental situations assessing the role of syllabic and metrical structures in French and English, respectively. Thus, some of the bilinguals showing evidence for syllabification in French may also have shown evidence of stress-based segmentation in English if they had been tested for this effect, and vice versa (Kearns, 1994). Second, as acknowledged by Cutler et al. (p. 407), the comparison between the two sets of results is further complicated by the use of different tasks that may tap different processes.

Hence, we tried to address the issue of coexistence of rhythmic segmentation cues in bilinguals whose two languages display opposite rhythmic structures by using the same task and matched materials (Goetry et al., in progress; see Goetry & Kolinsky, in press). That is, we presented French-Dutch bilinguals, all French-dominant, with both the French and the Dutch materials that had induced opposite juncture misperception patterns in monolinguals tested in their respective native language. We hypothesized that the bilinguals should have developed adapted segmentation cues to correctly segment their two languages, but that the stress cues indicating words-offsets in their (French) dominant language may interfere with the use of stress cues as words-onsets (opposite) in their (Dutch) non-dominant language. As such interference may be related to the age of acquisition of the second language, we examined both early bilinguals, who had acquired French and Dutch before the age of four, and late bilinguals, who had acquired Dutch after adolescence.

For both the French and the Dutch materials, we observed no significant difference between the segmentation patterns of the two bilingual groups and those of the monolinguals groups in their respective native language. This suggests that all the bilinguals were highly familiar to the typical rhythmic structures of words in their two languages and exploited these cues to locate word boundaries in the same way as the respective monolingual groups did in their native language.

No difference was observed between early and late bilinguals for the Dutch material. This may be related to the fact both groups displayed roughly equivalent levels of comprehension abilities, both in normal and in difficult listening conditions. Together with other recent results using reaction time and brain-imaging techniques (Perani et al., 1996; Perani et al., 1998; Sanders, Yamada & Neville, 1999), our results suggest that the attained level of proficiency may be more important than age of acquisition in some aspects of second language processing. Obviously, none of these results questions the idea that age of acquisition is a major determinant of ultimate proficiency level in a second language (e.g., Johnson & Newport, 1989). Rather, they suggest that when proficiency in the second language is kept constant, age of acquisition has weaker effects on the acquisition of prosodic segmentation cues than on the acquisition of other specific aspects of that language (like non-native phonemic contrasts, see e.g. Pallier, Bosch & Sebastiàn, 1997), at least for pairs of languages in which similar prosodic word boundary cues can be used. Given its considerable implications for foreign-language learning, this issue deserves further investigation.

We tried to illustrate how psycholinguitics has handled the problem of word segmentation for about three decades. Yet, an exhaustive panorama was impossible to present here. Although the search for word boundary markers has "shifted to the prosodic arena" (Mattys, 1997, p. 318), it should be noted that prosodic cues alone are not fully reliable. For example, many Dutch and English words do not bear word-initial primary stress or do not begin with a strong syllable. In these cases, listeners are likely to take advantage of other cues that can be indicative of word boundaries. There are several other potential sources of language-specific information that listeners could draw on in segmenting words from fluent speech, such as allophonic, phonotactic, and distributional cues (e.g., Church, 1987). Phonotactics refers to constraints on the possible ordering of phonetic segments within morphemes, syllables and words in a language. Similarly, different phonetic realizations (allophones) of the same phoneme are often restricted in terms of possible positions within a word. For example, /t/ will be aspirated at the beginning but not at the end of English words (e.g., Umeda & Cocker, 1974). Recognizing and segmenting words is helped by these additional cues (see review e.g., in Goetry & Kolinsky, in press). An approach to the word boundary problem relying on several sources of information extracted and integrated from infancy on may provide a more powerful and realistic account of listeners' accurateness in recognizing continuous speech (e.g., Mattys, Jusczyk, Luce & Morgan, 1999; Christiansen, Allen & Seidenberg, 1997; Morgan & Saffran, 1995; Myers Jusczyk, Kemler Nelson, Luce, Woodward & Hirsh-Pasek., 1996; Saffran, Newport & Aslin, 1996).

In addition, speech scientists often restrict their investigation to auditory presentation of the stimuli, thereby using rather impoverished speech. As a matter of fact, in face-to-face communication, phonetic perception takes both auditory and visual information into account. Interactions between auditory speech processing and lipreading can be disclosed by making the two kinds of cues conflict. In the classic "McGurk effect" (McGurk & McDonald, 1976), presenting participants with the sound "ba" and, at the same time, the video image of a speaker pronouncing "ga" yields the illusory percept "da" (fusion), while the reverse situation tends to yield "bga" (combination). Although this *integration of auditory and visual information* is considered by some authors as part of the speech module (Liberman & Mattingly, 1985), we obtained results suggesting that it involves a strategic component. In particular, we observed an increase in the occurrence of the McGurk illusion when the presentation rate is slowed down as well as when more difficult listening conditions are used (Colin, Radeau & Deltenre, 1998a, b; other arguments were presented by Radeau, 1994 a, b, 1996, 1997, 1998 and by Radeau & Colin, 1999). We are thus again faced with the problem that language comprehension involves both automatic, perceptual routines and strategic processes, and that behavioral phenomena need to come under closer scrutiny before one can draw definite conclusions. In the case of audio-visual interactions, it is worth noting that the McGurk effect is only one among several phenomena. For example, lipreading has been shown to contribute to the identification of phonemes (e.g., Colin, Radeau, Demolin & Soquet, in press; Vroomen & de Gelder, 2000). Further investigation will be needed to assess the status of such potentially powerful sources of information for speech recognition.

REFERENCES

Abercrombie, D. (1967). *Elements of general phonetics*. Edinburgh: Edinburgh University Press.

Banel, M.-H., & Bacri, N. (1994). On metrical patterns and lexical parsing in French. *Speech Communication, 15*, 115-126.

Banel, M.-H., & Bacri, N. (1997). Reconnaissance de la parole et indices de segmentation métriques et phonotactiques. *L'année Psychologique, 97*, 77-112.

Banel, M.-H., Fraunfelder, U. H., & Perruchet, P. (1998). Contribution des indices métriques à l'apprentissage d'un langage artificiel. *Actes des XXIIèmes Journées d'Etude sur la Parole* , pp. 29-32, Martigny, Suisse.

Bard, E. G., Shillcock, R. C., & Altmann, G. T. M. (1988). The recognition of words after their acoustic offset in spontaneous speech: Effects of subsequent context. *Perception & Psychophysics, 44*, 395-408.

Bertinetto, P. M. (1989). Reflections on the dichotomy "stress" vs. "syllable-timing". *Revue de Phonétique Appliquée*, 91-93.

Christiansen, M. H., Allen, J., & Seidenberg, M. S. (1998). Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes, 13*, 221-268.

Church, K. W. (1987). Phonological parsing and lexical retrieval. *Cognition, 25*, 53-69.

Colin, C., Radeau, M., & Deltenre, P. (1998a). Intermodal interactions in speech: A French study. *Proceedings of AVSP '98*, pp. 55-60, Terrigal-Sydney, Australia.

Colin, C., Radeau, M., & Deltenre, P. (1998b). Interactions audio-visuelles dans la perception de la parole en français. *Actes des XXIIèmes Journées d'Etude sur la Parole*, pp. 205-208, Martigny, Suisse.

Colin, C. Radeau, M. Demolin, D. & Soquet, A. (in press) Visual lipreading of voicing for French stop consonants. *Proceedings of ICSLP 2000*, Beijing, China.

Content, A., Dumay, N., & Frauenfelder, U. H. (2000). The role of syllable structure in lexical segmentation: Helping listeners avoiding mondegreens. *Proceedings of the workshop on Spoken Word Access Processes,* pp. 39-42, Nijmegen, The Netherlands.

Cutler, A. (1990). Exploiting prosodic probabilities in speech segmentation. In G. T. M. Altmann (Ed.), *Cognitive models of speech processing* (pp. 105-121). Cambridge, MA: MIT Press.

Cutler, A. (1994). Segmentation problems, rhythmic solutions. *Lingua, 92*, 81-104.

Cutler, A., & Butterfield, S. (1992). Rhythmic cues to speech segmentation: Evidence from juncture misperception. *Journal of Memory and Language, 31*, 218-236.

Cutler, A., & Carter, D. M. (1987). The predominance of strong inital syllables in the English vocabulary. *Computer Speech and Language, 2*, 133-142.

Cutler, A., Dahan, D., & van Donselaar, W. (1997). Prosody in the comprehension of spoken language: A litterature review. *Language and Speech, 40*, 141-201.

Cutler, A., & McQueen, J. (1995). The recognition of lexical units in speech. In B. de Gelder & J. Morais (Eds.), *Speech and reading: A comparative approach* (pp. 33-48). Trowbridge, Wiltshire: Erlbaum (UK) Taylor & Francis.

Cutler, A., Mehler, J., Norris, D., & Segui, J. (1983). A language specific comprehension strategy. *Nature, 304*, 159-160.

Cutler, A., Mehler, J., Norris, D., & Segui, J. (1986). The syllable's differing role in the segmentation of French and English. *Journal of Memory and Language, 25*, 385-400.

Cutler, A., Mehler, J., Norris, D., & Segui, J. (1992). The monolingual nature of speech segmentation by bilinguals. *Cognitive Psychology, 24*, 381-410.

Cutler, A., & Norris, D. (1988). The role of strong syllables in segmentation for lexical access. *Journal of Experimental Psychology: Human perception and Performance, 14*(1), 113-121.

Cutler, A., Norris, D., & McQueen, J. (1996). Lexical access in continuous speech: Language-specific realisations of a universal model. In T. Otake & A. Cutler (Eds.), *Phonological structure and language processing: Cross-linguistic studies* (pp. 227-242). Berlin: Mouton de Gruyter.

Cutler, A., & van Donselaar, W. (in press). Voornaam is not a homophone: lexical prosody and lexical access in Dutch.

Cutting, J. E. (1976). Anditory and linguistic processes in speech perception: Inferences from six fusions in dichotic listening. *Psychological Review, 83*, 114-140.

Dauer, R. M. (1983). Stress-timing and syllable-timing reanalyzed. *Journal of Phonetics, 11*, 51-62.

Day, R. S. (1968). Fusion in dichotic listening (Doctoral dissertation, Stanford University, 1968). *Dissertation Abstracts International, 29*, 2649B (University microfilm No. 2669-2211).

Dejean de la Batie, B., & Bradley, D. C. (1995). Resolving word boundaries in spoken French: Native and non-native strategies. *Applied Psycholinguistics, 16*, 59-81.

Delattre, P. (1966). A comparison of syllable-length conditioning among languages. *IRAL, 7*, 295-325.

Dumay, N., Banel, M. H., Frauenfelder, U. H., & Content, A. (1998). Le rôle de la syllabe: segmentation lexicale ou classification? *Proceeedings of XXIIèmes Journées d'Etude sur la Parole*, (pp. 33-36), Martigny, Suisse.

Dumay, N., Benraïss, A., Barriol, B., Colin, C., Radeau, M., & Besson, M. (in press). Behavioral and electrophysiological study of phonological priming between bisyllabic spoken words. *Journal of Cognitive Neuroscience*.

Dumay, N., & Radeau, M. (1997). Rime and syllabic effects in phonological priming between French spoken words, *Proceedings of the '97 Eurospeech Conference*. Vol. 4, pp. 2191-2194, University of Patras, Greece.

Dupoux, E., & Mehler, J. (1992). Unifying awareness and on-line studies of speech. A tentative framework. In J. Alegria & D. Holender & J. Junça de Morais & M. Radeau (Eds.), *Analytic approach to human cognition* (pp. 59-75). Amsterdam: North-Holland.

Fear, B. D., Cutler, A., & Butterfield, S. (1995). The strong/weak distinction in English. *Journal of the Acoustical Society of America, 97*, 1893-1904.

Flechter, J. (1991). Rhythm and final lengthening in French. *Journal of Phonetics, 19*, 193-212.

Fodor, J. A. (1983). *The modularity of mind: An essay on faculty psychology*. Cambridge, MA: MIT Press.

Foss, D. J., & Swinney, D. A. (1973). On the psychological reality of the phoneme: Perception, identification and consciousness. *Journal of Verbal Learning and Verbal Behavior, 12*, 246-257.

Frauenfelder, U. H. (1991). Une introduction aux modèles de reconnaissance des mots parlés. In R. Kolinsky & J. Morais & J. Segui (Eds.), *La reconnaissance des mots dans les différentes*

modalités sensorielles: études de psycholinguistique cognitive (pp. 7-36). Paris: Presses Universitaires de France.

Frauenfelder, U. H., & Content, A. (1999). The role of the syllable in spoken word recognition: Access or segmentation? *Proceedings of the IIemes Journees d'Etudes Linguistique*, pp. 1-8, Université de Nantes, France.

Frauenfelder, U. H., & Peeters, G. (1990). On lexical segmentation in TRACE: An exercise in simulation. In G. Altmann (Ed.), *Cognitive models of speech processing: Psycholinguistic and computational perspectives* (pp. 50-86). Cambridge, MA: MIT Press.

Frauenfelder, U. H., & Tyler, L. K. (1987). The process of spoken word recognition: An introduction. *Cognition, 25*, 1-20.

Goetry, V., & Kolinsky, R. (in press). The role of rhythmic cues for speech segmentation in monolingual and bilingual listeners. *Psychologica Belgica*.

Goetry, V., Van de Velde, H. & Kolinsky, R. (in progress). *Rhythmic cues for speech segmentation in French, Dutch, and bilingual listeners: The role of stress*.

Goldman, J.-P., Content, A., & Frauenfelder, U. H. (1996). Comparaison des structures phonologiques syllabiques en français et en anglais, *Proceedings of the XXIièmes Journées d'Etudes sur la Parole*, pp. 315-318, Avignon, France.

Goodman, J. C., & Huttenlocher, J. (1988). Do we know how people identify spoken words? *Journal of Memory and Language, 27*, 684-698.

Grosjean, F. (1982). *Life with two languages: An introduction to bilingualism*. Cambridge, MA: Harvard University Press.

Grosjean, F. (1985). The recognition of words after their acoustic offset: Evidence and implications. *Perception & Psychophysics, 38*, 299-310.

Halwes, G. T. (1969). Effects of dichotic fusions on the perception of speech. *Status report to speech research*, supplement, September, Haskins Laboratories.

Harris, R. J., & Nelson, E. M. M. (1992). Bilingualism: Not the exception any more. In R. J. Harris (Ed.), *Cognitive processing in bilinguals* (pp. 3-14). Amsterdam: North-Holland.

Inagaki, K., Hatano, G., & Otake, T. (2000). The effect of Kana literacy acquisition on the speech segmentation unit used by Japanese young children. *Journal of Child Experimental Psychology, 75*, 70-91.

Johnson, J., & Newport, E. L. (1989). Critical period effects in second language learning: The influence of maturational state on the acquisition of English as a second language. *Cognitive Psychology, 21*, 60-99.

Kager, R. (1989). *A metrical theory of stress and destressing in English and Dutch*. Dordrecht: Foris.

Kahn, D. (1980). *Syllable-based generalizations in English phonology*. New York: Garland.

Kearns, R. (1994). *Prelexical speech processing by mono- and bilinguals*. Unpublished doctoral dissertation, University of Cambridge, Cambridge.

Klatt, D. H. (1975). Vowel lengthening is syntactically determined in a connected discourse. *Journal of Phonetics, 3*, 129-140.

Kolinsky, R. (1992). Conjunction errors as a tool for the study of perceptual processes. In J. Alegria, D. Holender, J. Morais, M. Radeau (Eds.), *Analytic approaches to human cognition* (pp. 133-149). Amsterdam: North-Holland.

Kolinsky, R. (1998). Spoken word recognition: a stage-processing approach to language differences. *European Journal of Cognitive Psychology, 10*(1), 1-40.

Kolinsky, R., & Morais, J. (1993). Intermediate representations in spoken word recognition: A cross-linguistic study of word illusions. *Proceedings of the 3d European Conference on Speech Communication and Technology*, pp. 731-734, Berlin, Germany.

Kolinsky, R., & Morais, J. (1996). Migrations in speech recognition. In F. Grosjean & U. H. Frauenfelder (Eds.), *Spoken word recognition paradigms. A special issue of Language and Cognitive Processes* , *11*, 611-619.

Kolinsky, R., Morais, J., & Cluytens, M. (1995). Intermediate representations in spoken word recognition: Evidence from word illusions. *Journal of Memory and Language, 34*, 19-40.

Koster, M., & Cutler, A. (1997). Segmental and suprasegmental contributions to spoken-word recognition in Dutch, *Proceedings of the Fifth European Conference on Speech Communication and Technology*, pp. 2167-2170, Rhodes, Greece.

Ladefoged, P. (1975). *A course in phonetics*. New York: Harcour, Brace Jovanovich.

Liberman, A. M., Cooper, F. S., Shankweiler, D. P., & Studdert-Kennedy, M. (1967). Perception of the speech code. *Psychological Review, 74*, 431-461.

Liberman, A.M., & Mattingly, I.G. (1985). The motor theory of speech perception revised. *Cognition*, *21*, 1-36

Luce, P. A. (1986). Neighboorhoods of words in the mental lexicon. *Research on speech perception, Technical report No. 6.* Bloomington, Ind.: Department of Psychology, Speech Research Laboratory.

Malécot, A., Johnston, R., & Kiziar, P.-A. (1972). Syllabic rate and utterance length in French. *Phonetica, 26*, 235-251.

Marcel, A. J. (1983). Conscious and unconscious perception: An approach to the relations between phenomenal experience and perceptual processes. *Cognitive Psychology, 15*, 238-300.

Marslen-wilson, W. D., & Welsh, A. (1978). Processing interactions and lexical access during word recognition in continuous speech. *Cognitive Psychology, 10*, 29-63.

Mattys, S. L. (2000). The perception of primary and secondary stress in English. *Perception & Psychophysics, 62*, 253-265.

Mattys, S. L., Jusczyk, P. W., Luce, P. A., & Morgan, J. L. (1999). Phonotactic and prosodic effects on word segmentation in infants. *Cognitive Psychology, 38*, 465-494.

Mattys, S. L., & Samuel, A. G. (1997). How lexical stress affects speech segmentation and interactivity: Evidence from the migration paradigm. *Journal of Memory and Language, 36*, 87-116.

Mattys, S. L., & Samuel, A. G. (2000). Implications of stress pattern differences in spoken word recognition. *Journal of Memory and Language, 42*, 571-596.

McClelland, J. L., & Elman, J. L. (1986). The TRACE model of speech perception. *Cognitive Psychology, 18*, 1-86.

MCClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: Part I. An account of basic findings. *Psychological Review, 88*, 375-407.

McGurk, H., & McDonald, J. (1976). Hearing lips and seeing voices. *Nature*, 264, 746-748.

MCNeill, D., & Lindig, K. (1973). The perceptual reality of phonemes, syllables, words and sentences. *Journal of Verbal Learning and Verbal Behavior, 12*, 419-430.

McQueen, J. (1996). Word spotting. In F. Grosjean & U. H. Frauenfelder (Eds.), *Spoken word recognition paradigms. A special issue of Language and Cognitive Processes*, *11*, 695-699.

McQueen, J. M., Cutler, A., Briscoe, T., & Norris, D. (1995). Models of continuous speech recognition and the contents of the vocabulary. *Language & Cognitive Processes, 10*, 309-331.

McQueen, J. M., Norris, D., & Cutler, A. (1994). Competition in spoken word recognition: Spotting words in other words. *Journal of Experimental Psychology: Learning, Memory & Cognition, 20*, 621-638.

Mehler, J. (1981). The role of syllable in speech processing: Infant and adult data. *Philosophical Transaction of the Royal Society London, B295*, 333-352.

Mehler, J., Dommergues, J.-Y., Frauenfelder, U., & Segui, J. (1981). The syllable's role in speech segmentation. *Journal of Verbal Learning and Verbal Behavior, 20*, 298-305.

Meunier, C., Content, A., Frauenfelder, U. H., & Kearns, R. (1997). The locus of the syllable effect: prelexical or lexical ? *Proceedings of the '97 Eurospeech conference*, Vol. 5, pp. 2851-2854, University of Patras, Greece.

Morais, J. (1985). Literacy and awareness of the units of speech: implications for research. *Linguistics, 23*, 707-721.

Morais, J., Bertelson, P., Cary, L., & Alegria, J. (1986). Literacy training and speech segmentation. *Cognition, 24*, 45-64.

Morais, J., Cary, L., Alegria, J., & Bertelson, P. (1979). Does awareness of speech as a consequence of phones arise spontaneously. *Cognition, 7*, 323-331.

Morais, J., Castro, S.-L., & Kolinsky, R. (1991). La reconnaissance des mots chez les adultes illetrés. In R. Kolinsky, J. Morais & J. Segui (Eds.), *La reconnaissance des mots dans les différentes modalités sensorielles: Etudes de psycholinguistique cognitive* (pp. 59-80). Paris: Presses Universitaires de France.

Morais, J., Castro, S.-L., Scliar-Cabral, L., Kolinsky, R., & Content, A. (1987). The effects of literacy on the recognition of dichotic words. *Quarterly Journal of Experimental Psychology, 39A*, 451-465.

Morais, J., & Kolinsky, R. (1994). Perception and awareness in phonological processing: The case of the phoneme. *Cognition, 50*, 287-297.

Morais, J., & Kolinsky, R. (1995). Perception and awareness in phonological processing: The case of the phoneme. In J. Mehler & S. Franck (Eds.), *Cognition on cognition* (pp. 349-359). Cambridge, MA: MIT Press.

Morais, J., & Kolinsky, R. (in press a). The linguistic consequences of literacy. In: P. Bryant & T. Nunes (Eds.), *Handbook of Literacy*. Kluwer.

Morais, J. & Kolinsky, R. (in press b). The literate mind and the universal human mind. In E. Dupoux et al., (Eds). *Cognition: a critical look.* MIT Press. Also in *Cognition: un regard critique*. Ed. Odile Jacob.

Morais, J., Kolinsky, R., Ventura, P., & Cluytens, M. (1997). Levels of processing in the phonological segmentation of speech. *Language and Cognitive Processes, 12*, 35-39.

Morgan, J. L., & Saffran, J. R. (1995). Emerging integration of sequential and suprasegmental information in preverbal speech segmentation. *Child Development, 66*, 911-936.

Myers, J., Jusczyk, P. W., Kemler Nelson, D. G., Luce, J. C., Woodward, A. L., & Hirsh-Pasek, K. (1996). Infants' sensitivity to word boundaries in fluent speech. *Journal of Child Language, 23*, 1-30.

Nakamura, M., Kolinsky, R., Spagnoletti, C., & Morais, J. (1998). Phonemic awareness in alphabetically literate Japanese adults: the influence of the first acquired writing system. *Cahiers de Psychologie Cognitive /Current Psychology of Cognition, 17*, 417-450.

Nakatani, L. H., O'Connor, K. D., & Aston, C. H. (1981). Prosodic aspects of American English speech rhythm. *Phonetica, 38*, 84-106.

Nespor, M. (1990). On the rhythm parameter in phonology. In I. M. Roca (Ed.), *Logical issues in language acquisition* (pp. 157-175). Dordrecht: Foris.

Norris, D., & Cutler, A. (1985). Juncture detection. *Linguistics, 23*, 689-705.

Norris, D., & Cutler, A. (1988). The relative accessibility of phonemes and syllables. *Perception & Psychophysics*, *43*, 541-550.

Norris, D., Mc Queen, J., & Cutler, A. (1995). Competition and segmentation in spoken-word recognition. *Journal of Experimental Psychology: Human Perception and Performance, 21*(5), 1209-1228.

Norris, D. G. (1994). Shortlist: A connectionist model of continuous speech recognition. *Cognition, 52*, 189-234.

Otake, T., Hatano, G., Cutler, A., & Mehler, J. (1993). Mora or syllabe? Speech segmentation in Japanese. *Journal of Memory and Language, 32*, 258-278.

Pallier, C., Bosch, L., & Sebastiàn, N. (1997). A limit on behavioral plasticity in vowel acquisition. *Cognition, 64*, B9-B17.

Perani, D., Dehaene, S., Grassi, F., Cohen, L., Cappa, S. F., Dupoux, E., Fazio, F., & Mehler, J. (1996). Brain processing of native and foreign languages. *Neuroreport, 7*, 2439-2444.

Perani, D., Paulesu, E., Sebastiàn-Gallès, N., Dupoux, E., Dehaene, S., Bettinardi, V., Cappa, S. F., Fazio, F., & Mehler, J. (1998). The bilingual brain: Proficiency and age of acquisition of the second language. *Brain, 121*, 1841-1852.

Pike, K. L. (1945). *The intonation of American English*. Ann Arbor, MI: University of Michigan Press.

Pisoni, D. B., & McNabb, S. D. (1974). Dichotic interactions of speech sounds and phonetic feature processing. *Brain and Language, 1*, 351-364.

Port, R. F., Dalby, J., & O'Dell, M. (1987). Evidence for mora timing in Japanese. *Journal of the Acoustical Society of America, 81*, 1574-1585.

Quené, H. (1993). Segment durations and accent as cues to word segmentation in Dutch. *Journal of the Acoustical Society of America, 94*(4), 2027-2035.

Quené, H., & Koster, M. L. (1998). Metrical segmentation in Dutch: vowel quality or stress? *Language and Speech, 41*(2), 185-202.

Quené, H., & Smith, Y. (1992). On the absence of word segmentation at "weak" syllables, *Proceedings of the International Conference of Spoken Language Processing*, Vol. 1, pp. 213-215, University of Alberta, Alberta.

Radeau, M. (1994 a). Auditory-visual spatial interaction and modularity. *Cahiers de Psychologie Cognitive /Current Psychology of Cognition*, Target Paper, 13, 3-51.

Radeau, M. (1994 b). Ventriloquism against audio-visual speech: Or, where Japanese-speaking barn owls might help. *Cahiers de Psychologie Cognitive /Current Psychology of Cognition, Author's response*, 13, 124-140.

Radeau, M. (1996). Auditory-visual interactions: from adults to neonates. *Proceedings of the Twelfth Annual Meeting of the International Society for Psychophysics,* pp. 101-106, Padua, Italy.

Radeau, M. (1997). Du ventriloque à l'embryon: Une réponse à Molyneux. In J. Proust (Ed.) *Perception et Intermodalité. Approches actuelles de la question de Molyneux* (pp. 223-252). Collection Psychologie et Sciences de la Pensée. Paris: Presses Universitaires de France.

Radeau, M. (1998). Auditory-visual interactions in spatial scene analysis: development and neural bases. *Proceedings of AVSP '98*, pp. 97-102, Terrigal-Sydney, Australia.

Radeau, M., Besson, M., Fonteneau, E., & Castro, S. L. (1998). Semantic, repetition and rime priming between spoken words: behavioral and electrophysiological evidence. *Biological Psychology, 48*, 183-204.

Radeau, M. & Colin, C. (1999). The role of spatial separation on ventriloquism and McGurk illusions. *Proceedings of Eurospeech 1999*, pp. 1295-1298, Budapest, Hungria.

Radeau, M., & Morais, J. (1990). The uniqueness point effect in the shadowing of spoken words. *Speech Communication, 9*, 155-164.

Radeau, M., Morais, J., Mousty, P., & Bertelson, P. (2000). The effect of speaking rate on the role of uniqueness point in spoken word recognition. *Journal of Memory and Language, 42*, 406-422.

Radeau, M., Morais, J., & Segui, J. (1995). Phonological priming between monosyllabic spoken words. *Journal of Experimental Psychology: Human Perception and Performance, 21*, 1297-1311.

Radeau, M., Mousty, P., & Bertelson, P. (1989). The effect of the uniqueness point in spoken-word recognition. *Psychological Research, 51*, 123-128.

Radeau, M., Segui, J., & Morais, J. (1994). The effect of overlap position in phonological priming between spoken words, *Proceedings of the 1994 International Conference on Spoken Language Processing*. Vol. 3, pp. 1419-1422, Yokohama, Japan.

Radeau, M., & van Berkum, J. A. (1996). Gender Decision. In F. Grosjean & U. H. Frauenfelder (Eds.), *Spoken word recognition paradigms. A special issue of LAnguage and Cognitive Process*, *11*, 605-610.

Ramus, F., & Mehler, J. (1999). Language identification with suprasegmental cues: A study based on speech resynthesis. *Journal of the Acoustical Society of America, 105*(1), 512-521.

Ramus, F., Nespor, M., & Mehler, J. (1999). Correlates of linguistic rhythm in the speech signal. *Cognition, 73*, 265-292.

Rietveld, A. C. M. (1980). Word boundaries in the French language. *Language and Speech, 23*, 289-296.

Roach, P. (1982). On the distinction between stress-timed and syllable-timed languages. In D. Crystal (Ed.), *Linguistic controversies* (pp. 73-79). London: Edward Arnold.

Rossi, M. (1972). Le seuil differentiel de durée. *Papers in Memory of Pierre Delattre* (pp. 435-450). The Hague: Mouton.

Saffran, J. R., Newport, E. L., & Aslin, R. N. (1996). Word segmentation: The role of distributional cues. *Journal of Memory and Language, 35*, 606-621.

Sanders, L., Yamada, Y., & Neville, H. J. (1999). Speech segmentation by native and non-native speakers: An ERP study. *Society for Neuroscience, 25*(1), 358.

Savin, H. B., & Bever, T. G. (1970). The nonperceptual reality of the phoneme. *Journal of Verbal Learning and Verbal Behavior, 9*, 295-302.

Slowiaczek, L. M., McQueen, J. M., Soltano, E. G., & Lynch, M. (2000). Phonological representations in prelexical speech processing: evidence from form-based priming. *Journal of Memory and Language, 43*, 530-560.

Sluijter, A. M. C. (1995). *Phonetic correlates of stress and accent*. Unpublished doctoral dissertation, Rijksuniversiteit Leiden, The Netherlands.

Studdert-Kennedy, M., & Shankweiler, D. P. (1970). Hemispheric specialization for speech. *Journal of the Acoustical Society of America, 48*, 579-594.

Tabossi, P., Burani, C., & Scott, P. (1995). Word identification in fluent speech. *Journal of Memory & Language, 34*, 440-467.

Tartter, V. C., & Blumstein, S. E. (1981). The effects of pitch and spectral differences on phonetic fusion in dichotic listening. *Journal of Phonetics, 9*, 251-259.

Tranel, B. (1987). *The sounds of French: an introduction*. Cambridge: Cambridge University Press.

Treisman, A. (1979). The psychological reality of levels of processing. In L. S. Cermak & F. I. M. Craik (Eds.), *Levels of processing and human memory* (pp. 301-330). Hillsdale, NJ: Lawrence Erlbaum Associates Inc.

Treisman, A., & Patterson, R. (1984). Emergent features, attention, and object perception. *Journal of Experimental Psychology: Human Perception and Performance, 10*, 12-32.

Umeda, N., & Cocker, C. H. (1974). Allophonic variation in American English. *Journal of Phonetics, 2*, 1-5.

Vaissière, J. (1983). Language-independant prosodic features. In A. Cutler & D. R. Ladd (Eds.), *Prosody: Models and Measurements* (pp. 53-66). Berlin: Springer-Verlag.

Vaissière, J. (1991). Rhythm, accentuation and final lengthening in French. In L. Nord & R. Carlson (Eds.), *International Symposium Series* (Vol. 2, pp. 108-120). New York: MacMillan Press.

Van der Hulst, H. (1984). *Syllable structure and stress in Dutch*. Dordrecht: Foris.

Ventura, P., Kolinsky, R., Brito-Mendes, C., & Morais, J. (in press). Mental representations of the syllable internal structure are influenced by orthography. *Language and Cognitive Processes*.

Vroomen, J., & de Gelder, B. (1997). Trochaic rhythm in speech segmentation, *Abstracts of the 38th Annual Meeting of the Psychonomic Society*, Vol. 2, pp. 73, Philadelphia.

Vroomen, J., & de Gelder, B. (2000). Lipreading and the compensation for coarticulation mechanism, *Proceedings of the workshop on Spoken Word Access Processes*. pp. 83-86, Nijmegen, The Netherlands.

Vroomen, J., & de Gelder, B. (submitted). Stress is a cue to word onset: Stress based speech segmentation.

Vroomen, J., & de Gelder, B. (1995). Metrical segmentation and lexical inhibition in spoken word recognition. *Journal of Experimental Psychology: Human Perception and PErformance, 21*, 98-108.

Vroomen, J., Tuomainen, J., & de Gelder, B. (1998). The role of word stress and vowel harmony in speech segmentation. *Journal of Memory and Language, 38*, 133-149.

Vroomen, J., van Zon, M., & de Gelder, B. (1996). Cues to speech segmentation: Evidence from juncture misperception and word spotting. *Memory and Cognition, 24*, 744-755.

Wenk, B. J., & Wioland, F. (1982). Is French really syllable-timed? *Journal of Phonetics, 10*, 193-216.

Williams, B., & Hiller, S. (1989). Investigating randomness in foot timing patterns in English. *Journal of the Acoustical Society of America, 85*, S59.

Zwitserlood, P. (1996). Form priming. In F. Grosjean & U. H. Frauenfelder (Eds.), *Spoken word recognition paradigms. A special issue of Language and Cognitive Processes*, *11*, 589-596.

Modeling Analogical Projection based on Pattern Perception*

Mehdi Dastani
Vrije Universiteit Amsterdam, The Netherlands
mehdi@cs.vu.nl

Bipin Indurkhya
Tokyo University of Agriculture and Technology, Japan
bipin@cc.tuat.ac.jp

Remko Scha
University of Amsterdam, The Netherlands
scha@hun.uva.nl

Abstract

One of the major unsolved problems in perception is to explain why a sensory pattern, which may in principle be perceived as having different constituent structures (gestalts), is usually perceived as having one particular gestalt. Structural Information Theory (SIT) treats Gestalt perception as disambiguation and describe it by a perceptually motivated preference relation defined on possible gestalts of patterns. The existing models of SIT ignore the context effects which is essential in determining the gestalt of patterns. We introduce a SIT model and embed it in a framework where two types of context effects are integrated: 1) the effect of simultaneous presence of other patterns, and 2) the effect of the task to be accomplished on the basis of the patterns. In order to illustrate the interaction of these types of context effects with perception, we consider proportional analogy problems defined on strings (alphabetic patterns). Such a proportional analogy problem consists of three simultaneously present strings and the task is to find a fourth string by means of analogical projections on the given strings.

To this end, we propose an extended algebraic model of SIT for strings. In this model, strings are considered as domain elements of an algebra and different gestalts of a string are represented as different algebraic terms corresponding to that string. A perceptually motivated preference relation orders different algebraic terms (gestalts) of one string. Analogical relations are defined as mappings between algebras. To integrate the context effects, the model is embedded in a framework that is characterized by two constraints. One concerns the simplicity of algebras generating the terms as well as the simplicity of the mapping between algebras (the first context effect), and the second concerns the possibility of a mapping between algebras (the second context effect).

Keywords: Algebra, analogy, context effect, grouping, information load, perception, projection, proportional analogy, structural information theory.

1 Introduction

A fundamental activity underlying cognition (by a natural or artificial agent) is that of integrating perceptual stimuli into a system of concepts. This, in turn, requires structuring and representing the stimuli. Gestalt psychologists since Wertheimer have noted that any sensory stimulus allows many structural descriptions and have sought to articulate principles which explain why certain

*We are grateful to Cees van Leeuwen and Peter van Emde Boas for many valuable discussions and suggestions during the research presented here.

structures (rather than others) are usually perceived. For example, the Gestalt research demonstrates that the structure that our cognitive system assigns to a particular stimulus is largely dependent on what context it takes into consideration. In fact, our cognitive system analyzes (either deliberately or unconsciously) the components of the stimulus as structurally analogous to components of the context. In our view, this capability of 'analogical projection' plays a crucial role in perception. We also believe that people are essentially perception machines, and their other cognitive prowess is parasitic on their perceptual capabilities. Therefore, studying and modeling the perceptual processes will also provide us an insight into the higher cognitive processes such as metaphor and creativity. (See also Hofstadter (1995) and Indurkhya (1992)). This is the prime motivating factor underlying our research program.

A well-defined class of problems which involve analogical projection in interaction with gestalt perception is constituted by proportional analogy problems with perceptual patterns. Proportional analogies follow a scheme that can be represented as "A is to B as C is to D", abbreviated as $A : B :: C : D$. The elements A, B, C and D can be verbal descriptions of concepts, as in "*gills* are to *fish* as *lungs* are to *humans*"; but they can also be perceptual patterns, such as the letter strings of Hofstadter's Copycat domain Hofstadter (1984), as in Figure 1; or line-drawings, as in Figure 2.

	A		B		C		D
1.	abba	:	abab	::	pqrrqp	:	pqrpqr
2.	abba	:	abbbbba	::	pqrrpq	:	pqrrrrrpq

Figure 1: Examples of proportional analogies based on letter strings

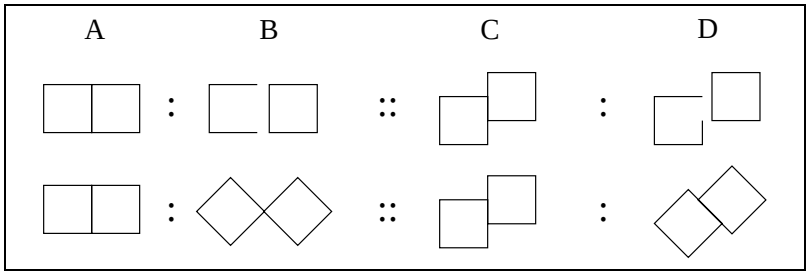


Figure 2: Examples of proportional analogies based on line-drawings

A proportional analogy problem is constructed by omitting one element in a proportional analogy relation. We write $A : B :: C : X$; the task is to find an X which is related to C in a similar way as B is related to A. Solving a proportional analogy problem requires 1) a mutual contextualization of the terms A and B, and 2) construction of a mapping between A and C which generalizes to a broader domain that includes B.

Proportional analogies between perceptual patterns demonstrate the working of gestalt disambiguation: namely how the perceived structure of one pattern influences the perceived structure of another pattern by analogical projection. In particular, the structures of A and C are to be construed as analogous to each other; and the structure of B must be construed in such a way that it is in the domain of the function that articulates that analogy. Proportional analogy problems are thus fairly complex: there are *three* patterns whose perceived structures mutually influence each other through analogical mappings. This complex interaction between gestalt perception and analogy computation can be evidenced even in very simple domains, as was demonstrated by Hofstadter (1984). For example, the first term in the two proportional analogy relations shown

in Figure 1 is the same. Yet the context influence caused by the terms B and C forces different gestalt decompositions of the term A in each case.

In this paper, we propose a method for solving proportional analogy problem by integrating the context effect in gestalt perception, analogical projection, and their interaction. To this end, we employ *Structural Information Theory* [SIT] initiated by Leeuwenberg (1971) as a starting point. This theory explains how perceptually relevant constituent structures of isolated patterns can be determined and how they can be ordered according to their perceptual preferences. The claim of SIT is that the actually perceived gestalt of a pattern in isolation is represented by its most preferred constituent structure. We then introduce an algebraic model of SIT that generates the actual perceived gestalt of one-dimensional string patterns in isolation. This algebraic model is extended to cover the mutual contextualization effect, namely how gestalts of two patterns can influence each other. Finally, we demonstrate all these features of our approach in Hofstadter's Copycat domain, and show how proportional analogy problems such as shown in Figure 1 can be solved in it.

This article is organized as follows. In the next section we briefly review the structural information theory that aims to explain the gestalt of perceptual patterns. Then, in Section 3, we develop an algebraic version of SIT. Following that, an algebraic model for proportional analogy is introduced in Section 4. In Section 5, we discuss various factors that constrain the search for appropriate gestalts in proportional analogies. In Section 6, we present an algorithm for solving proportional analogy problems — it computes the fourth term of a proportional analogy relation given the other three terms — and compare it with other computational approaches. Finally, in Section 7, we summarize the major points of our paper and briefly discuss some future research issues.

2 Gestalt Perception and SIT

The idea that our perceptual system, on encountering a stimulus, imposes a certain structure on it, rather than an arbitrary one, can be traced back to the German psychologist Wertheimer (1923). He argued that sensations, caused by stimulus patterns, are experienced as organized structures for which he used the term *Gestalt*. An important objective of the Gestalt research was to determine a set of principles which explain why certain types of patterns are experienced as having a certain organization. In particular, Wertheimer proposed five principles — known as *proximity, similarity, continuity, closure* and *habit or past experience* — to constrain the perceptual organization of patterns, and he claimed that these constitute the laws underlying the mechanism of human perception.

Relatively recently, Leeuwenberg (1971) argued, following the notion of Prägnanz introduced by Koffka (1935), that the Gestalt principles can be explained by one more basic principle of human perception. This principle is related to the structural regularity and the simplicity of stimulus patterns, which forms the fundamentals of Leeuwenberg's theory of pattern perception called Structural Information Theory (SIT). Structural regularity of a pattern is a certain hierarchical arrangement of identical pattern parts. For example, the regularity of the string pattern *abab* may be defined in terms of the identity of the first and the third a's, the second and the fourth b's, and the identity of the first and the second substrings ab. Note that these identities are in a hierarchical structure such that the identities of the substrings ab at a higher hierarchical level implies the identities of the a's and the b's at a lower hierarchical level.

Although there are various kinds of regularities in terms of which patterns may be described, only a small subset of these regularities is claimed to be perceptually motivated Van der Helm and Leeuwenberg (1991). These perceptually relevant structural regularities can be specified by means of a set of operators which are conjectured to reflect innate principles of mental representation. These operators are called ISA operators which stands for *Iteration, Symmetry* and *Alternation* operators. Each of these operators specifies a different kind of pattern regularities to which the perceptual system is sensitive. They are defined as follows:

Iteration :	$kkk\cdots kk\ (N \geq 2\ times\ k)\ \rightarrow\ N*(k)$
e.g.	$ababab\ \rightarrow\ 3*(ab)$
Symmetry :	$k_1k_2\cdots k_npk_n\cdots k_2k_1\ \rightarrow\ S[\ (k_1)(k_2)\cdots(k_n)\ ,\ (p)\ ]$
e.g.	$abccba\ \rightarrow\ S[\ abc\ ,\ ()\ ]$
	$abcab\ \rightarrow\ S[\ (ab)\ ,\ (c)\ ]$
Alternation :	$kx_1kx_2\cdots kx_n\ \rightarrow <(k)>/<(x_1)(x_2)\cdots(x_n)>$
	$x_1kx_2k\cdots x_nk\ \rightarrow <(x_1)(x_2)\cdots(x_n)>/<(k)>$
e.g.	$abkabtabw\ \rightarrow <(ab)>/<ktw>$
	$kabtabwab\ \rightarrow <ktw>/<(ab)>$

Notice the use of parentheses to indicate grouping. For instance, in the second example of symmetry, '(ab)' acts as an indivisible unit, so that its internal structure is not changed by the symmetry operator. Actually, in the SIT notation, parantheses are used around single elements too, so the first symmetry example above would be written as 'S[((a)(b)(c)), ()]', and the second as 'S[(((a)(b))), (c)]'. However, here we omit the parantheses around single elements to make the notation less cumbersome and easier to read.

Based on these operators, a pattern can be parsed into its perceptual descriptions. A perceptual description of a pattern represents a specific Gestalt of that pattern. A pattern may be parsed according to different sequences of ISA operators such that different perceptual descriptions, representing different Gestalts of the pattern, result. For example, $3*(\ (ab)\ )$ represents one Gestalt for *ababab*, which sees it as composed of repeating (ab) thrice. However, $<a>/<bbb>$ represents a different Gestalt for the same string which sees it as a being alternated by b's. The process of finding the Gestalt of a stimulus, then, becomes essentially an exercise in disambiguation.

In order to disambiguate the set of alternative Gestalts and decide the preferred perceptual description of a pattern, a measure is introduced which assigns a complexity value to each description of that pattern. The complexity value, called *information load* [IL], is designed to measure the amount of perceptually motivated regularities within a pattern. As SIT has evolved over the years, the precise manner of computing IL has also been subject to revisions. We describe the general idea intuitively here. Later, in Section 5, we will present a precise manner of computing IL based on the 'new' IL definition proposed in Van der Helm and Leeuwenberg (1991).

The intuitive idea behind IL is that it should reflect the structural complexity of a gestalt. In one version, the IL of a Gestalt description is computed by adding the number of occurrences of individual elements in that description (not including the structural operators) to the number of units larger than one element. For example, the information load of $3*(\ (ab)\ )$ is 3 because it has two elements, namely a and b, and there is one unit, namely (ab), consisting of more than one element. On the other hand, the information load of $<a>/<bbb>$ is 4 because it has four elements (note that each occurrence of b is counted separately). A description of a pattern which has the lowest complexity value is thought to employ the highest amount of perceptually motivated regularities of that pattern. Therefore, the pattern description with the lowest complexity value is claimed to represent the pattern in the most simple and cognitively economical way. Thus, in the above example, the Gestalt description $3*(ab)$ is preferred for *ababab* because its complexity value is lower.

3 An Algebraic Version of SIT

There are three motivating factors in our wanting to develop an extended version of SIT. First of all, notice that the perceptual structures covered in SIT are based only on identities between constituents of the structure. For example, when *aaaa* is written in SIT as $4*(a)$ it is recognized that the four elements appearing in the pattern are identical. But it is also possible to consider other relations that may be specific to the domain. For instance, in the geometric figures domain, one may consider the operation *rotate-left.* Or in a domain with ordered elements, such as Copycat, we may consider successor and predecessor functions, generalized to also apply to sequences of elements. In the Copycat domain, this mean we define functions *succ* and *pred* such that $succ(b) = c$, $succ(abc) = bcd$, and so on. Using these functions, we may notice a regularity in *abc*, namely that it is a successor sequence: the next element of the sequence is obtained by taking the successor

of the previous element. [1] Another factor is that though SIT works well in choosing the preferred gestalt for isolated stimuli, it cannot model the context effect, or how different gestalts of the same stimulus are preferred in different contexts. This phenomenon is demonstrated by the proportional analogies of Figs. 1 and 2. Finally, for modeling analogies and metaphors, we would like to be able to consider mappings between gestalts.

The algebraic approach developed here is aimed at addressing these three issues. An algebra is essentially a domain of objects, and a number of operators (or functions) defined on these objects. An n-ary operator takes as input n objects, and results in another object in the domain. The operators of the algebra essentially endow the objects of its domain with structure in that they allow some objects to be combined into another object; or, in another way of looking at it, allow an object to be decomposed into its component objects. In order to develop an algebraic version of SIT, we define the structural operators of SIT as algebraic operators. We now demonstrate how to do this in Hofstadter's Copycat domain. First, we must define the domain of objects that are of interest to us. For the Copycat domain, this is simply the set of all finite non-null strings composed from letters $\{a, b, \cdots, z\}$. We call this the set of *simple* string patterns.

Definition 1 *The domain D_s of simple one-dimensional string patterns is defined as the set of all finite strings of length one or more composed from $\{a, b, \ldots, z\}$.*

Notice that SIT uses an implicit grouping operator to make a string pattern into an indivisible unit. This is indicated in SIT by nested levels of parentheses. These indivisible units are needed to describe non-mirror symmetrical patterns. For example, the mirror-symmetrical string *abccba* can be described as $S[abc, ()]$, but the non-mirror-symmetrical string *abcbca* can be described as $S[a(bc), ()]$. In the second case, the string *bc* is treated as a single unit and is written as (bc) to distinguish it from *bc*. We follow the same convention. In order to define an algebraic symmetry operator, we now extend the domain of simple one-dimensional strings to include 'grouped' strings.

Definition 2 *The domain of grouped one-dimensional string patterns D is defined as follows:*
1) If $x \in \{a, \ldots, z\}$ then $x \in D$,
2) If $x_1, \ldots, x_n \in D$ then $x_1 \cdots x_n \in D$,
3) If $x_1, \ldots, x_n \in D$ and $n > 1$ then $(x_1 \cdots x_n) \in D$

Note that $x_1 \cdots x_n$ in this definition is one string. According to this definition (ab), $(ab)(cd)$, $ab(cd)$, $(ab)(c(de))fg$, etc. are all in D. Notice that we do not allow single element groups like (a) to be generated because it makes the notation needlessly cumbersome by generating groups like $(((a)))$ or $(((abc)))$ which are essentially equivalent to a and (abc), respectively. It is this domain D that we use for defining the SIT operators algebraically. We also define the notion of *unit* object, which is either a simple one-element string like a or x, or a single unit enclosed by a pair of parentheses like (ab), $(ab(cd))$, $((abc)(de))$, etc.

Definition 3 *An object x of D is called a unit object if either:*
1) $x \in \{a, \ldots, z\}$, or
2) x is of the form $(x_1 \cdots x_n)$

Thus, according to this definition, grouped strings like abc, $(ab)(cd)$, $ab(cd)$ are ruled out as unit objects.

The SIT operators can now be easily defined over D. As mentioned earlier, we would also like to allow *domain-dependent* regularities and cognitive operators to influence the preferred structural gestalt. Consequently, we let F_D be the set of all domain-dependent or cognitive operators that might play a role in the construction of gestalts. For the time being we limit ourselves to one-place

[1] This issue is explicitly addressed in Van der Helm and Leeuwenberg (1991); however they argue that such operators are cognitive and not perceptual, and therefore do not need to be incorporated in the perceptual coding principles. But we feel that while operators requiring addition or otherwise complex operations may be left out, simple operators like 'successor', 'rotate-left' do end up playing a critical role in perception, and ought to be included in the gestalts. In any case, we are interested in representing cognitive operators as well, and being able to model how they affect the structural representation of perceptual stimuli.

operators. We then augment the iteration and alternation operators so that they take an extra argument that is an element of F_D. Moreover, we will represent the concatenation operator, which is tacitly assumed in SIT, explicitly in our algebra by introducing an operator called *Con*. Finally, in order to express the unit structure of string patterns explicitly, a new operator, called *Unit* operator, is added to the set of algebraic operators. Since the unit structure of string patterns is already expressed by the parentheses, the Unit operator may seem to be superfluous. However, the introduction of the Unit operator has the advantage that all pattern structures are expressed by explicit operators. We are now ready to define the *SIT algebra* over the domain D as follows:

Definition 4 *A SIT-algebra over the domain D is a quadruple $< D, \mathcal{N}, \mathcal{F}, \mathcal{S} >$, where D is the domain of objects, $\mathcal{N}$ is the set of natural numbers; F is a set of domain-dependent operators (so that each $f \in F$ is a function from D^n to D for some n); and $\mathcal{S}$, the set of SIT-operators, contains the following operators. In the following, $X = x_1 \cdots x_k$ is a string pattern from D, where each of x_i is an object; $Y, Y_1, \ldots, Y_m$, with $m > 1$, are arbitrary string patterns from D; f is a one-place domain-dependent operator ($f \in \{id, succ, pred, \ldots\}$); and $n \in \mathcal{N}$. We write $f^n(X)$ to indicate that the one-place function f is applied n times to X.*

$$
\begin{array}{ll}
Iter(Y, f, n) & \rightarrow Y\ f(Y) \cdots f^{n-1}(Y) \\
Sym_e(X) & \rightarrow x_1 \cdots x_k x_k \cdots x_1 \\
Sym_o(X, Y) & \rightarrow x_1 \cdots x_k Y x_k \cdots x_1 \\
Alt_r(Y, f, X) & \rightarrow Y\ x_1\ f(Y)\ x_2 \cdots f^{k-1}(Y)\ x_k \\
Alt_l(Y, f, X) & \rightarrow x_1\ Y\ x_2\ f(Y) \cdots x_k\ f^{k-1}(Y) \\
Con(Y_1, \ldots, Y_m) & \rightarrow Y_1 \cdots Y_m \\
Unit(Y_1 \cdots Y_m) & \rightarrow (Y_1 \cdots Y_m)
\end{array}
$$

Note that in the definitions above, the particular variable used in each argument position conveys its sort. For example, the first argument of *Iter* must come from D, the second argument from F, and so on. The output of each operator belongs to D. Note also that we need to introduce the identity operator *id* explicitly, whenever we want the iteration and alternation operators to use identity of elements and units in constructing gestalts.

Below are some examples of gestalt of string patterns as provided by the SIT-algebra.

$$
\begin{array}{ll}
Iter(a, succ, 3) & \rightarrow abc \\
Iter(Iter(c, id, 2), pred, 3) & \rightarrow ccbbaa \\
Sym_e(Con(a, Unit(Con(b, c)), d)) & \rightarrow a(bc)dd(bc)a \\
Alt_r(a, succ, Con(f, k, t)) & \rightarrow afbkct \\
Con(k, f, u) & \rightarrow kfu
\end{array}
$$

Now we can consider the gestalts of patterns as terms of the *SIT-algebra*. Thus, in the *SIT-algebra*, the string *abc* has an iteration structure according to which it is considered a single chunk. Also note that the structural operators of the SIT-algebra do not specify any regularity in the pattern kfu; its perceptual chunking therefore consists of three subchunks, k, f, and u.

The structural description of an object shows how the object is built out of other elements. This corresponds to what is called a *term* or an Ω-*word* of an algebra Cohn (1981)(page 116). Thus, the set of all terms of the *SIT-algebra* corresponds to the set of all possible gestalts for the universe D. This is defined as follows:

Definition 5 *The class of gestalts over the SIT-algebra, denoted by $\mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, is recursively defined as follows:*

- *For all $X \in D$, $X \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$*
- *If $f \in F$, $n \in \mathcal{N}$, and $X \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, then $Iter[X, f, n] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$*
- *If $X \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, then $Sym_e[X] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$*
- *If $X_1, X_2 \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, then $Sym_o[X_1, X_2] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$,*

- *If* $f \in F_D$, *and* $X_1, X_2 \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, *then*
  $Alt_r[X_1, f, X_2] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$ *and*
  $Alt_l[X_1, f, X_2] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$,

- *If* $X_1, \ldots, X_n \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, *with* $n > 1$, *then* $Unit[X_1 \cdots X_n] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, *and*

- *If* $X_1, \ldots, X_n \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, *with* $n > 1$, *then* $Con[X_1, \ldots, X_n] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$,

Notice that we use square brackets instead of round ones to emphasize that these gestalts are not evaluated. Then, an *evaluation* function can be defined in the standard way which evaluates each gestalt into a string (grouped or simple). For lack of space, we will omit its definition here.

Based on this evaluation function, the extensional identity (which is not the structural identity) of gestalts can be defined. Two gestalts are extensionally identical if and only if they both get evaluated to the same element. Two extensionally identical terms may thus constitute different gestalts of the same pattern.

Though Defs. 4 and 5 make a clear distinction between the domain-dependent operators F and SIT-operators $\mathcal{S}$, which is necessary because one-place domain-dependent operators can appear as arguments to some of the SIT-operators, for defining analogical mappings and describing algorithms for computing them, we can lump these two classes of operators together in a set, say $\mathcal{F}$. Then, by also dropping the explicit mention of $\mathcal{N}$, we can simplify our notation of SIT algebra to be simply $< D, \mathcal{F} >$, where D is the set of objects and $\mathcal{F}$ is the set of operators.

4 Formalizing Proportional Analogy in SIT

Given that gestalts are formalized as algebraic terms, we can now turn to the problem of how to characterize analogies between different gestalts — in particular, how to characterize proportional analogy relations. One obvious approach is to use the notion of structural identity between gestalts. For example, $Iter(a, succ, 3)$ has the same structure as $Iter(p, pred, 3)$. This can be formalized using *term unification* Siekmann (1989) between tree-like structural descriptions (gestalts). There are two ways to do it. One is to define a notion of *structural template* which is essentially a gestalt containing variables, and then say that two gestalts g_1 and g_2 in $\mathcal{G}_{<\mathcal{U},\mathcal{F}>}$ are analogous if there exists some structural template g_{st}, and substitutions θ_1 and θ_2 such that $\theta_1 \circ g_{st} = g_1$ and $\theta_2 \circ g_{st} = g_2$. An equivalent definition is to allow *inverse substitutions* — so that if θ is a substitution replacing certain variables with terms, θ^{-1} would replace the terms with the corresponding variables — and say that two gestalts g_1 and g_2 are analogous if there exists substitutions θ_1 and θ_2 such that $\theta_1^{-1} \circ g_1 = \theta_2^{-1} \circ g_2$.

This approach, however, seems too rigid because trivial changes in the structural description tree destroy the analogy. For example, the operator *Con* is associative, so that *agk* could be written as $Con(a, Con(g, k))$ or $Con(Con(a, g), k)$. However, the former would be considered analogous to $Con(b, Con(i, j))$ but the latter would not. To make the characterization of analogy invariant with respect to such trite changes in structural descriptions, we can allow a *unification theory* to be specified and used in the unification algorithm. The unification theory is essentially a set of axioms, where each axiom specifies a difference that may exist (or is allowed) between the terms that have to be unified. These differences may be concerned with, for instance, arity, recursion depth, order of arguments, etc. For example, to reflect the associativity of *Con*, we could include the axiom $Con(X, Con(Y, Z)) = Con(Con(X, Y), Z))$ in the unification theory. Now both the above mentioned gestalts of *agk* would unify with the above gestalt for *bij*. It should be pointed out, however, that adding axioms makes the unification problem in most cases computationally intractable Baxter (1977).

Another approach to this problem is to follow the framework presented by Indurkhya (1991) and Indurkhya (1992). Here the source and the target domains are formalized as algebras – in particular, as sub-algebras of the SIT-algebra – and analogical relations between these domains are defined as local homomorphisms between these subalgebras, which are referred to as *representation algebras*. This is the approach we will employ in this paper. We explain it in more detail in the rest

of this section. To do that, we successively introduce the notions of correspondence, sub-algebra, and (local) homomorphism.

Definition 6 *A correspondence over two algebras is a relation between them that preserves the algebraic structures. In other words, given two algebras* $< D_1, F_1 >$ *and* $< D_2, F_2 >$*, a* correspondence *between them is a pair* $< \Delta, \Omega >$ *where:*

1. $\Delta \subseteq D_1 \times D_2$,
2. $\Omega(n) \subseteq (F_1(n) \times F_2(n))$ *for all* n*, where* n *refers to the arity of a function, and*
3. *if* $[a_1, b_1], \ldots, [a_n, b_n] \in \Delta$ *and* $[\omega, \sigma] \in \Omega(n)$ *then* $[\omega(a_1 \ldots a_n), \sigma(b_1 \ldots b_n)] \in \Delta$

In particular, a correspondence is itself an algebra. The elements of this algebra are pairs of elements, one from each initiating algebra, and the operators of this algebra are pairs of operators, also one from each initiating algebra.

The algebraic model can be applied directly to the *SIT-algebra*. However, in any given context, we may want to restrict our attention to only a small subset of the universe and a small class of operators; in other words, to a representation algebra. To define this formally, we need the notion of a subalgebra.

Definition 7 *A pair* $< E, G >$ *is a subalgebra of an algebra* $< D, F >$ *when* $E \subseteq D$*,* $G(n) \subseteq F(n)$ *for all* n*, and if* $e_1, \ldots, e_n \in E$ *and* $g \in G$*, then* $g(e_1, \ldots, e_n) \in E$*.*

A representation algebra is essentially a subalgebra that can be finitely generated. Finite generativity means that the set of operators is finite; and all the objects in its domain can be generated by a finite subset of it (by repeated application of operators).

Using these definitions, we model proportional analogies of the form "A is to B as C is to D" as consisting of two representation algebras of the SIT algebra, one generating the terms A and B, and the other generating the terms C and D, and a correspondence between the two representation algebras. For example, consider the proportional analogy "$abc : abd :: ijk : ijl$". The representation algebra $< \{ab, c\}, \{Con, succ\} >$ generates some descriptions for both "abc" and "abd". These descriptions might be "$Con(ab, c)$" and "$Con(ab, succ(c))$", respectively. Similarly, the representation algebra $< \{ij, k\}, \{Con, succ\} >$ generates (among others) the descriptions "$Con(ij, k)$" and "$Con(ij, succ(k))$", respectively, for "ijk" and "ijl". The proportional analogy is then represented by the following correspondence between the two representation algebras: $< \{[ab, ij], [c, k]\}, \{[Con, Con], [succ, succ]\} >$.
This characterization seems quite resilient because for any given pair of gestalts deemed analogous by this definition, any change in one of them resulting from the semantic properties of the domain-dependent operators can be reflected in an analogous change in the other gestalt.

There are two major consequences of defining the analogy relation as a correspondence. The first one is that many-to-many analogy relations are allowed, so there is no directionality to an analogy. Secondly, there is the constraint due to Definition 6, which says that though partial relations are allowed, there must be a closure. Incidentally, this closure property ensures that the domain and the co-domain of an analogy relation are algebras themselves.

For computational modeling of proportional analogy problems we would like to tighten the first constraint and loosen the second constraint. First, we introduce the functionality requirement on the analogy relations, so that one-to-many relations are not allowed. Such correspondences are known as *homomorphisms* and are formally defined as follows:

Definition 8 *Let* $< D_1, F_1 >$ *and* $< D_2, F_2 >$ *be two algebras. A correspondence* $< \Delta, \Omega >$ *over* $< D_1, F_1 >$ *and* $< D_2, F_2 >$ *is a homomorphism if and only if* Δ *and* Ω *are such that whenever* $[x, y]$ *and* $[x, y']$ *are both in* Δ *or in* Ω *then* $y = y'$*.*

Imposing the homomorphism condition on analogy relations gives them directionality: it results in a mapping *from* $< D_1, F_1 >$ *to* $< D_2, F_2 >$, rather than between them. Reversing the direction of the mapping does not always yield a homomorphism (though it will always be a correspondence.) The homomorphism requirement is necessary because in solving a proportional analogy problem, there is the implicit direction from the known elements to the unknown.

Now with respect to the closure characteristic mentioned above, we find that it is computationally expensive, because to establish whether a mapping is structure preserving, a large amount of computation is necessary to check for all the ways of applying various operators to all possible tuples of objects. To address this issue, we introduce the notion of *local homomorphism.*

Definition 9 *Let* $< D_1, F_1 >$ *and* $< D_2, F_2 >$ *be two algebras. A relation* $< \Delta, \Omega >$ *between* $< D_1, F_1 >$ *and* $< D_2, F_2 >$ *is a* local homomorphism *if and only if,*

- *whenever* $[x, y]$ *and* $[x, y']$ *are both in* Δ *or in* Ω *then* $y = y'$*; and*
- *whenever* $[a_1, b_1], \ldots, [a_n, b_n] \in \Delta$, $[f, g] \in \Omega(n)$*, and there is* y *such that* $[f(a_1, \ldots, a_n), y] \in \Delta$ *then* $y = g(b_1, \ldots, b_n)$

Thus, a local homomorphism is concerned with only those objects that are included in its domain. Notice that it is a weaker condition in that every homomorphism is a local homomorphism, but not vice versa. In developing a computational model of proportional analogy in Section 6, we will use the notion of local homomorphism.

5 Introducing Constraints on Gestalts

Given a *SIT-algebra*, there are many possible representation algebras (subalgebras); given any representation algebra, there are many possible structural descriptions; and given any two representation algebras, there may be many possible correspondences between them. In this section, we present various factors that constrain the generation of gestalts for proportional analogy problems.

As mentioned before in Section 2, SIT assigns a complexity value called information load (IL) to each gestalt description. We also described there the IL definition proposed by Van der Helm and Leeuwenberg (1991), according to which the information load of a Gestalt description is computed by adding the number of occurrences of individual elements to the number of objects (See def. 3) consisting of more than one element (like (ab)). For adapting this to the algebraic version of SIT, we introduce one small modification: we count any embedded domain-dependent operators (except the identity operator) as 'elements'; but, as in Van der Helm and Leeuwenberg (1991), SIT operators are not viewed as contributing to the information load. So, for instance, given the pattern *abccba*, its gestalt $g_1 = Sym_e(Iter(a, succ, 3))$ has an IL of 2, because it has two elements a and *succ*, and neither is an object consisting of more than one element. But the gestalt $g_2 = Sym_o((ab), Iter(c, id, 2))$ has an IL of 4, because it has three elements a, b and c, and one object consisting of more than one element (ab). Applying the minimality principle — which states that the preferable gestalt for a given pattern P, among all extensionally equivalent terms evaluating to P, is the one with the lowest complexity value — predicts that g_1 would be chosen over g_2.

However, the minimality principle ignores the mutual contextualization effect in proportional analogies. This is because the lowest complexity gestalts, taken in isolation, of two individual patterns may not always result in the lowest collective complexity when the two patterns are presented together. This effect can be seen in the analogy *abccba* : *ccabbacc* :: *pqrrqp* : *rrpqqprr*. We just saw above that the first pattern, namely *abccba*, considered out of context has the preferred gestalt g_1 of an even symmetry structure, but within the context of the analogy, it would be preferable to see it as an odd symmetry structure g_2, even though it has a higher information load. The reason is that g_2 shares common substructures with the preferred gestalt $g_3 = S_o(Iter(c, id, 2), Sym_e(ab))$ of *ccabbacc*. The fact that the preferred gestalts of two simultaneously present patterns share most substructures and result therefore in a lower collective information load will be called the *simplicity constraint.*

In order to incorporate this feature, what we would like is that when two patterns are present together, any common substructure between them adds to the IL only once. This effect can also be achieved by defining a complexity ordering on representation algebras, which is determined by the number of elements in it: the more elements in it, the higher the complexity. The underlying idea here is that when two patterns have gestalts that overlap, the representation algebra that generates these two gestalts will have a lower complexity. For instance, the minimal representation algebra that generates the gestalts g_1 and g_3 for the first two patterns of the proportional analogy relation mentioned above is $< \{a, ab, c\}, \{succ\} >$, which has four elements. (Notice that we do not mention or count the ISA operators.) But if we consider the gestalt g_2 for *abccba*, then g_2 and g_3 can be generated by the representation algebra $< \{ab, c\}, \emptyset >$, with only two elements.

For proportional analogy, there is an additional constraint, which we refer to as the *projectability constraint*, namely that it must be possible to construct an analogical mapping between the corresponding terms of the analogy relation. The lowest complexity gestalts of two patterns may not be appropriate for constructing analogical relations between them. For example, consider the analogy relation $abccba : abcccccba :: ppqrpp : ppqrstupp$. The first term, the same as in the example above, has a preferred gestalt of $Sym_e(Iter(a, succ, 3))$ $(IL = 2)$, but this does not form any appropriate analogical mapping with the preferred gestalt for the third term (*ppqrpp*), namely, $Sym_o(pp, (qr))$ $(IL = 5; p, p, q, r, (qr))$. To discover the mapping underlying this analogy, we may consider a higher information load gestalt for the first term, namely $Sym_o(ab, (cc))$ $(IL = 5; a, b, c, c, (cc))$. Thus, the preferred gestalts for patterns occurring in a proportional analogy relation ought to have a low complexity, ought to be generated by a simple representation algebra, and ought to satisfy the projectability condition. Notice that we are saying 'ought to' because these three constraints interfere with each other. We will now see one heuristic approach that tries to combine these constraints.

6 Computational Modeling of Proportional Analogy

In this section we propose a heuristic algorithm based on our formalism for solving proportional analogy problems. We first discuss an additional constraint imposed on the formalism due to computational requirements, and then formally define the notion of 'projectability' that plays a key role in our algorithm. Then, in Section 6.2, we describe Van der Helm & Leeuwenberg's algorithm for computing the preferred gestalt of a pattern, on which the top-level structure of our algorithm is based. In Section 6.3, we present an algorithm in our algebraic framework to solve proportional analogies, which can be seen as an extension of Van der Helm & Leeuwenberg's algorithm to incorporate the mutual contextualization effect. This algorithm uses a "test for projectability condition" module to generate mapping between algebras, and the algorithm for this module is explained in Section 6.4. We will present two simple examples in Section 6.5 to illustrate the working of our algorithm. Finally, in Section 6.6, we discuss a limitation of our algorithm.

6.1 Formal Preliminaries: Projectability

Earlier in Section 4, we defined proportional analogy "A is to B as C is to D" as a correspondence between two algebras, one of which generates gestalts for the terms A and B, and the other for C and D. In solving the proportional analogy problems, however, the task is to find an appropriate fourth term D, given the other three terms A, B and C. In this case, it is more useful and computationally efficient to require a stricter condition so that the analogy is a local homomorphism from the algebra that generates A and B to the algebra that generates C. Recall that the homomorphism condition rules out one-to-many relations, and the locality constraint requires a check for structure preserving property only in a restricted domain. Given these conditions, we can apply the local homomorphism to the gestalt of B to generate a gestalt for the fourth term D, thereby coming up with a solution of the analogy problem.

Next we define the concept of *projectability* that was introduced at the end of Section 5. Intuitively, a triple of gestalts (t_1, t_2, t_3) is said to be projectable if and only if there exists a representation algebra that generates gestalts t_1 and t_2, a representation algebra that generates

gestalts t_3, and a local homomorphism from the first algebra to the second which, when applied to t_1, yields t_3.[2] Formally, it is defined as follows:

Definition 10 *A triple of gestalts (t_1, t_2, t_3) is said to be projectable if and only if there exists a representation algebra $< D_1, F_1 >$ such that t_1 and t_2 are in $\mathcal{G}_{<D_1,F_1>}$, a representation algebra $< D_2, F_2 >$ such that t_3 is in $\mathcal{G}_{<D_2,F_2>}$, and a local homomorphism $< \Delta, \Omega >$ from $< D_1, F_1 >$ to $< D_2, F_2 >$ such that $< \Delta, \Omega > (t_1) = t_3$.*

Later in Section 6.4 we will present a heuristic algorithm to test for projectability.

6.2 Computing the Context-Independent Preferred Gestalt

Van der Helm and Leeuwenberg (1986) present an algorithm for computing the preferred perceptual gestalt of a pattern. This algorithm works in three steps. In the first step, a directed acyclic graph is constructed for the given pattern. If we place an index after each element in the pattern, starting from the leftmost element, then each node in the graph would correspond to an index, and each link in the graph from node i to j corresponds to a gestalt for the subpattern starting at position i and ending at position j. For the pattern *abccba*, this is illustrated in Figure 3(A). In the second step, all those links that represent subpatterns of length more than one, and are not covered by a SIT structural operator are removed. This is shown in Figure 3(B). Now in this reduced graph, each path from the start node to the end node corresponds to a gestalt for the whole pattern. So in the third step, the shortest path from the start node to the end node is computed.

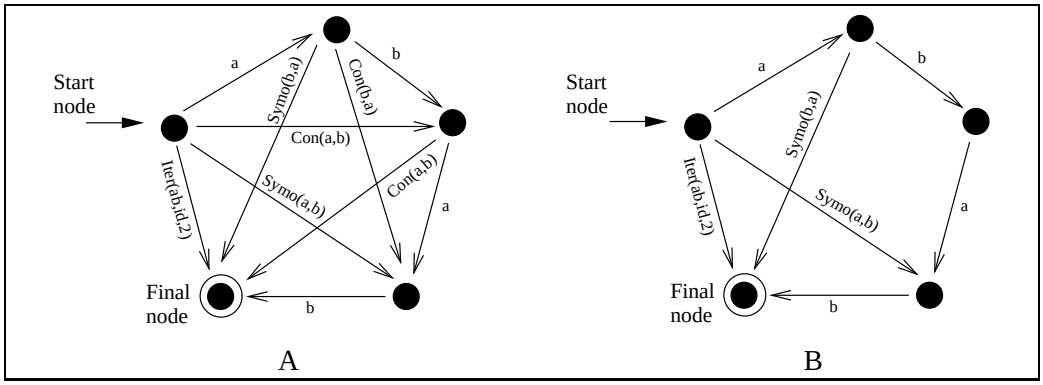


Figure 3: The complete top-level graph (A) and its reduced graph (B) of the pattern "abab"

Notice that this approach envisages a recursive structure between graphs: applied once, the algorithm gives the structural organization at the highest level only. For example, if the pattern is *aabbccaabbcc*, then the shortest path in its graph will be a single link corresponding to the gestalt $Iter(aabbcc, id, 2)$. But on a recursive application of the algorithm to *aabbcc*, we can find a preferable gestalt for it, namely $Iter(aa, succ, 3)$; and on a further application on *aa* we will find $Iter(a, id, 2)$; resulting in the final gestalt $Iter(Iter(Iter(a, id, 2), succ, 3), id, 2)$.

6.3 An Algorithm to Solve Proportional Analogies

We now present an algorithm to solve proportional analogies in our algebraic framework. This algorithm can be seen as an extension of Van der Helm and Leeuwenberg's algorithm to incorporate the mutual contextualization effect and the projectability constraint (as explained in Section 5).

Let us assume that the perceptual patterns A, B, and C of the proportional analogy problem $A : B = C : X$ are given, and the goal is to find the pattern X. We will utilize the simplicity and

[2]In applying a local homomorphism to a term, all the elements of the term that are mapped by the local homomorphism are replaced by the mapped elements.

projectability criteria to direct the search process among possible gestalts of the given patterns, determine the appropriate representation algebras, and finally decide the fourth pattern X. The top-level structure of our algorithm is as follows:

1. For each of the perceptual patterns A , B, and C separately, generate the set of possible gestalts (algebraic terms).

2. Order triples of gestalts for A, B, and C according to their collective information load. This yields a list of sets of triples where elements of each set have the same collective information load.

3. Test for the Simplicity Condition: Iterate through the list of sets of triples in the order of increasing information load until the end of the list is reached; and for each set do:

   (a) Pick an arbitrary triple from the selected set.

   (b) Test the triple for Projectability Condition: Check if there are two minimal representation algebras such that:

      i. One representation algebra $\mathcal{S}$ that can generate the gestalts for A and B.

      ii. One representation algebra $\mathcal{T}$ that can generate the gestalt for C.

      iii. A local homomorphism M between $\mathcal{S}$ and $\mathcal{T}$ exists which maps A to C.

   (c) If the end of the list is reached and no correspondence is found, return Fail.

4. Apply the local homomorphism M to the gestalt of B (in the triple that satisfied the projectability condition in the previous step) to get a gestalt for X.

A flow chart for the algorithm is shown in Figure 4.

In the first step of this algorithm, three sets S_1, S_2, and S_3 that contain algebraic terms for perceptual patterns A, B, and C, respectively, are generated. Then, from these three sets, the triples having the lowest collective information load are computed and put in the set *Set-of-triples*. Note that *Set-of-triples* is a subset of the Cartesian product of the sets S_1, S_2, and S_3. We compute *Set-of-triples* by keeping a variable *Curr-min-load* that is initialized to be the sum of the lowest information load gestalts in S_1, S_2, and S_3. All triples that have their collective information load equal to *Curr-min-load* are put into *Set-of-triples*. Then we increment *Curr-min-load* so that the set of triples with the next higher information load can be computed (if necessary.)

In the next step, a triple is randomly selected from *Set-of-triples*, and the projectability criterion is applied to it. If the triple is not projectable, then this module returns 'Fail'. Otherwise, it returns a representation algebra $\mathcal{S}$ that can generate the gestalts for the terms A and B (the first two gestalt in the triple), a representation algebra $\mathcal{T}$ that can generate the gestalt for the term C (the third gestalt in the triple), and a local homomorphism F from $\mathcal{S}$ to $\mathcal{T}$. The details of this algorithm for testing projectability are presented in Section 6.4.

If the projectability condition is satisfied for the selected triple, the fourth term of the analogy is generated by applying the local homomorphism F to the gestalt of the second term B. But if the projectability condition fails, then we select another triple from *Set-of-triples*, and repeat the procedure. If *Set-of-triples* becomes empty without yielding any projectable triple, we compute a new *Set-of-triples* using the incremented *Curr-min-load*. This procedure is repeated until a projectable triple is found.

Notice that we select the possible triples first according to the simplicity criterion and then according to the projectability criterion. Obviously, the inverse order of applying these criteria is also possible but would be very inefficient. Moreover, our algorithm incorporates the assumption that those descriptions that represent the most salient features of patterns have the best chance of being the intended descriptions within the context of the proportional analogy.

Note that if the context-free lowest information load descriptions of the patterns are not appropriate for the proportional analogy, then our algorithm would choose different descriptions. Therefore, our model can construct creative as well as non-creative analogies Indurkhya (1992).

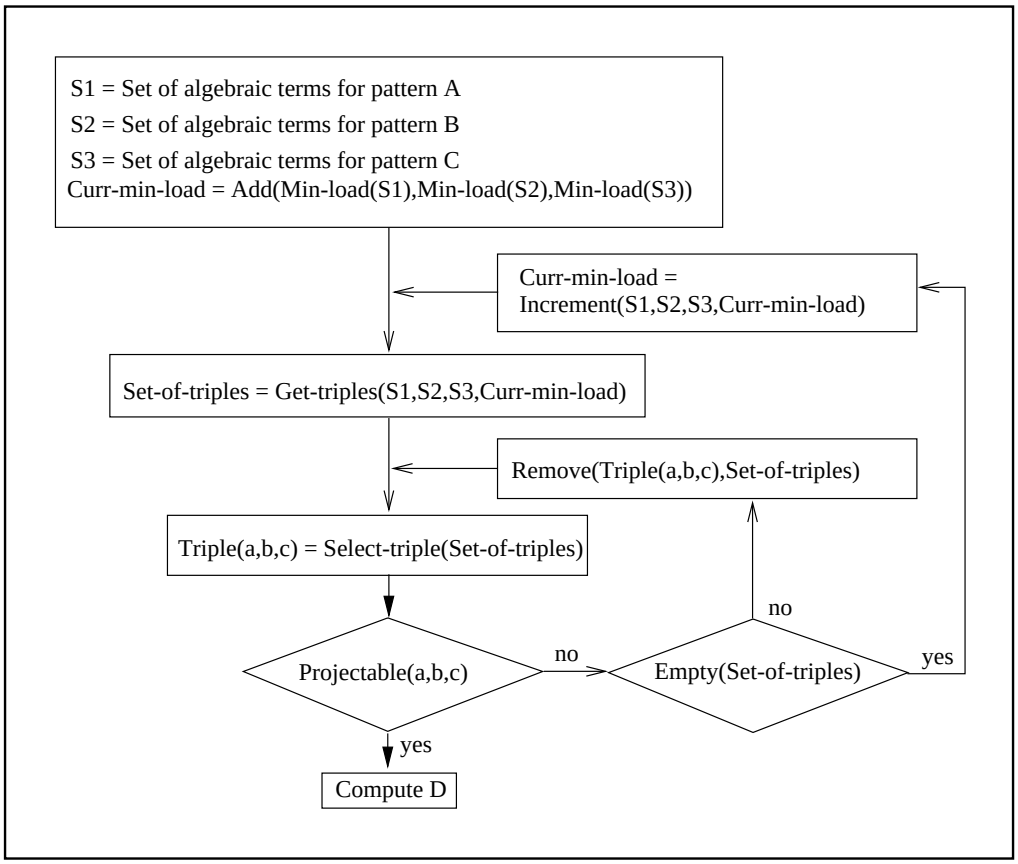


Figure 4: *The top-level structure of the algorithm for solving proportional analogy problems.*

6.4 An Algorithm for Testing the Projectability Condition

As mentioned above, a triple of gestalts (t_1, t_2, t_3) is considered to be projectable if we can find a representation algebra that generates t_1 and t_2, a representation algebra that generates t_3, and a local homomorphism from the first algebra to the other such that it maps t_1 into t_3. Moreover, in keeping with the constraints introduced in Section 5, we would like to keep the complexity of the two algebras and of the mapping to a minimum. In this section we present a heuristic algorithm for generating the two algebras and the mapping. The algorithm returns 'Fail' if the triple is not projectable.

As both representation algebras are subalgebras of the SIT algebra $< \mathcal{U}, \mathcal{F} >$, our algorithm focuses on finding a local homomorphism $M =< \Delta, \Omega >$ that maps t_1 to t_3. The two representation algebras are then simply obtained by taking the closure of the domain and the co-domain of $< \Delta, \Omega >$. The algorithm, a function called 'projection', starts by initializing Δ and Ω both to be empty sets. It then recursively examines the hierarchical structure of the three gestalts t_1, t_2 and t_3, starting from the outermost operator, and gradually adds the necessary elements to Δ and Ω.

In examining the outer structure of the gestalts t_1, t_2 and t_3, we distinguish between four cases, each of which is shown in Figure 5. In the cases shown in Figures 5(a) and 5(b), all the three gestalts are primitive terms, meaning that they all are either strings (i.e. belong to $D = \{a, abd, a(bc), \ldots\}$) or domain-dependent operators (i.e. belong to $F_D = \{succ, pred, \ldots\}$ or natural numbers (i.e. belong to $\mathcal{N} = \{\infty, \in, \ldots\}$). In the case shown in Figure 5(a), t_1 and t_2 are identical. In this case we increment Δ by adding the mapping $[t_1, t_3]$ to it. In the case shown in Figure 5(b), t_1 and t_3 are identical, and we add the two identity elements $[t_1, t_1]$ and $[t_2, t_2]$ to Δ.

In the cases shown in Figures 5(c) and 5(d), all three gestalts have some SIT operator at the outermost level. In the case shown in Figure 5(c), t_1 and t_2 have the same outermost operator,

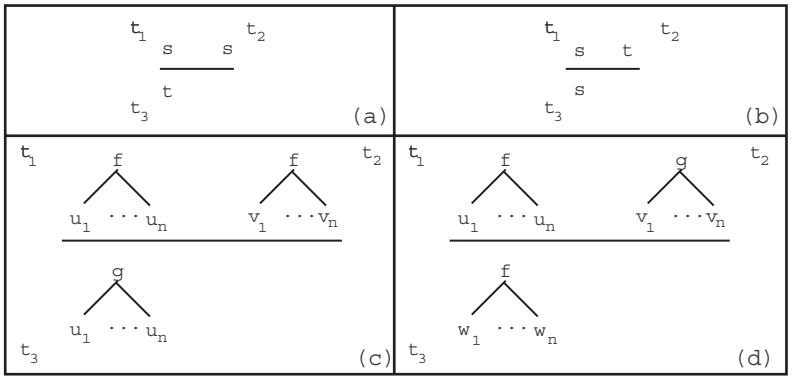


Figure 5: *Four cases for the projection algorithm.*

say f, and t_3 has a different outermost operator, say g, but of the same arity as f. Moreover, t_1 and t_3 have the same arguments to the outermost operator, say $u_1, \ldots, u_n$, assuming the arity of f and g to be n. In this case, we increment Ω by adding the mapping $[f, g]$ to it, and increment Δ by adding $[u_i, u_i]$, $[v_i, v_i]$, for $i = 1, \ldots, n$, assuming that $v_1, \ldots, v_n$ are the arguments to the outermost operator of t_2.

The fourth case shown in Figures 5(d) is recursive. Here, t_1 and t_3 have the same outermost operator, say f, and t_2 has a different outermost operator, say g, but of the same arity as f (say n.) The arguments to the outermost operators are all different. In this case, we call the projection function recursively with the triples formed by picking the arguments at the same positions of the outermost operators of t_1, t_2 and t_3. In other words, if the first arguments to the outermost operators of t_1, t_2 and t_3 were u_1, v_1 and w_1, respectively, then we make a recursive call to the projection function with (u_1, v_1, w_1); and so on for the second arguments, the third arguments, etc. All these recursive calls, would return us some mappings, say $M_1 =< \Delta_1, \Omega_1 >; \ldots;\ M_n =< \Delta_n, \Omega_n >$. Then $\Delta_1, \ldots, \Delta_n$ are joined together to get Δ, and $\{[f, f], [g, g]\}$, $\Omega_1, \ldots, \Omega_n$ are joined together to get Ω, making sure that the condition of homomorphism (Definition 9) is not violated. In all other cases, the projection function ends in a failure and returns 'fail'. The specification of the Projection function and its implementation is shown in Figure 6. Note that in this algorithm we do not follow the algebraic closure property to generate all possible objects. In fact, we generate only those objects that are needed to construct a mapping between the given terms.

The repeat loop in the algorithm implements the recursive case shown in Figure 5(d). The algorithm calls itself recursively by passing the arguments in the i^{th} positions of each of three terms. The returned value is accumulated in the variable M using a function called *Join,* which joins the objects and operators of two given mappings into a single mapping, making sure that the partial local homomorphism condition is satisfied, or else returns 'Fail'. Whenever the Join function returns 'Fail', the loop is immediately terminated.

In combining two mappings with the Join function, because the mappings are added incrementally, we can assume that the first mapping is already a partial local homomorphism. Then what remains to be seen is that adding the second mapping (1) does not make it so that a single element (an object or an operator) is mapped to two different elements, and (2) does not violate the operational structure of the two algebras. Though the first of these condition is easily checked, the second one requires the operators in the combined mapping be applied to the objects in the combined mapping to see that the algebraic structure is preserved. However, here also the fact that the mappings are added incrementally can be used to develop a more efficient algorithm. Figure 7 shows the specification and implementation of the Join function. Here M_1 and M_2 are two mappings that are known to be partial local homomorphisms. If M_1 and M_2 can be combined into a partial local homomorphism, then their union is returned as M, otherwise 'Fail' is returned.

Note that in this algorithm Ω is applied to Δ and it is checked if the local homomorphism

/* t_1, t_2 and t_3 are terms of the gestalt algebra for A, B and C, respectively. M is a local homomorphism that is returned by this function. */

Function : Projection (t_1, t_1, t_3): return $M =< \Delta, \Omega >$

Projection(t_1, t_2, t_3)
begin

/* Check to see if it is the case shown in Figure 5(a) */

if $t_1, t_2, t_3 \in D$, or $t_1, t_2, t_3 \in F_D$, or $t_1, t_2, t_3 \in \mathcal{N}$, then,
if $t_1 = t_2$, then return $M =< \{[t_1, t_3]\}; \emptyset >$,

/* Check to see if it is the case shown in Figure 5(b) */

else if $t_1 = t_3$, then return $M =< \{[t_1, t_1], [t_2, t_2]\}; \emptyset >$,
else return 'Fail'

/* Check to see if it is the case shown in Figure 5(c) */

else if t_1 is of the form $f(u_1, \ldots, u_n)$,
t_2 is of the form $f(v_1, \ldots, v_n)$, and
t_3 is of the form $g(u_1, \ldots, u_n)$, then
return $M =< \{[u_1, u_1], \ldots, [u_n, u_n], [v_1, v_1], \ldots, [v_n, v_n]\}; \{[f, g]\} >$,

/* Check to see if it is the case shown in Figure 5(d) */

else if t_1 is of the form $f(u_1, \ldots, u_n)$,
t_2 is of the form $g(v_1, \ldots, v_n)$, and
t_3 is of the form $f(w_1, \ldots, w_n)$, then
$M =< \emptyset, \{[f, f], [g, g]\} >$
$i = 1$
repeat
$M =$ Join$(M,$ Projection$(u_i, v_i, w_i))$
$i = i + 1$
until ($M =$'Fail') or ($i > n$)
return M
else return 'Fail'.
end;

Figure 6: The algorithm that computes the Projection function.

/* M_1 and M_2 are two partial local homomorphisms. M_1 and M_2 are combined into a partial local homomorphism M. */

Function: Join ($M_1 =< \Delta_1, \Omega_1 >$, $M_2 =< \Delta_2, \Omega_2 >$):
return $M =< \Delta, \Omega >$

Join($M_1 =< \Delta_1, \Omega_1 >, M_2 =< \Delta_2, \Omega_2 >$)
begin

/* Check that the combination of two mappings is still a function and remove repetitions. */

For each pair $[x, y] \in \Delta_2$, if there exists a pair $[x, y'] \in \Delta_1$ then,
If $y = y'$ then remove $[x, y]$ from Δ_2,
else return 'Fail'.
For each pair $[x, y] \in \Omega_2$, if there exists a pair $[x, y'] \in \Omega_1$ then,
If $y = y'$ then remove $[x, y]$ from Ω_2,
else return 'Fail'.
Set $\Delta = \Delta_1 \cup \Delta_2$,
Set $\Omega = \Omega_1 \cup \Omega_2$,

/* Check to see that the combination of two mappings is a partial local homomorphism. */

For each $[f, g] \in \Omega_1$, and for each $[x, y] \in \Delta_1$,
if there is some y' such that $[f(x), y'] \in \Delta_2$ but $y' \neq g(y)$
then return 'Fail'.
For each $[f, g] \in \Omega_2$, and for each $[x, y] \in \Delta_1$,
if there is some y' such that $[f(x), y'] \in \Delta$ but $y' \neq g(y)$
then return 'Fail'.
For each $[f, g] \in \Omega_1$, and for each $[x, y] \in \Delta_2$,
if there is some y' such that $[f(x), y'] \in \Delta$ but $y' \neq g(y)$
then return 'Fail'.
For each $[f, g] \in \Omega_2$, and for each $[x, y] \in \Delta_2$,
if there is some y' such that $[f(x), y'] \in \Delta_1$ but $y' \neq g(y)$
then return 'Fail'.
Return $M =< \Delta, \Omega >$.
end;

Figure 7: The algorithm that computes the Join function.

condition is violated. But because $< \Delta_1, \Omega_1 >$ is already known to be a local homomorphism, we can reduce some of the tests. So, when Ω_1 is applied to Δ_1, we don't need to check for it in Δ_1, but only in Δ_2 (which are new elements that are being added.) Similarly, when Ω_2 is applied to Δ_2, we don't need to check for it in Δ_2, but only in Δ_1. But when we apply Ω_1 to Δ_2, we need to check in both Δ_1 and Δ_2 (hence Δ), and when we apply Ω_2 to Δ_1, we need to check in both Δ_2 and Δ_1 (hence Δ).

Regarding the constraints mentioned in Section 5, namely that the complexity of the two algebras and the mapping should be kept at a minimum, we would like to note that our incremental approach starts with empty algebras and adds elements only as necessary to generate the given gestalts. Similarly, the mapping is also started an empty set, and pairs of elements are added only when necessary and as far as possible identically. This turns our approach into a kind of *greedy algorithm.* This is essentially a heuristic approach, which does not always guarantee an optimal solution.

6.5 Two Examples

In this section, we present two proportional analogy problems, and show their underlying homomorphisms as computed by our projection algorithm of the last section.

Consider the proportional analogy problem "$abc : abcd = zyx : X$". If we consider $Iter(a, succ, 3)$, $Iter(a, succ, 4)$, and $Iter(z, pred, 3)$ to be the gestalts for the first three terms, respectively, then the local homomorphism induced by them is obtained as shown in Figure 8.

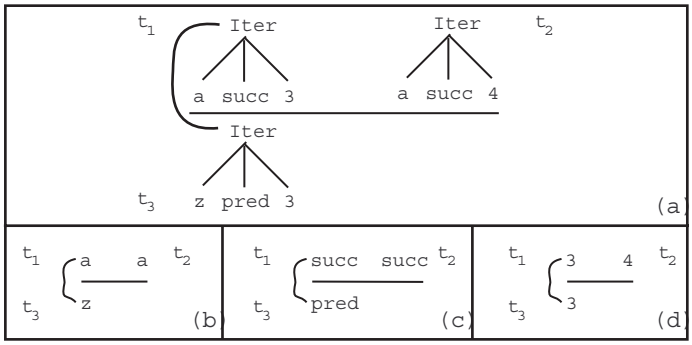


Figure 8: *An example of a mapping created by the projection algorithm.*

In the first step, the outer structures of the gestalts are compared as shown in Figure 8(a), to which the case of Figure 5(d) applies. As a result, the local homomorphism, initialized to be $[\emptyset, \emptyset]$, is set to $< \emptyset, \{[\text{iter}, \text{iter}]\} >$, and the projection function is recursively called on three triples formed by taking the arguments in the same position from each of the three terms, as shown in Figures 8(b), (c), and (d). The situations in Figures 8(b) and (c) correspond to the case shown in Figure 5(a), and the one in Figure 8(d) corresponds to Figure 5(b). Thus, the final value of the local homomorphism becomes $< \{[a, z], [3, 3], [4, 4]\}, \{[iter, iter], [succ, pred]\} >$. Note that this homomorphism maps the second element of the proportional analogy problem (i.e. $abcd$) to $zyxw$ which is considered as the solution (i.e. the fourth term) of the problem.

As a second example, consider the proportional analogy problem "$abc : abd = iijjkk : X$". Assume that the selected gestalts are $Con(ab, id(c))$, $Con(ab, succ(c))$, and $Con(iijj, id(kk))$, respectively. The local homomorphism induced by these gestalts for A, B and C is recursively obtained as shown in Figure 9.

As in the last example, the outer structure of the three terms, shown in Figure 9(a), corresponds to Figure 5(d), and so the local homomorphism is set to $< \emptyset, \{[\text{con}, \text{con} >\}]$, and the projection function is called with the two sets of triples formed by taking the first and the second

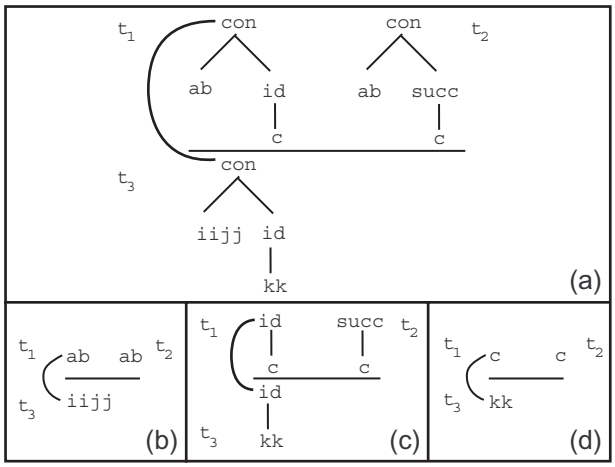


Figure 9: *Another example of a mapping created by the projection algorithm.*

arguments, respectively, from each of the three terms. The first of these calls, shown in Figure 9(b), corresponds to the case shown in Figure 5(a), and contributes the pair $[ab, iijj]$ to the local homomorphism. The second call, shown in Figure 9(c), corresponds again to Figure 5(d), and makes another recursive call to the projection function with the set of triples formed by the single argument of each of the three terms. At this point, the value of the local homomorphism is $< \{[ab, iijj]\}, \{[\text{con}, \text{con}], [\text{id}, \text{id}], [\text{succ}, \text{succ}]\} >$. In the last step, shown in Figure 9(d), which corresponds to the case shown in Figure 5(a), the element $[c, kk]$ is added to the local homomorphism, resulting in a final value of $< \{[ab, iijj], [c, kk]\}, \{[con, con], [id, id], [succ, succ]\} >$. Note that this homomorphism maps the second element of the proportional analogy problem (i.e. abd) to $iijjll$ which is considered as the solution (i.e. the fourth term) of the problem.

6.6 A Limitation

We would like to point out here that there is an assumption built into our formalism: namely that the same object occurring at different places in an algebraic term is the same. But there are proportional analogies that do not satisfy this assumption. For example, consider the algebraic term $Con(Iter(Con(a, b), id, 2), a)$ which is a description of the first term of the proportional analogy $ababa : abbaa = cdcdg : cddcg$. In order to construct a reasonable analogical mapping for this example, one must distinguish between the two occurrences of the letter "a" in the mentioned algebraic term. In fact, the first "a" should be mapped to "c" and the second "a" should be mapped to "g".

This is a general phenomenon which exists for proportional analogies involving visual patterns as well. In order to cover such analogies, one needs a mechanism to distinguish different occurrences of identical elements. In our formalization, one solution is to assign indices to the elements that occur at different places in algebraic terms: such indices can be generated on the basis of the positions of the elements in the algebraic term. For example, we may rewrite the above algebraic term as $Con_{1,1}(Iter_{2,1}(Con_{3,1}(a_{4,1}, b_{4,2}), id, 2), a_{2,2})$. Notice that the two occurrences of the letter "a" can now be distinguished. The assignment of indices to algebraic terms and the construction of algebraic correspondences are further elaborated by Dastani (1998).

7 Conclusions and Future Research Issues

In this article, we have outlined an algebraic approach to modeling a process by which new gestalts of a perceptual stimulus can emerge in the context of proportional analogy. Our approach extends Leeuwenberg's structural information theory in two significant ways. One is to embed it in an

algebraic framework so that regularities based on domain-dependent relations are allowed to play a role in structural descriptions. The other is to modify the minimality principle so that mutual contextualization effect can be modeled. That is, when two (or more) patterns are presented together, we prefer those perceptual gestalts for the patterns that minimize the overall complexity; the gestalts of the patterns that minimize their individual complexity in isolation do not always result in the minimum overall complexity. This effect is modeled by introducing the concept of *representation algebras,* which generate a class of patterns, and defining a measure of complexity on them. Then mutual contextualization is modeled by examining representation algebras for all the patterns that are to be considered together, and choosing one that is minimal and also minimizes the complexity of all the patterns together.

Then we also formalize the notion of analogical mapping in our algebraic framework, and show how all these constraints can be integrated in an algorithm for solving proportional analogy problems. We claim that our algorithm is superior to other approaches to modeling proportional analogies in that we are able to incorporate the mutual contextualization effect, and the principles underlying our algorithm are clearly and formally explicated.

We would like to make some remarks now comparing our algorithm to the other existing approaches to solving proportional analogy problems. In the early days of artificial intelligence research, Evans (1968) implemented a system for solving proportional analogy problems in the geometric figures domain. However, in Evans' system, the representations of the figures (terms of the analogy relation) were determined first, and then the mappings were computed. Though Evans explicitly discusses the mutual contextualization effect, his system did not yet model it. Moreover, even the context-independent gestalts for individual figures were computed in a somewhat *ad hoc* fashion in Evans' system. In our system, on the other hand, our main goal has been to model the contextualization effect, and also because our system is based on the structural information theory, for which a considerable empirical support has been found, we feel that it is much less *ad hoc.*

More recently, Mitchell (1993) implemented the Copycat system, which was expressly designed to model the creativity phenomenon in proportional analogies as we discussed in the introduction. Consequently, in Copycat, representations of the terms are constructed hand in hand with the mappings, and thus the mutual contextualization effect is fully taken into account. However, many of the features of Copycat are not clearly, or formally, specified, so it is not clear how its approach can be applied to other domains. For example, consider the concept of *temperature,* that plays a key role in the Copycat architecture. Intuitively, the idea is that the 'deeper' or more cognitively appealing an analogy, the lower its temperature (which is based on an analogy with thermodynamics). However, nowhere in the Copycat architecture, or in its discussion, one finds any principles or rules or any explicit description for computing the temperature of an analogy relation. In fact, as the concept of information load in *SIT-algebra* can be considered analogous to Copycat's temperature, this reveals starkly the contrast between our two approaches, for the focus of our research has been on explicating the principles that constrain the gestalts of a pattern, both in isolation and in context.

Another point to emphasize is that our algebraic model is aimed at modeling only the input-output functionality of the human perceptual process. That is, we do not claim that people carry algebraic descriptions in their heads, or that the algorithm presented in Section 6 mirrors in anyway how humans solve proportional analogy. By contrast, Copycat implicitly claims to model the human perceptual processes.

Needless to say, there are many other open issues that still need to be investigated, and we would like to mention a few of them here. The algorithm presented here provides one way to integrate the three constraints on analogy. We need to develop and study other possible computational models based on the theory.

Secondly, structural information theory, including the extension of it presented here, considers only one-dimensional patterns. While the coding system of Leeuwenberg allows some two-dimensional regularities to be captured in one-dimensional patterns, there are many other regularities that cannot be so captured. We are currently working on generalizing SIT in this respect Dastani (1998).

Finally, we would like to be able to model cognitive operations of grouping as well, which will allow us to explain how creative insights occur in cognitive domains. One cognitive domain where analogies play a key role, and where creative aspects of analogy can be glimpsed, is legal reasoning Indurkhya (1997). At this point, however, it remains an open issue whether and how the model of gestalt perception and disambiguation that we outlined in this paper would apply to creativity in cognitive domains such as legal reasoning. This, nevertheless, remains our long-term research goal.

References

Baxter, L. (1977). The complexity of unification. Technical Report Internal Report CS- 77-25, University of Waterloo, Faculty of Mathematics, Ontario, Canada.

Cohn, P. (1981). Universal algebra. Revised edition, D.Reidel, Dordrecht, The Netherlands.

Dastani, M. (1998). *Languages of Perception.* Ph.D. thesis, Institute of Logic, Language, and Computation (ILLC), The Netherlands.

Evans, T. (1968). A program for the solution of a class of geometric-analogy intelligence-test questions. In Minsky, M., editor, *Semantic Information Processing*, pages 271–353, Cambridge, Mass. MIT Press.

Hofstadter, D. (1984). The copycat project: An experiment in nondeterminism and creative analogies. In *A.I. Memo 755, Artificial Intelligence Laboratory.* MIT, Cambridge, Mass.

Hofstadter, D. (1995). *The Fluid Analogies Research Group.* Basic Books, New York.

Indurkhya, B. (1991). On the role of interpretive analogy in learning. *New Generation Computing*, 8:385–402.

Indurkhya, B. (1992). *Metaphor and cognition: an interactionist approach.* Kluwer Academic Publishers, Dordrecht, The Netherlands.

Indurkhya, B. (1997). On modeling creativity in legal reasoning. In *Proceedings of the Sixth International Conference on AI and Law*, pages 180–189, Melbourne, Australia.

Koffka, K. (1935). *Principles of Gestalt Psychology.* Harcourt, Brace and World, New York.

Leeuwenberg, E. (1971). A perceptual coding language for visual and auditory patterns. *American Journal of Psychology*, 84:307–349.

Mitchell, M. (1993). *Analogy-Making as Perception.* Bradford Books/MIT Press, Cambridge, Mass.

Siekmann, J. (1989). Unification theory. *Journal of Symbolic Computation*, 7:207–274.

Van der Helm, P. and Leeuwenberg, E. (1986). Avoiding explosive search in automatic selection of simplest pattern codes. *Pattern Recognition*, 19:181–191.

Van der Helm, P. and Leeuwenberg, E. (1991). Accessibility: A criterion for regularity and hierarchy in visual pattern code. *Journal of Mathematical Psychology*, 35:151–213.

Wertheimer, M. (1923). Untersuchungen zur lehre von der gestalt. *Psychologische Forschung*, 4:301–350.

The Concept of Minimal 'Energy' Change (MEC) in Relation to Fourier Transform, Auto-Correlation, Wavelets, AMDF, and brain-like timing networks – Application to the Recognition of Repetitive Rhythmical Patterns in Acoustical Musical Signals*

Marc Leman and Bart Verbeke
IPEM - Dept. of Musicology, Ghent University
Blandijnberg 2, B-9000 Ghent, Belgium
Marc.Leman@rug.ac.be

Abstract

Repeating patterns have a more or less constant 'energy' over the period of the repeating pattern. Minimal changes of this 'energy' (MEC) in general point to the period of the repetitive pattern. This paper gives a thorough analysis of the MEC-concept and algorithm and shows that a MEC function is common to the Fourier transform, the auto-correlation, and wavelet transform. MEC provides an interpretation of the average magnitude difference function (AMDF) and can be related to brain-like computation in timing networks. The MEC-concept is applied to the detection of repetition in rhythm patterns. Based on its energy, the kind of rhythm pattern detected is taken to be an image of expressive body movement rather than division of time.

1 Introduction

In this paper we discuss an approach to repetitive pattern analysis based on the concept of minimal (energy) change (MEC). Although we use the acronym MEC to denote this concept, it will become clear that our interpretation of 'energy' is rather tolerant. 'Energy' may point to a quantity such as intensity, magnitude, probability of neuronal discharge, or derived quantities. The concept of MEC is furthermore general as it provides a concept to deal with repetition in patterns. It can be applied to different perceptual domains but we restrict here our application to the rhythm domain.

We first introduce the concept of MEC, and show its connection with classical signal processing techniques such as Fourier transformation, auto-correlation, wavelets, and the average magnitude difference function (AMDF), and timing networks. MEC is then applied to repetitive pattern recognition in the rhythmical/gesture domain. A final part is dedicated to discussion.

2 The MEC-concept

The basic idea behind MEC is (i) that repeating patterns have a more or less constant energy over the period of the repeating pattern and (ii) that minimal changes of this energy point to the period of the repetitive pattern. We analyse the implications of this idea first in the discrete time domain and then in the continuous domain.

*An extended version of this paper is submitted to Journal of New Music Research, Special Issue on Rhythm Perception, Periodicity, and Timing Nets. Sound examples of original and resynthesized sounds are available at http://www.ipem.rug.ac.be/staff/marc/sounds.

2.1 Continuous Domain

We denote $s(t)$ as the *energy* signal, with the domain of periods τ as given. For different τ we want to determine the minimum change of the 'energy' values. The 'energy' $E(t,\tau)$ of $s(t)$ over period τ is defined as:

$$E(t,\tau) \;=\; \int_{t'=t}^{t+\tau} s(t')\; dt' \tag{1}$$

The partial derivate of the energy, in function of t, gives the change of energy over time. It gives:

$$\begin{aligned} \frac{\partial}{\partial t}E(t,\tau) \;&=\; \int_{t}^{t+\tau} \frac{\partial s(t')}{\partial t'} dt' \\ &=\; s(t+\tau) - s(t) \end{aligned} \tag{2}$$

From this function, we want to find the values that come close to zero. Our option is to take the minimum of the absolute value of this function over all τ (greater than zero), which then gives the signal period:

$$\begin{aligned} p(t) \;&=\; min_{\tau>0}\; \left(\left|\; \frac{\partial}{\partial t}E(t,\tau) \;\right| \right) \\ &=\; min_{\tau>0}\; (\; |\; s(t+\tau) - s(t) \;|\;) \end{aligned} \tag{3}$$

$p(t)$ contains for fixed t the index τ for which $|\frac{\partial}{\partial t}E(t,\tau)|$ at this fixed t is minimal. In order to make $p(t)$ less vulnerable to local variations, the values can first be integrated with the previously obtained derivatives. This gives:

$$p(t) \;=\; min_{\tau>0} \left(\int_{0}^{t} \;|\; s(t'+\tau) - s(t') \;|\; dt' \right) \tag{4}$$

In practice, however, it is more appropriate to add an exponential function such that the most recent values get a higher weight:

$$p(t) \;=\; min_{\tau>0} \left(\int_{0}^{t} \;|\; s(t'+\tau) - s(t') \;|\; e^{\beta(t-t')}\; dt' \right) \tag{5}$$

The parameter β defines the degree of the past influence on the present. Normally, we take $\beta < 0$ in order to enhance the more recent found values ($\beta > 0$ would give more weight to the older periods found).

3 Derivation of the MEC-algorithm from Classical Methods

In the previous sections, one could argue that the energy concept does not longer appear in our final MEC function (Expression 5) and that it is rather straightforward to first define the energy as the integral and then differentiate to come to the conclusion where we started from, namely that a signal is repetitive when the signal values at its period approach zero. In this section, however, we show that the MEC-concept, or minimalization of 'energy' change, is more fundamental in that it underlies the classical signal processing approaches. In particular, we show that the application of the MEC-concept to the Fourier transformation, the auto-correlation, and wavelets brings us back to the core observations made in the previous section, and that these observations provide an interpretation of the average magnitude difference function, which is strongly related to the MEC function derived in the previous section.

3.1 Derivation from Fourier Transformation

The short-term Fourier transform finds the period of the signal by a mapping of the original signal onto elementary signals. In what follows, we show that Expression 3 can be derived from the short-time Fourier transform through a minimalization of the obtained 'energy' (which, in the present case, is the magnitude).

The short-time Fourier transform is characterized as:

$$S(t,f) \;=\; \int_{-\infty}^{+\infty} s(t')\; h(t'-t)\; e^{-j2\pi f t'} dt' \tag{6}$$

where $h(t'-t)$ is a window function. We assume that the window is rectangular and obeys the condition that

$$h(t'-t,f) \;=\; \begin{cases} 1 \text{ if } |t'-t| & \leq \frac{\tau}{2} \\ 0 \text{ if } |t'-t| & > \frac{\tau}{2} \end{cases} \tag{7}$$

where $f = \frac{1}{\tau}$. This condition implies a modification of the 'classical' short-time Fourier transform in the sense that the window is dependent on τ, which makes it more similar (though not equal) to a wavelet transform. Then we can rewrite Expression 6 as:

$$S(t,f) \;=\; \int_{t}^{t+\tau} s(t')\; e^{-j2\pi f t'}\; dt' \tag{8}$$

In applying the MEC concept, we are interested in how $S(t,f)$ changes over time, so we calculate:

$$\frac{\partial}{\partial t} S(t,f) \;=\; \int_{t}^{t+\tau} \frac{\partial}{\partial t'} \left[s(t')\; e^{-j2\pi f t'} \right]\; dt' \tag{9}$$

which gives:

$$\frac{\partial}{\partial t} S(t,f) \;=\; s(t+\tau)\; e^{-j2\pi f(t+\tau)} \;-\; s(t)\; e^{-j2\pi f t} \tag{10}$$

Given the fact that $f = \frac{1}{\tau}$, and $e^{-j2\pi f(t+\tau)} \;=\; e^{-j2\pi f t}$ this can be further reduced to:

$$\frac{\partial}{\partial t} S(t,\tau) \;=\; [s(t+\tau) \;- s(t)]\; e^{-j2\pi \frac{t}{\tau}} \tag{11}$$

We reduce this to the magnitude (or 'energy') part in which we are interested. Minimizing the magnitude of the partial derivative of the Fourier transform for all τ then gives the period or frequency of the most repetitive pattern so that we obtain:

$$min_{\tau>0} \left(\left| \frac{\partial}{\partial t} S(t,f) \right| \right) \;=\; min_{\tau>0} \;(\; |s(t+\tau) \;- s(t)| \;) \tag{12}$$

This is similar to Expression 3 which defines the MEC function.

3.2 Derivation from Auto-Correlation

In a similar way, we obtain the basic MEC function by applying the notion of minimal 'energy' change to the auto-correlation. We define the short-term (circular) auto-correlation on the energy signal $s(t)$ as:

$$\begin{aligned} C(t,\tau,\delta) \;=\; & \int_{t}^{t+\tau} s(t')s(t'+\delta)\; dt', \\ & \text{where } 0 \leq \delta < \tau \\ & \text{and, given circularity,} \\ & s(t+\delta) \;=\; s(t+\tau+\delta) \end{aligned} \tag{13}$$

Auto-correlation takes a copy of a (windowed) signal $s(t)$, shifts this copy over a time-lag δ, multiplies the values, and calculates the sum. In the circular auto-correlation the shifted signal is rotated (circular). In this definition we take the window length equal to τ such that $s(t + \delta) = s(t + \tau + \delta)$.

In considering changes of $C(t, \tau, \delta)$, we investigate the partial derivation with respect to t, which gives:

$$\begin{aligned} \frac{\partial}{\partial t} C(t, \tau, \delta) &= \int_{t}^{t+\tau} \frac{\partial}{\partial t'} \left[s(t') s(t' + \delta) \right] dt' \\ &= s(t + \tau) s(t + \tau + \delta) - s(t) s(t + \delta) \end{aligned} \tag{14}$$

which, given the equality of $s(t + \delta) = s(t + \tau + \delta)$, can be further reduced to:

$$\frac{\partial}{\partial t} C(t, \tau, \delta) = \left[s(t + \tau) - s(t) \right] s(t + \delta) \tag{15}$$

We are interested in values that come close to zero, so we minimize the absolute values of this function with respect to τ (greater than zero), which gives:

$$min_{\tau>0} \left(\left| \frac{\partial}{\partial t} C(t, \tau, \delta) \right| \right) = min_{\tau>0} \left(\left| \left[s(t + \tau) - s(t) \right] s(t + \delta) \right| \right) \tag{16}$$

and since $s(t+\delta)$ remains the same for different τ (provided, of course, that $\delta < \tau$) we can further reduce to:

$$min_{\tau>0} \left(\left| \frac{\partial}{\partial t} C(t, \tau) \right| \right) = min_{\tau>0} \left(\left| s(t + \tau) - s(t) \right| \right) \tag{17}$$

which, again, is similar to Expression 3.

3.3 Derivation from Wavelet Transformation

A Fourier transformation decomposes a signal using a set of basic sinusoids. The wavelet transformation uses a set of functions with varying frequency. We define the wavelet transformation as:

$$W(t, a) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} s(t') \Psi \left(\frac{t' - t}{a} \right) dt' \tag{18}$$

with

$$\Psi \left(\frac{t' - t}{a} \right) = \begin{cases} \sqrt{a} & \text{if } |t' - t| \leq a \\ 0 & \text{if } |t' - t| > a \end{cases} \tag{19}$$

where $a = \frac{1}{f}$. Then we can write Expression 18 as

$$W(t, a) = \int_{t}^{t+a} s(t') \, dt' \tag{20}$$

which, after derivation and minimalization, again gives Expression 3. Notice that we impose certain conditions on the wavelet window in order to obtain this result.

3.4 Comparison with AMDF, and relationship to Timing Nets

The MEC function turns out to be closely related to the so-called *Average Magnitude Difference Function (AMDF)* which is often proposed as an alternative for the auto-correlation (Rabiner and Schafer, 1978, p. 149). A version of AMDF is often defined as:

$$AMDF(t,\tau) \;=\; \frac{1}{t}\int_{t'=0}^{t} \mid s(t'+\tau) \;-\; s(t') \mid \; dt' \tag{21}$$

which is related to Expression 4. The MEC-concept, however, is based on the notion of minimal change of an 'energy' (a positive value quantity), and given its relationship to the classical signal processing techniques, it provides an interpretation of the classical notion of AMDF. Our interpretation suggests that AMDF is more general than auto-correlation, since it can also be derived from the Fourier and wavelet transform, provided that the signal analyzed is an 'energy' signal.

The MEC-concept can also be used as a short-hand for the timing nets proposed by Cariani 1999. Timing nets work with time-delay lines and coincidence detectors. In the feedforward network proposed by Cariani, two signals are fed into a coincidence array by means of tapped delay lines. If the output along the coincidence lines is summed across time, then the function computed is the cross-correlation function. If the outputs are summed across the delay channels then a convolution function is obtained. Timing nets are further conceived of in terms of loop structures that somehow hold and reinforce the repetitive patterns in memory loops. The computational cost of such a timing net is rather high (Leman et al., 1999). For real-time applications, we believe that the MEC-concept provides a much shorter and more efficient way to obtain a similar result: find the period of the repetition pattern and reinforce it in a memory loop. What is needed on top of the MEC-function, in order to make it more similar to a timing network, is a memory that somehow integrates the patterns found.

4 MEC Analysis

Figure 1 gives a general overview of the MEC-analysis and resynthesis. APM is an auditory peripheral module which decomposes the signal in frequency bands (called auditory channel). RMS is the 'energy' extraction module applied to each auditory channel. MEC extracts the period with minimal 'energy' change in each channel and PEM is the pattern extraction module which uses the RMS-patterns to resynthesize a musical signal which can be compared with the original signal. In the applications that follow, we tend to use the MEC function in its most simple form, without the introduction of additional refinements.

5 MEC Examples

5.1 Ligeti's Presto energico

Figure 2a shows a waveform excerpt of *Presto energico* from the *Musica Ricercate per pianoforte* (1951-53) by G. Ligeti (Bis-CD-53). The piece is characterized by a repeating rhythmic pattern with an irregular beat. The period of this pattern is indicated on the figure. Figure 2b shows the synthesized waveform based on the minimal energy change (MEC) algorithm with $t_0 = 3$ seconds.

The original signal was first transformed into an auditory image . Figure 3 shows ten auditory sub-bands, at 1.5 critical bandwidth from each other, with center frequencies from 215 Hz to 3266 Hz. The auditory images are then transformed into RMS patterns (averages are taken over 20 milliseconds, in steps of 10 milliseconds) (Fig. 4).

The MEC-analysis is applied onto the RMS patterns. Figure 5 shows the curves for each channel. Each curve indicates the length of the period τ found at each t. The precise values of τ are not indicated but the graphs give a general impression of the length of the detected τ as well as its stability over different auditory channels over time. The curves, at each time step t, show the minimal change in 'energy' for varying τ.

Figure 6a shows the representation of rhythm patterns. The resulting patterns are denoted $\tilde{q}(t)$. At each fixed t_0 we get a pattern $\tilde{q}(t_0)$ that is filled with RMS values from $r(t_0)$ to $r(t_0 + m(t_0))$

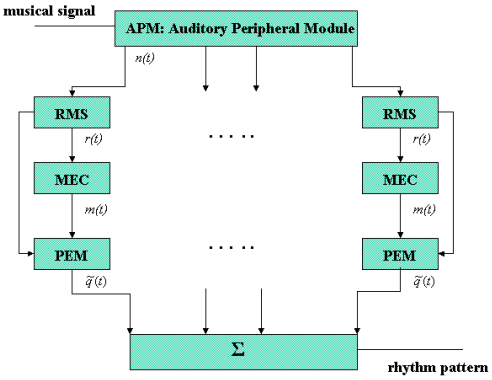


Figure 1: Overview of the MEC-analysis and resynthesis algorithm. APM is the auditory peripheral module, RMS is the 'energy' extraction module, MEC extracts the period with minimal 'energy' change, PEM is the pattern extraction module

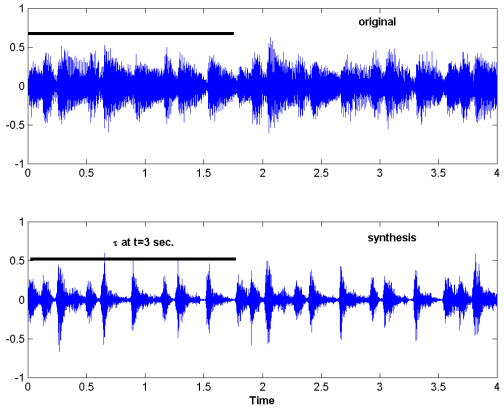


Figure 2: Original soundwave and resynthesised soundwave. (a) Waveform representing an excerpt from the *Presto energico* from the *Musica Ricercate per pianoforte*(1951-53) by G. Ligeti, (b) Synthesised waveform based on the MEC-analysis after 3 seconds. The loop-patterns of the different auditory channels are repeated. The resulting modulation patterns are then multiplied with band-pass noises having the center frequency of the corresponding auditory channels

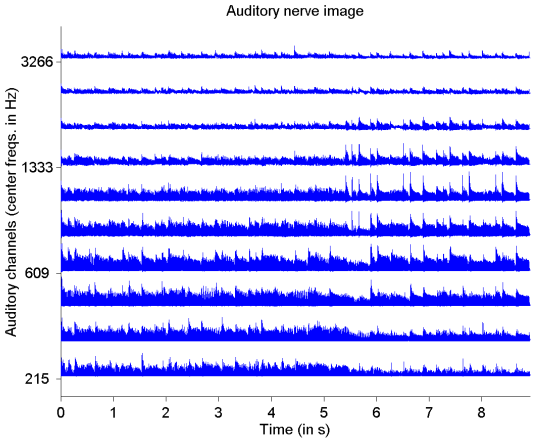


Figure 3: Auditory nerve images of Ligeti's *Presto energico*

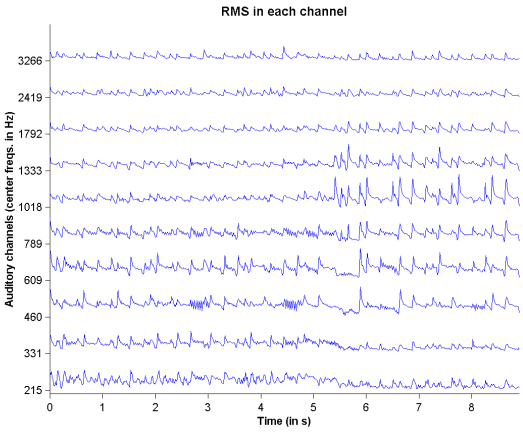


Figure 4: RMS patterns of Ligeti's *Presto energico*

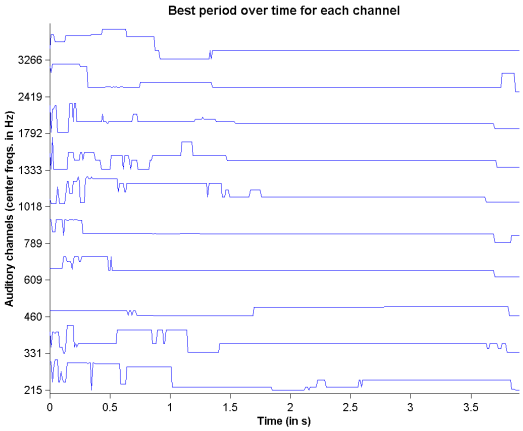


Figure 5: Best period over time for each channel of Ligeti's *Presto energico*

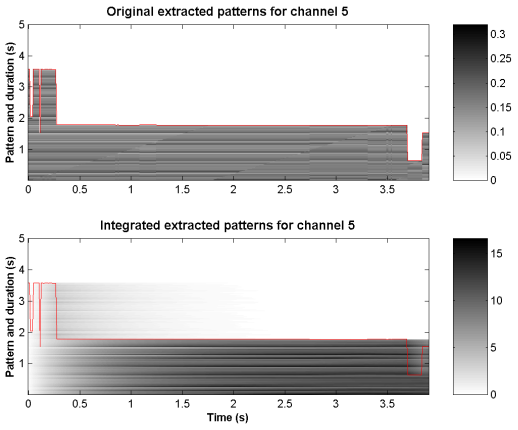


Figure 6: The rhythm patterns of auditory channel 5 detected in Ligeti's *Presto energico*. The figure on top gives the original patterns. The figure below integrates these patterns in time

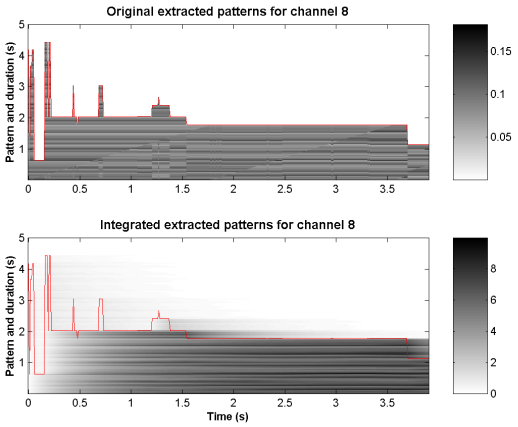


Figure 7: The rhythm patterns of auditory channel 8 detected in Ligeti's *Presto energico*. The figure on top gives the original patterns. The figure below integrates these patterns in time

followed by by a number of zeros, so that all patterns get the same amount of components, up to the maximum τ which is taken in our analysis. In this case, each pattern $\tilde{q}(t_0)$ is filled up to 5 seconds. All patterns thus have an equal length, but they typically have a part filled with values followed by an empty part, filled with zeros. These patterns are plotted vertically in the figure. However, in order to cancel out a fase effect in the plotting, the part which is filled with values is rotated in function of the elapsed time. As a result repeating pulses appear as horizontal lines in the plots.

Figure 6b shows the same function but the patterns are now integrated. The integration smears out the small deviations hence it may be useful to use the integrated patterns for resynthesis. Figure 7 shows the patterns for channel 8.

For resynthesis, we take in each channel the $\tilde{q'}(t_0)$ pattern, where t_0 is a fixed time point and $\tilde{q'}$ is the time integrated $\tilde{q}$ pattern, reiterate the pattern (without the zeros) in order to obtain a sound of a few seconds, and apply it onto a noise band with the same center frequency as the center frequency of the $\tilde{q'}(t_0)$ pattern. The sum of all these amplitude modulated noises is then the resynthesized sound. Note that due to the phase shift, the resynthesized sound *should be* in phase with the original sound. In the sound examples that accompany this paper, the listener will

notice that when the original sound is played together with the resynthesized sound, phase become different towards the end. This is because the resynthesized sound repeats a single pattern and does not take into account short time lapses or slight increase in tempo. The resynthesized sound is just a single period which is reiterated a few times. When the periods are not the same in the different channels it may happen that the synchronization of the sub-band noises becomes spoiled after a few seconds. This problem could be solved by a granular synthesis approach instead of the present crude synthesis technique.

6 Discussion

The MEC approach in its present form has a number of constrains and limitations.

One problem is related to the choice of the local minima detected. A second problem is that the algorithm looks at absolute energy values instead of relative energy values. This may pose problems in cases with rapid crescendo and decrescendo, or with repetitive patterns that contain contrasting energy effects. The physical correlate is indeed based on an increase and decrease of the energy. As the algorithm is based on energy change it might be necessary to adapt local $r(t)$ values as a function of the surrounding energy in order to obtain a relative energy. One possible solution is to subtract the DC-component from $r(t)$.

Change in tempo has an effect on the period. In its present form, the algorithm always detects a minimum in the energy-change curves. A probability threshold could be installed to account for the certainty/uncertainty of the found period.

Further limits concern the detection of the periodicity pattern and its relationship to beat and tempo extraction. In its present form we do not attempt to extract the beat but it might be interesting to note that the $\tilde{q}(t)$ patterns show constant horizontal lines when a pulse pattern is present in the music. A rhythm pattern may consist of several such pulse patterns and the application of a resonator similar to the one of Van Noorden and Moelants 1999 may be useful to find the pulse corresponding to the beat. This aspect, however, needs further study because we don't want to limit this approach to the detection of regular beats since irregular beats are often used in music. Furthermore, we believe that this analysis should be undertaken in view of the expressive features contained in the rhythm patterns.

The MEC-resynthesis offers a number of different possibilities. In the present implementation the detected envelopes in different channels have been normalized and each channel has an equal weight in the resynthesis. It may be necessary to attenuate the very low and very high channels. Furthermore, due to the fact that the periods detected in different channels may have a different length, it may happen that after a while, the resynthesized pattern of different channels gets out of synchrony. The resynthesis, however, has been used to get a rapid auditory feedback. When used in a real-time setup, it may be more opportune to make synthesis dependent on the running periods detected, rather than using a detected period which is then repeated for a while. Hence a granular synthesis approach would be more appropriate.

Our application of the MEC-concept to rhythm analysis and synthesis relates to Scheirer's (Scheirer, 1998) work in that we also use a sub-band division and calculate the RMS values. Our focus, however, is not on regular beat detection but on regular pattern detection. Both Scheirer and we recognize that onset detection is very difficult and perhaps not necessary to account for beat and period. Scheirer looks for a regular beat, whereas we are looking for a regular pattern which may contain an irregular beat. Scheirer uses resonators as analysis-concept whereas we use the minimalization of energy-change as analysis concept. In resynthesis, our approach is different in that we focus on a resynthesis of the pattern associated with a particular found period.

The MEC function (Equation 5) is closely related to AMDF (Equation 21). This function is of course well-known in music and speech processing but it has no epistemological (in the sense of perceptive/cognitive) foundation. The application of MEC to the Fourier Transform, Auto-Correlation and Wavelets leads to a MEC function that is very similar to AMDF. This shows that the MEC concept as minimalization of energy change, is indeed rather fundamental. From our point of view, MEC is interesting because it provides a simple rule-of-thumb and henceforth, a hypothesis of a perceptual theory in which rhythm perception can be conceived of in the most

general terms, without any further assumptions concerning onsets and regularity of pulses and beats. The cognitive load of MEC is minimal and from the viewpoint of perceptive/cognitive modeling it provides a fundamental concept that deserves further elaboration in view of beat and tempo perception.

The MEC-concept can also be applied to pitch perception where minimalization of the energy change would point to the recognized period in the pitch domain.

7 Conclusion

The MEC concept assumes that the minimal change of energy points to the period of a repeating musical acoustical pattern. From this assumption we derived a function which was applied to rhythm recognition. The MEC concept can be related to classical approaches (Fourier transformation, auto-correlation, wavelet transformation) and it provides a foundation for AMDF in the sense that AMDF should no longer merely be seen as a fast alternative for auto-correlation because it can also be derived from Fourier transform and auto-correlation. Using the analysis-by-synthesis paradigm, the model turned out to perform well on rather complex musical patterns, in particular also repeating patterns with irregular beats.

The philosophy underlying the development of the MEC-concept is that repeating rhythmical patterns are expressions of body movement, called gestures, rather than divisions of time. Hence our focus on the energy of the global pattern rather than on onsets. Once extracted, these patterns can be fed into a memory where they can be categorized thus forming memories for gestures.

The results obtained with this simple approach are promising but further work is needed to refine the algorithm in view of rapid changes in crescendo and decrescendo, to implement the algorithm in real-time, and to extract from these patterns information concerning beat and tempo.

References

Cariani, P. (1999). Timing nets for rhythm perception. In Leman, M., editor, *Proceedings of the Tenth Meeting of the FWO Research Society on Foundations of Music Research – Music and Timing Networks*, page 7pp., Ghent. Dept. of Musicology – IPEM.

Leman, M., Tanghe, K., Moelants, D., and Carreras, F. (1999). Analysis of music using timing networks with memory: Implementation and preliminary results. In Leman, M., editor, *Proceedings of the Tenth Meeting of the FWO Research Society on Foundations of Music Research – Music and Timing Networks*, page 7pp., Ghent. Dept. of Musicology – IPEM.

Rabiner, L. and Schafer, R. (1978). *Digital processing of speech signals*. Prentice-Hall, Englewood Cliffs, NJ.

Scheirer, E. (1998). Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, pages 588–601.

van Noorden, L. and Moelants, D. (1999). Resonance in the perception of musical pulse. *Journal of New Music Research*, 28(1):43–66.

Diverse Classifiers for NLP Disambiguation Tasks
Comparison, Optimization, Combination, and Evolution

Jakub Zavrel ♣ Sven Degroeve ♢ Anne Kool ♣
Walter Daelemans ♣
Kristiina Jokinen ♢
♣ CNTS - Language Technology Group, University of Antwerp
{zavrel|kool|daelem}@uia.ua.ac.be
♢ Center for Evolutionary Language Engineering, Ieper, Belgium
{sven.degroeve|kristiina.jokinen}@sail.com

Abstract

In this paper we report preliminary results from an ongoing study that investigates the performance of machine learning classifiers on a diverse set of Natural Language Processing (NLP) tasks. First, we compare a number of popular existing learning methods (Neural networks, Memory-based learning, Rule induction, Decision trees, Maximum Entropy, Winnow Perceptrons, Naive Bayes and Support Vector Machines), and discuss their properties vis à vis typical NLP data sets. Next, we turn to methods to optimize the parameters of single learning methods through cross-validation and evolutionary algorithms. Then we investigate how we can get the best of all single methods through combination of the tested systems in classifier ensembles. Finally we discuss new and more thorough methods of automatically constructing ensembles of classifiers based on the techniques used for parameter optimization.

Keywords: Models and algorithms for computational neural architectures

1 Introduction

In recent years the field of Natural Language Processing (NLP) has been radically transformed by a switch from a deductive methodology (i.e. explaining data from theories or models constructed manually) to an inductive methodology (i.e. deriving models and theories from data) (see e.g. Abney (1996) for a review). An important component of this transformation is the realization that many NLP tasks can be modeled as simple classification tasks or as ensembles of simple classifiers (Daelemans, 1996; Ratnaparkhi, 1997). Thus NLP has been able to capitalize on a large body of research in the field of machine learning and statistical modeling. This, accompanied by the continuing explosion of computer power, storage size, and availability of training corpora, has lead to increasingly accurate language models for a quickly growing number of language modeling tasks. However, which machine learning methods have the best performance on NLP data sets is still an active area of research.

In this paper, we empirically study the performance of a range of supervised learning techniques on a selection of benchmark tasks in NLP. In classification-based, supervised learning, a learning algorithm constructs a classifier for a task by processing a set of examples. Each example associates a feature vector (the problem description) with one of a finite number of classes (the solution). Given a new feature vector, the classifier assigns a class to the problem description it represents by means of extrapolation from a "knowledge structure" extracted from the examples. Different learning algorithms construct different types of knowledge representations: probability distributions, decision trees, rules, exemplars, weight vectors, etc.

Classification-based supervised learning methods seem to be especially well suited for NLP tasks because they fit the main task in all areas of NLP very well: implementing a complex, context-sensitive, mapping between different levels of linguistic representation. Sub-tasks in such

a transformation can be of two types (Daelemans, 1996): segmentation (e.g. decide whether a word or tag is the start or end of an NP), and disambiguation (e.g. decide whether a word is a noun or a verb). We can even carve up complex NLP tasks like syntactic analysis into a number of such classification tasks with input vectors representing a focus item and a dynamically selected surrounding context. Output of one classifier (e.g. a tagger or a chunker) is then used as input by other classifiers (e.g. syntactic relation assignment).

Although the existence of machine learning data repositories (such as UCI (Blake et al., 1998)) has made it easy to compare and benchmark machine learning algorithms, such studies (e.g. Michie et al., 1994) have not usually focused on natural language learning. However, language data sets have characteristics that make them quite different from typical machine learning data sets:

- Size: Millions of example cases, large numbers of features, many of which redundant or irrelevant, and very large numbers of feature values (e.g. all words of a lexicon). This places a high computational burden on many existing learning algorithms.

- Disjunctiveness: Language is characterized by an interplay between rules, (sub)regularities, and exceptions. Even exceptions (that are difficult to distinguish from noise) can be members of small but productive families. Daelemans et al. (1999) have observed that in language datasets low-frequent and exceptional events are important for accurate generalization to unseen events.

- Sparse data: Since language is a system of infinite expression by limited means, the available examples usually cover only a very small portion of the possible space.

With these issues in mind we have conducted a benchmarking study consisting of data sets for several NLP tasks: Grapheme to phoneme conversion, Part of speech tagging, and Word sense disambiguation.

The algorithms which have been evaluated are: Neural networks, Memory-based learning, Rule learning, Decision tree learning, Maximum Entropy learning, Winnow Perceptron, Naive Bayes, and Support Vector Machines. We have run these algorithms on the NLP data sets under identical conditions, and present an overview of the experimental results. These experiments reveal that although some algorithms stick out on average, with their default setting, their relative performace on a new NLP task can vary widely.

It seems worthwhile to look at the application of so called ensemble systems. In ensemble systems, different classifiers are performing the same task, and their differences are leveraged to yield a combined system that has a higher accuracy than the single best component. The reason for this is that, to some degree, the different weaknesses cancel each other out, and the different strengths improve the ensemble system. Thus combination might (always) be a better idea than competition (and selection of the best). As a direct by-product of the system comparison, we already obtain a basic ensemble system, namely one that uses different base learners. The utility of this approach has already been demonstrated to work well for Part-of-speech tagging (Van Halteren et al., 1998; Brill and Wu, 1998), and is a natural fall-out of any system competition (see e.g. Tjong Kim Sang et al., 2000; Kilgarriff and Rosenzweig, 2000). However, the potential of combination is much larger, as there are many ways in which differences between components can be introduced. Preliminary results in building more elaborate ensembles have been obtained and evaluation looks promising.

In the remainder of this paper, we first describe the base machine learning algorithms used in our experiments (Section 2). In Section 3, we then describe the NLP data sets, and next in Section 4.1 the experimental methodology. The algorithms times the data sets define the space of our experiments, the results of which are presented in Section 4. After the benchmark results, we turn to parameter optimization, and present the evolutionary methods we use to optimize large parameter spaces. Then, in Section 6, ensemble methods are introduced, and in Section 7 the design of effective ensembles is rephrased as a large parameter optimization problem. We report the first results with this approach, and finally conclude in Section 8.

2 Base Algorithms

In this section, we give a short description of each of the machine learning methods. These are all supervised classification methods. The basis of this framework is that each algorithm is trained on a set of labeled examples. These examples, which are basically feature-value vectors, are then used to induce decision boundaries in the very high dimensional feature space. As Roth (1998, 2000) shows, under several limiting assumptions, all of the following algorithms can be seen as particular instantiations of a linear classifier in the feature space that consists of all combinations of all features. However, the computational method to arrive at a trained classifier and the representational strategy used by an algorithm can differ greatly. E.g. many learning algorithms are not suited to account for the influence of combinations of features unless this is explicitly represented in their input, some algorithms only use binary features, others multi-valued, some algorithms start from random initialization and others are deterministic, etc. Here we will only give a concise description of each system, in its most common formulation, and describe a few important parameters for each system. Most importantly, we do not manipulate the original feature space to include all feature combinations. This means that Roth's observations about the equivalence of these methods do not hold, and that algorithms which depend on this manipulation of the feature space (e.g. SNoW) are at a somewhat unreasonable disadvantage.

2.1 Memory-Based Learning

Memory-based learning is based on the hypothesis that performance in cognitive tasks is based on reasoning by similarity to *stored representations of earlier experiences*, rather than on the application of rules abstracted from earlier experiences. Historically, memory-based learning algorithms are descendants of the k-nearest neighbor algorithm (Cover and Hart, 1967; Aha et al., 1991).

During learning, training instances are simply stored in memory. To classify a new instance, the similarity between the new instance $\mathbf{z}$ and all examples $\mathbf{x_i}$ in memory is computed using a *distance metric* $\Delta(\mathbf{z}, \mathbf{x_i})$, a weighted sum of the distance per feature.

$$\Delta(\mathbf{z}, \mathbf{x_i}) = \sum_{j=1}^{n} w_j \ \delta(z_j, x_{ij}) \tag{1}$$

where n represents the number of features in an instance. The test instance is assigned the most frequent category within its k least distant (i.e. similar) neighbors. Depending on the system used, a number of different choices are available for the metric. In our experiments, we have used TiMBL, a system described in detail by Daelemans et al. (2000).

2.1.1 Basic MBL metrics

In TiMBL, we can use either an Overlap (Eq 2) or a Modified Value Difference Metric MVDM as the basic metric for patterns with symbolic features. Overlap simply counts the number of mismatching features. The k-nearest neighbor algorithm using Overlap and $k = 1$ is called IB1 (Aha et al., 1991)

$$\delta(z_j, x_j) = \begin{cases} 0 & \text{if } z_j = x_j \\ 1 & \text{if } z_j \neq x_j \end{cases} \tag{2}$$

MVDM (Eq 3; Stanfill and Waltz (1986); Cost and Salzberg (1993)) is a method to determine a graded similarity of the values of a feature by looking at co-occurrence of values with target classes. For two values V_1, V_2 of a feature, we compute the difference of the conditional distribution of the classes C_k for these values.

$$\delta(V_1, V_2) = \sum_{k=1}^{m} |P(C_k|V_1) - P(C_k|V_2)| \tag{3}$$

where m represents the number of classes. A further parameter of the metric is the weighting method. TiMBL's default weights are computed using Information Gain (IG) (Quinlan, 1993),

which looks at each feature in isolation, and measures how much it reduces, on average, our uncertainty about the class label (Eq 4).

$$w_j = H(C) - \sum_{v \in V_j} P(v) \times H(C|v) \tag{4}$$

Where C is the set of class labels, V_j is the set of values for feature j, and $H(C) = -\sum_{k=1}^{m} P(C_k) \log_2 P(C_k)$ is the entropy of the class labels. The probabilities are estimated from relative frequencies in the training set. Information Gain tends to overestimate the relevance of features with large numbers of values. To normalize for features with different numbers of values, Quinlan (Quinlan, 1993) has introduced a normalized version, called **Gain Ratio**, which is Information Gain divided by the entropy of the feature-values ($-\sum_{v \in V_j} P(v) \log_2 P(v)$).

Unfortunately, as White and Liu (1994) have shown, the Gain Ratio measure still has a bias towards features with more values. TiMBL also supports weights based on a chi-squared statistic, which can be corrected explicitly for the number of degrees of freedom.

So, in sum TiMBL has three tunable parameters, the metric, the number of neighbors (k), and the method to compute weights. Unless explicitly optimizing these settings, we have used TiMBL's defaults: Overlap metric, Gain Ratio weighting, and $k = 1$.

2.1.2 Family-based MBL

FAMBL, or Family-based learning, is an extension to MBL where instances in memory are merged into families. The FAMBL package has many options that allow for many types of abstraction. We will only discuss those options that are used for this research. A detailed description of FAMBL can be found in Van den Bosch (1999).

Instead of just placing every training instance in memory, FAMBL tries to merge instances that have the same annotated class and are close together, given a distance measure (see Section 2.1.1). Families are extracted iteratively by randomly selecting an instance from memory and merging this instance with its neighbors of the same class. Merging data points means replacing mismatching symbolic feature values with a wild-card. The distance between a symbolic feature value and a wild-card is always zero.

Since family extraction is done randomly, FAMBL introduces a probing-phase which defines threshold values for the size of a family and the maximum distances between instances in a family. These threshold values are then used in the actual family extraction phase to limit the size of the extracted families. This is referred to as careful abstraction.

So, the classifier induced by FAMBL is a memory of families. A classifier predicts a classification category for a test instance by searching for the closest family in memory ($k = 1$) and assigning the class of the family found.

2.2 Decision Tree Learning

As a representative of the class of decision tree learners, we used the well-known program C4.5 (Quinlan, 1993), which performs *top-down induction of decision trees*, followed by confidence-based pruning. On the basis of an instance base of examples, C4.5 constructs a decision tree which compresses the classification information in the instance base by exploiting differences in relative importance of different features. Instances are stored in the tree as paths of connected nodes ending in leaves which contain classification information. Nodes are connected via arcs denoting feature values. Feature Information Gain ratio (Eq 4) is used dynamically in C4.5 to determine the order in which features are employed as tests at all levels of the tree (Quinlan, 1993).

C4.5 has three parameters, the *pruning confidence level* (the c parameter), the *minimal number of instances represented at any branch of any feature-value test* (the m parameter), and the choice whether to group feature values or not during tree construction (the sub-setting parameter). The first two parameters directly affect the degree of 'forgetting' of individual instances by C4.5, and in previous work (Daelemans et al., 1999), we have shown that for NLP tasks the best results are

obtained at the minimal amount of forgetting. However, in the present experiments we use C4.5's default settings, $c = 25\%$; $m = 2$, and no sub-setting.

2.3 Maximum Entropy Modeling

Maximum Entropy Modeling (ME), tackles the classification task by building a probability model that combines information from all the features, without making any assumptions about the underlying probability distribution.

This type of model represents examples of the task (given by multi-valued features: $F_1...F_n$) as sets of binary indicator features ($f_1...f_m$), for classification tasks the binary features are typically conjunctions of a particular feature value and a particular category. The model has the form of an exponential model:

$$p_\Lambda(C|F_1...F_n) = \frac{1}{Z_\Lambda(F_1...F_n)} exp(\sum_i \lambda_i f_i(F_1...F_n, C)) \quad (5)$$

where i indexes all the binary features, f_i is a binary indicator function for feature i, Z_Λ is a normalizing constant, and λ_i is a weight for binary feature i.

Learning is the search for a model (i.e. a vector of weights), within the constraints posed by the observed distribution of the features in the training data, that has the property of having the maximum entropy of all models that fit the constraints, i.e. all distributions that are not directly constrained by the data are left as uniform as possible (Berger et al., 1996; Ratnaparkhi, 1997). The model is trained by iteratively adding binary features with the largest gain in the probability of the training data, and estimating the weights using a numerical optimization method called *Improved Iterative Scaling*.

In our experiments, the training was done using a hundred iterations of the Improved Iterative Scaling algorithm. The implementation which we use is called MACCENT, and is available from `http://www.cs.kuleuven.ac.be/~ldh`.

2.4 Rule Induction

Ripper (RIP) (Cohen, 1995) is a well-known effective rule induction algorithm. During training it grows rules by covering heuristics. The training set is split in two parts. On the basis of one part, rules are induced. When the induced rules classify instances in the second part of the training set below some classification accuracy threshold, they are considered to overfit and are not stored. Rules are induced on a class by class basis, starting with the least frequent class, leaving the most frequent class as the default rule, which, in general, produces small rule sets (i.e. one class is taken as *'positive'* and the remainder of the instances as *'negative*).

2.5 Winnow Perceptrons

The Winnow algorithm (WIN) (Littlestone, 1988), is a multiplicative update algorithm for single layer perceptrons, i.e. very simple linear neural networks. A single perceptron takes as input the set of active features $\mathcal{F}$ in an example[1], and returns a binary decision as to whether it is a positive or negative example. Let w_i be the weight of the i'th feature. The Winnow algorithm then returns a classification of 1 (positive) iff:

$$\sum_{f \in \mathcal{F}} w_f > \theta,$$

where θ is a threshold parameter and $f \in \mathcal{F}$ runs through the active feature set. In the experiments reported here, θ was set to 1.

A multi-class classifier is constructed out of as many units as there are classes. Each example is treated as a positive example for the classifier of its class, and as a negative example for all the other classifiers.

[1] Active features are a set of indexes of the feature values present in an example.

Training is done incrementally: an instance is presented to the system, the weights are updated, and the example is then discarded. Weights are only added as needed, initially all connections are empty. The updating of the weights is, as said before, done using the multiplicative Winnow update rule, updating the weights only when a mistake is made. If the classifier predicts 0 for a positive example (i.e., where 1 is the correct classification), then the weights are promoted:

$$\forall f \in \mathcal{F},\, w_f \leftarrow \alpha \cdot w_f,$$

where $\alpha > 1$ is a promotion parameter. If the classifier predicts 1 for a negative example (i.e., where 0 is the correct classification), then the weights are demoted:

$$\forall f \in \mathcal{F},\, w_f \leftarrow \beta \cdot w_f,$$

where $0 < \beta < 1$ is a demotion parameter.

In this way, weights on non-active features remain unchanged, and the update time of the algorithm depends on the number of *active* features in the current example, and not on the total number of features in the domain.

The implementation we used is called SNoW (Carlson et al., 1999). We used all its default settings.

2.6 Naive Bayes

Another popular algorithm, also implemented in the SNoW package, is Naive Bayes (NB). Naive Bayes follow the Bayes optimal decision rule, that tells us to assign the class C that maximizes $P(C|F_1...F_n)$, or the probability of the class C given the features $F_1...F_n$). By using Bayes' rule we can rewrite this as:

$$C = argmax_{c_i} \frac{P(F_1...F_n|c_i) \times P(c_i)}{P(F_1...F_n)} \tag{6}$$

The Naive Bayes method then simplifies the problem of estimating $P(F_1...F_n|c_i)$ by making the arguable indpendence assumption that the probability of the features given the class is the product of the probabilities of the individual features given the class:

$$P(F_1...F_n|c_i) = \prod_{1<j<n} P(F_j|c_i) \tag{7}$$

Each probability on the right-hand side can now be estimated directly from the training data using a maximum-likelihood estimate.

2.7 Support Vector Machines

Support Vector Machines (SVM) are an application of the principle of structural risk minimization, introduced by Vapnik (1982). They can be used to induce classifiers that solve binary classification tasks, i.e. that assign one of two classes to an instance. The classifier induced by an SVM is represented by one hyper-plane $\mathbf{w} * \mathbf{x} + b$ that separates the classes in the training set, so that:

(a) The largest possible fraction of training instances of the same class are on the same side of the hyper-plane, and

(b) The distance of either class from the hyper-plane is maximal.

The classifier's prediction, 1 or -1, for a test instance $\mathbf{z}$ is then defined as

$$sgn(\mathbf{w} * \mathbf{z} + b) \tag{8}$$

When both constrains (a) and (b) are satisfied, the upper bound on the generalization error (or true risk) of the induced classifier will be minimal and the hyper-plane is defined as optimal.

In Burges (1998), it is shown that finding an optimal hyper-plane, or training an SVM, is equal to maximizing:

$$W(\mathbf{a}) = \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i,j=1}^{l} a_i a_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) \tag{9}$$

$$\text{(constraint to: } 0 \leq a_i \leq C \text{ and } \sum_{i=1}^{l} a_i y_i = 0)$$

where $\mathbf{a}$ is a variable vector containing the so called Lagrange multipliers (one for each training instance $\mathbf{x_i}$), the y_i are the true class-labels for each $\mathbf{x_i}$, l is the number of training instances in the training set and C is a user defined parameter that reduces the effect of outliers and noise. Function $K()$ is known as a kernel funtion. In the present experiments, we have only tried a linear kernel

$$K(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} * \mathbf{x_j}) \tag{10}$$

Once $\mathbf{a}$ is known, all training instances $\mathbf{x_i}$ for which $a_i > 0$ are the support vectors. From these support vectors, $\mathbf{w}$ and b (Eq 8) can be derived and the optimal hyper-plane is induced.

A one-on-one representation is used in which, for each feature, each symbolic feature-value is mapped to a seperate binary feature. This binary feature has value 1 if the corresponding symbolic feature-value is present in the symoblic representation. Otherwise it has value 0.

A multi-class classifier is constructed by combining single-class classifiers. Each single-class classifier distinguishes one class from all other classes. The max-operator is used to combine the single-class classifiers, i.e. the class corresponding to the single-classifier with the highest prediction (see Eq 8) is chosen as the multi-class classifiers prediction.

The implementation we used is `SVM_light` (Joachims, 1999). In `SVM_light`, the default value for the C parameter is 1000. Unless stated otherwise, this value is used in the experiments below.

2.8 Multi-layer Perceptrons

A multi-layer perceptron (denoted below by NN), is a type of neural network that is able to make a nonlinear mapping from input to output, because it develops internal intermediate representations in its so called 'hidden layer'. The classifier induced by training a typical feed forward back-propagation neural network is a weighted combination of q hyper-planes.

$$h_j(\mathbf{x}) = \mathbf{w_j} * \mathbf{x} + b_j \qquad (j = 1...q) \tag{11}$$

where q is a user defined parameter, also known as "the number of hidden nodes", $\mathbf{w_j}$ and b_j define the hyper-plane and $\mathbf{x}$ is an instance variable. For each class C_i, the confidence for a test instance $\mathbf{z}$ to belong to C_i is obtained by a weighted combination of the distances $h_j(\mathbf{z})$ of $\mathbf{z}$ to each of the q hyperplanes.

$$\mathit{Conf}(C_i) = \sum_{j=1}^{q} (\mathbf{w_i} * f(h_j(\mathbf{z}))) \tag{12}$$

where $\mathbf{w_i}$ is a weight vector associated with class C_i and $f()$ is an activation function, used to translate the distance value, allowing the combination of the hyper-planes to separate classes that are not linearly separable. For the activation function, which is a user-defined parameter, we have used a sigmoid function.

Usually, the class with the highest confidence is chosen to be the classifiers' prediction. Training a neural network means positioning the q hyper-planes such that the confidence values (Eq 12) for

the training instances are as close as possible to their true confidence values. In the case of back-propagation, the weights in (Eq 12) and (Eq 11) are gradually adjusted, given a starting point, by a backward propagation of the difference between the confidence values given by (Eq 12) and the true confidence values. Usually, a validation set (a subset of the training instances, not used during training) or other techniques such as early stopping (Prechelt, 1998) are used to prevent the network from overfitting the training instances.

A one-on-one representation is used in which, for each feature, each symbolic feature-value is mapped to a seperate binary feature. This binary feature has value 1 if the corresponding symbolic feature-value is present in the symoblic representation. Otherwise it has value 0.

We chose to train a separate neural network (single-class classifier) for each class in the training set. As in the SVM architecture, a multi-class classifier is constructed by combining single-class classifiers. Each single-class classifier distinguishes one class from all other classes. The max-operator is used to combine the single-class classifiers, i.e. the class corresponding to the single-classifier with the highest prediction is chosen as the multi-class classifiers' prediction.

The experiments were performed using the SNNS package (Zell et al., 1995). All weight-values $\mathbf{w}$ (Eq 11 and Eq 12) where initialized randomly within $[-1, 1]$. A default learning rate of 0.2 is used. The number of hidden nodes q is taken as the best from $\{10, 20, 30, 40, 50, 60, 70\}$.

3 Data

In this section we describe the data sets used in our benchmarking experiments (Grapheme to Phoneme, Part-of-speech Tagging, Conversion, Word Sense Disambiguation). The selected tasks reach from low-level phonetic processing, through shallow syntactic processing, to higher level semantic judgments, respectively. The choice of these data sets was made to include both small and large values on number of dimensions (size, number of features, number of values, number of categories, regularity). Each of the selected tasks is in itself a challenging problem, but here we do not focus on the solution of these problems, but rather take the selected tasks and data sets as given, and just use them for comparison of algorithms. As said in the introduction, we restrict ourselves to single classification tasks, whereas many interesting NLP tasks would be composed of many such decisions in cascades.

The pre-processing we used for the datasets is mostly common practice and is described in detail in the descriptions of the datasets. To maximize comparability with other published results we tried to keep as closely as possible to publicly available datasets, or datasets extracted from well-known generally available datasets.

3.1 Grapheme-Phoneme With Stress

In this data set, further be referred to as GS, the mapping to be learned is from a letter in context to a phonetic representation with stress markers. This dataset is based on the CELEX dictionary (Baayen et al., 1993) for English. For every word in that dictionary, the letter to be transcribed (the focus), and a context window of three letters to the left and to the right are given as features.

An example of the word "above" converted to windowed training instances is:

```
_,_,_,a,b,o,v,0@.
_,_,a,b,o,v,e,1b.
_,a,b,o,v,e,_,0V.
a,b,o,v,e,_,_,0v.
b,o,v,e,_,_,_,0-.
```

The first character of the target category represents whether the syllable starting with that letter receives stress in the pronunciation (1) or not (0). The second letter is the phoneme corresponding to the focus letter (the 4th feature).

The dataset consists of 77565 words divided into a training set GS-DATA (69808) and a test set GS-TEST (7757). The total number of instances in training and test set is respectively 608228

and 67517. For some experiments we have considered the DATA and TEST parts as separate tasks (large and small version). The number of features is modest and each feature has the same amount of values (number of symbols in the alphabet). In previous research (Van den Bosch, 1997), it has been shown that this task is one where exceptions, and sub-regularities play a large role. Also, obviously, the interaction between the features (and in particular the focus and variable sized portions of the context) is crucial.

3.2 Part-Of-Speech Tagging

Part-of-speech (POS) tagging is the task of assigning the single most appropriate morpho-syntactic category to a word on the basis of its context. If the word has been observed in the training data, we have lexical information available (possible categories, also called "ambiguity classes"); if the word has not been seen before, we must guess on the basis of form and context features. Hence there are two versions of the data, one involved with predicting the POS for known words, and one for unknown words[2].

Our dataset is based on the TOSCA tagged LOB corpus Johansson (1986) of English[3]. The features represent information about the word to be tagged (focus) and its context, and are similar but slightly different for the two sets.

3.2.1 Known Words

The known words set (1045541 cases, henceforth: POS-KNOWN, was made from every 1st through 9th sentence of the corpus (90% of the total), using the following ten features:

```
W : The focus word itself
d : the POS tag of the word at position n-2
d : the POS tag of the word at position n-1
f : the ambiguity class of the focus word (position n)
a : the ambiguity class of the word at position n+1
a : the ambiguity class of the word at position n+2
s : the 3rd last letter of the focus word
s : the 2nd last letter of the focus word
s : the last letter of the focus word
h : does the word contain a hyphen?
c : does the word start with a capital letter?
```

3.2.2 Unknown Words

The unknown words set (65275 cases, henceforth: POS-UNKNOWN) was also made from every 1st through 9th sentence of the corpus (90% of the total). However, as the distribution of unknown words closely resembles that of the low-frequent words, the only words that are included in this set are words that occurred 5 or less times in the whole dataset. The features for this set are the same as for the known words, except that the focus word itself and its ambiguity class are omitted.

The POS-KNOWN set is very large, whereas the POS-UNKNOWN set is of intermediate size. The number of features is intermediate, and some features (e.g. the focus word) have very large numbers of values, whereas others (e.g. hyphen) are only binary. The number of categories is 201 for KNOWN, and for 118 for UNKNOWN. The KNOWN words data is quite regular (i.e. the most frequent category of a word, regardless of context, already scores more than 90% correct; most capitalized unknown words are proper nouns, etc.), but there seems to be a large number of infrequent exceptions.

[2]Note that this is a task that *resembles* POS tagging, but is not actually comparable to the tagging of unseen text. Here each word is processed in isolation, assuming a correctly disambiguated left context. Also the unknown words are not really unknown, they are just infrequent.

[3]Kindly provided to us by Hans van Halteren of the TOSCA Research Group at the University of Nijmegen.

3.3 Word Sense Disambiguation

Word sense disambiguation (WSD) is the task to select the appropriate sense for a word from a predefined finite set on the basis of its context. Our dataset is based on the 1998 Senseval competition (Kilgarriff and Rosenzweig, 2000), which compared machine learning methods on a small sample of ambiguous words:

> accident (1279), amaze (327), band (1418), behaviour (1009), bet-n (168), bet-v (102), bitter (193), bother (350), brilliant (481), bury (344), calculate (289), consume (111), derive (294), excess (290), float-a (57), float-n (94), float-v (261), generous (339), giant-a (316), giant-n (389), invade (82), knee (530), modest (415), onion (43), promise-n (622), promise-v (1472), sack-n (125), sack-v (195), sanction (117), scrap-n (81), scrap-v (47), seize (340), shake (1099), shirt (564), slight (427), wooden (378)

The total number of examples for a word is between brackets (all words together form a set of 14648 instances). Training and testing is done for each word separately. The features in these data sets represent the following information: The first nine features represent two words to the left, the word to be disambiguated (focus), and two words to the right, each word is followed by its part of speech (Penn Treebank tagset). After this immediate context come a number of binary features indicating the presence (1) or absence (0) of a number of focus-specific keywords in a wider context around the word of interest. These keyword features are different (also in number) for each word, and they were selected using the default method suggested by Ng and Lee (1996).

An example instance for the word "accident" is:

> after, IN, an, DT, accident, NN, at, IN, the, DT, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 532675

The target category (here 532675) is a six-digit code that corresponds to a sense-entry in the HECTOR dictionary[4].

In sum the WSD data has few training examples, many features, some of which have large numbers of values, and others are just binary. The number of categories is relatively small, but differs considerably from word to word, as does the ratio between regularity and exceptional cases. For some words, the task is very difficult, because of the interaction between features, and the lack of sorely needed common-sense knowledge.

4 Comparison Of Algorithms

4.1 Methodology

In this benchmark study, we are especially interested in differences between algorithms with regard to their generalization ability. How well does a particular algorithm process new data, when trained on a particular training set. We estimate this by the *accuracy* (or its inverse, error), operationalized as the percentage of previously unseen data items classified correctly by the algorithm.

The underlying assumption in most work on machine learning is that new data is drawn according to the same distribution as the training data. We operationalize this by randomizing the data set, and then splitting into train and test partitions. To ensure statistical reliability, we do a *10-fold cross-validation* (10CV) (Weiss and Kulikowski, 1991) (divide the data set into ten partitions after randomization, use each partition in turn as test set and the other nine as training set, compute evaluation criteria by averaging over the results for the ten test sets).

For those algorithms that parameterize their settings, we used the default settings in most experiments, except where an optimal set of parameters settings was explicitly selected. This was done by cross-validating different parameter settings (within reasonable bounds) on the training set. This means that part of the training set was used as a validation test set, and the parameter settings providing the best result on this validation test set were used for the real test set.

In the case of 10CV experiments, the train/test split were identical for all algorithms, allowing direct comparison. For some experiments 10CV was computationally not feasible. In such cases only a single train/test run was done on the first partition of the 10CV.

[4]See http://www.itri.brighton.ac.uk/events/senseval/

	Accuracy			
Algorithm	POS-KNOWN	POS-UNKNOWN	GS-DATA	GS-TEST
TiMBL	97.5	82.8	**92.8**	**81.9**
FAMBL	–	80.5	91.3	–
C4.5	–	79.2	–	80.3
ME	**98.1**	**83.7**	79.3	76.7
RIP	96.4	80.1	76.2	73.7
WIN	97.4	74.4	64.5	62.1
NB	96.6	79.2	70.1	68.4

Table 1: Generalization accuracies (10CV) for the POS and GS tasks.

4.2 Experimental Results

As of October 2000, a large part of the experimental matrix has already been completed. However, only for the small task WSD, we have succeeded so far in getting results for all described algorithms (TiMBL, FAMBL, C4.5, RIP, ME, WIN, NB, NN, and SVM). The results for all 36 words of the WSD task are given in Table 6 at the end of the paper. We see that the difficulty of the task varies considerably across words, and that different algorithms give the best performance for different words. Nonetheless, some clear tendencies can be observed. The bottom of the table summarizes these, by giving the number of words for which an algorithm is the winner, and the average rank of the algorithm. From the average rank we can obtain an overall order between the algorithms in terms of their consistent performance (from best to worst):

$$\text{SVM} > \text{TiMBL} > \text{ME} > \text{NB} > \text{FAMBL} > \text{RIP} > \text{NN} > \text{WIN} > \text{C4.5}.$$

From the number of first places it is very clear that SVM performs very well on this data set, and the other algorithms remain far behind. This clearly shows that SVM's can both maintain a rich representation of the decision boundaries, while at the same avoid overfitting on the small data sets provided for each word. Several other studies (Mooney, 1996; Escudero et al., 2000) have shown similar results, in particular the good performance of Naive Bayes on WSD is a recurring reason for wonder, given the simple model this algorithm makes.

For the remainder of the tasks not all algorithms could be applied. The data sets were either too big to fit in memory with a particular implementation, or the algorithm did not scale up well and took too long to terminate[5]. This is disappointing, as one of the victims of this was SVM which produced excellent results for WSD. More sophisticated task decomposition strategies, such as e.g. pairwise coupling (Moreira and Mayoraz, 1998), should however improve this situation in future research.

The results of the experiments for POS and GS are shown in Table 1. The systems tested here are: TiMBL, FAMBL (POS-KNOWN not done), C4.5 (POS-KNOWN terminated), ME, RIP, WIN, NB. For the POS-KNOWN task we get the order (from best to worst):

$$\text{ME} > \text{TiMBL} > \text{WIN} > \text{NB} > \text{RIP}.$$

The systems which allow a better modeling of inter-feature dependencies seem to be superior to the systems which consider the features in isolation (NB) or produces small rule sets (RIP). On the POS-UNKNOWN task we get the following order, from which a similar conclusion can be drawn, although the lower echelons are slightly different:

$$\text{ME} > \text{TiMBL} > \text{FAMBL} > \text{RIP} > \text{NB}\|\text{C4.5} > \text{WIN}.$$

The resulting ordering for the large version of the GS task is:

$$\text{TiMBL} > \text{FAMBL} > \text{ME} > \text{RIP} > \text{NB} > \text{WIN}.$$

and for the small version (GS-TEST):

[5]we have run the experiments on dual Pentium III machines, 512 MB, Redhat Linux. If an algorithm did not produce results within a week it was terminated.

TiMBL > C4.5 > ME > RIP > NB > WIN.

Here, again, the algorithms that can model complex feature interactions win over those that cannot, and moreover, TiMBL is at an advantage as this task is well-known for being ridden with exceptions, and semi-regularities. This is also strikingly demonstrated by the fact that when going from 10% (GS-TEST) to 90% (GS-DATA) of the data, the other algorithms improve only slightly ($< 2.6\%$), whereas TiMBL gains an extra 10.9%.

5 Parameter Optimization

So far we have only used default parameter settings for each algorithm. This results in a certain ordering of the algorithms on each task. To make this result worthwile, it should help us in picking an appropriate learning algorithm for a new task. However, we see that the ordering depends on the task at hand. In this section we will first look into additional improvements of the single algorithms, and after that (in Section 6) we will look at system combination as a method to always do as well or better than the best single algorithm. It turns out that the methods to tune and improve a single algorithm, can be reused to get good combinations, i.e. tuned ensembles.

There are at least two ways in which the results for each of the algorithms could be improved. First, we could fine-tune the parameter settings for each system for each task. And second, we could try to adapt the problem representation to make a task better fit the bias of a particular algorithm. This will be left for future research. In this section we consider a case-study with TiMBL as a part of a limited excursion into the first territory (parameter tuning). A full study of all parameter tunings of all algorithms is beyond the scope of this paper. The main point we want to make here, is that a) parameter tuning can make a huge difference to the outcome of any benchmarking study, and b) exhaustive parameter tuning is often impossible.

In memory-based learning, what we want to tune is the number of k nearest neighbors, the metric, and the weighting scheme. Given a few settings per parameter, it is not unfeasible to exhaustively explore this parameter space on a validation set. However, it can also be beneficial to select a subset of features. Moreover, parameter optimization and feature selection or weighting are likely to interact. This situation, typical for an algorithm with a medium to large number of parameters calls for non-exhaustive optimization capable of efficiently avoiding local minima. Therefore, evolutionary algorithms algorithms promise to be of use.

In the experiments, we linked TiMBL to PGAPACK[6]. During the feature subset selection experiments the string is composed of binary values, indicating presence or absence of a feature. During the simultaneous optimization experiments, the first gene in the string encodes the values for k (only odd values are used, to avoid ties), the second gene indicates which weight settings are used and the remaining genes are reserved for the features. In these experiments we look at feature selection as an optimization process, where each feature has three possible values: a feature can either be present, it can be absent or its MVDM can be calculated. Each feature-gene can take on any of these three values and subset selection is then optimization of these values for the specific features. The fitness of the strings is determined by running the memory-based learner with each string on a validation set, and returning the resulting accuracy as a fitness value for that string. Hence, selection with the GA is an instance of a *wrapper* approach as opposed to a *filter* approach such as information gain (John et al., 1994).

For comparison with evolutionary feature selection, we include two popular classical wrapper methods: backward elimination (henceforth BA) and forward selection (henceforth FO).

In Table 2 we show the results of our experiments on POS-KNOWN, POS-UNKNOWN, and GS-TEST. We can see that *a)* exhaustive search (EX) for optimal parameter settings improves the classification accuracy and that *b)* selection of a subset of features leads to similar or better results with a reduction in the number of features used. For *c)* simultaneous parameter optimization and feature selection, show improvement for the POS-KNOWN task (significant; McNemar's chi-square; $p<0.001$) , and the GS-TEST task (not significant; $p=0.318$). The exhaustive search

[6] A software environment for evolutionary computation developed by D. Levine, Argonne National Laboratory, available from ftp://ftp.mcs.anl.gov/pub/pgapack/

for optimal parameters is better than the simultaneously optimized case for POS-UNKNOWN unknown (but not significantly; p=0.684). For a more detailed discussion of these results, see Kool et al. (2000).

Task	Results		
POS-UNKNOWN		Default Parameters	Optimized Parameters
	All Features	DE 82.6	EX 85.4
	Optimized Features	GA 84.4	GA 84.9
		BA 84.4	BA 85.2
		FO 84.5	FO 85.0
POS-KNOWN		Default Parameters	Optimized Parameters
	All Features	DE 97.5	EX 98.3
	Optimized Features	GA 98.3	GA 98.2
		BA 98.3	BA 98.4
		FO 98.3	FO 98.4
GS-TEST		Default Parameters	Optimized Parameters
	All Features	DE 81.6	EX 81.7
	Optimized Features	GA 81.6	GA 82.0
		BA 81.6	BA 81.5
		FO 81.6	FO 81.6

Table 2: Feature and parameter optimization results.

Although the improvements are by no means dramatic, they do already have consequences for the ranking of algorithms in our benchmark. Compare the best results on POS (resp. 98.1% for KNOWN and 83.7% for UNKNOWN, Table 1) which were obtained by ME, with the best results obtained here with a parameter optimized version of TiMBL (resp. 98.4% and 85.4%, Table 2). These are indeed much larger differences than e.g. between TiMBL and ME with their defaults on. A less pronounced, but still interesting, difference is found for GS, where TiMBL is able to improve its result from 81.6% to 82.0%.

So, the optimization of small numbers of parameters is always to be recommended, and can be done by an exhaustive search on the validation set. Simultaneous application of feature selection and parameter optimization has shown some performance gains, but further work on better search algorithms is needed to realize the full potential of the approach. The applicability of this approach goes well beyond TiMBL. Other machine learning algorithms are confronted with similar feature weighting, feature selection, and parameter optimization problems, and these results are likely to be relevant for these other algorithms as well. For example, an small optimization run of SVM parameters on the WSD task (C and the dimension of the kernel function) resulted in an average improvement of 3.7 % per word over the already very good accuracy results in Table 6.

6 System Combination

As argued throughout this paper, disambiguation tasks in NLP can be characterized as complex mappings from large amounts of features to large amounts of categories. From the benchmarks we can see that learning these tasks from corpora tends to push existing learning algorithms to their limits. A possible solution for this problem is to modularize a task as a series of more simple problems. However, for most tasks a good decomposition is difficult to design.

An alternative, and fully automated approach towards modularization is offered by recent work in Machine Learning. Starting from the observation that different learning systems make different errors when trained to perform the same task, and among all the system's outputs the right output is more likely to be present somewhere than in any single system, so called "combination methods" attempt to train an ensemble of diverse classifiers and combine these to yield a composite classifier with higher accuracy. There are four dimensions on which diversification

can be attempted (Dietterich, 2000), and we can in fact consider these as possible paths towards modularization:

1. Data modularization. E.g. in AdaBoost (Freund and Schapire, 1997), each consecutive component system receives a training set in which the items classified wrong by the previous components are given a higher weight.

2. Target category modularization. Error Correcting Output Codes, ECOC (Dietterich and Bakiri, 1991) train an ensemble in which each component learns one binary split between categories. A similar approach is followed by Pairwise Coupled Classifiers (Moreira and Mayoraz, 1998).

3. Feature modularization: E.g in Bay (1998), a performance gain is obtained by combining several nearest neighbor classifiers, each trained with a random subset of the available features.

4. Bias modularization: Different learning algorithms can be used as components (e.g. Van Halteren et al., 1998), or the same algorithm with different parameter settings.

Interestingly, each dimension of variation can results in accuracy gains, even though the only criterion used to make ensemble members is to ensure that they have some diversity (see e.g. evidence in Dietterich, 2000). Oftentimes, it is also stated that the components must be "accurate enough". As we will see, however, this criterion depends on the combination method used.

6.1 Combination Methods

Once we have trained a set of diverse components, there is a number of ways to combine their outputs. The most straightforward way to do this is **voting**. Voting can be very simple, i.e. each component cast a single vote for its own output, or more sophisticated, by casting weighted votes, and perhaps even countervotes. Although it is certainly by far the most popular combination method, certain properties of voting make it a bad choice for constructing ensembles. First, voting can only result in an ensemble output that is present between the component outputs[7]. Second, following from this, for voting to work, all the components should use the same class labels. This can be a problem when we want to integrate diverse sources of knowledge in the ensemble. Third, and not least, bad components will drag the whole ensemble down.

A much more powerful and effective way to do combination, called **stacking** was proposed by Wolpert (1992). Stacking involves two levels of learning. On top of the components, we place a second level, or meta learner, which is trained to map the vector of component outputs to the correct ensemble output. This gives us a much greater freedom: we can use a completely different code at the intermediate level than at the output level, we can use components with diverse codes, and we can even use very misguided components, as long as their outputs are is some way systematic. In fact we can even emulate voting, because (weighted) voting is a special case of stacking, where each output class has one codebook vector [8]. The downside of this freedom, is that the second level must be trained, and for training we need enough data. If we train the second level using training data that was also used to train the components, these will be too correct to reliably estimate error patterns from, and the second level will fail to learn any error-correction. Hence we must use a separate tuning set, or produce a cross-validated output of the components on the training set (which we do in the experiments below). Another point that complicates stacking somewhat is that the choice of the second level learner is as much an open issue as the choice of components. (In our experiments we have found that unweighted TiMBL-IB1 works well, as does TiMBL-MVDM with $k = 9$).

Stacking is a very powerful framework, because, given the freedom of different intermediate representations, we can also use diverse recodings of the original features as 'ensemble components'.

[7] Unless a special voting code book is used, as in ECOC's.

[8] This insight is due to Dietterich (2000).

A special case of this is what we will call **arbiter** learning, where in a stacked ensemble, the meta-learner is also given all the original input features. This allows the meta-learner to error-correct the patterns produced by the component outputs based on their place in the input space. Another way to see this, is that the components are producing compressed representations of their inputs. But when their compression rate is too high, because of a coarse-grained class scheme, the second level looses to much information about the context of the decision. The arbiter method allows us to partially remedy this.

6.2 Basic Ensembles

In this section we report experiments with two of the easiest dimensions of variation. The first (bias) naturally falls out of the benchmark experiments. Since we have done a 10CV of each algorithm on the same data, we can easily construct an ensemble of these.

	POS-UNKNOWN	POS-KNOWN	GS-DATA	GS-TEST
Single components				
TiMBL	82.6	97.5	92.8	81.6
ME	83.2	98.1	79.2	77.0
RIP	80.1	96.4	–	–
WIN	74.1	97.5	63.7	62.5
Ensembles				
majority	84.7	98.3	83.2	78.8
stacked (IB1)	85.1	98.4	92.6	81.5
arbiter (IB1)	86.4	98.4	93.4	83.4
arbiter (MVDM-k9)	86.4	98.6	93.1	83.6
oracle	93.3	99.4	95.9	90.5

Table 3: System combination ensemble results for POS and GS tasks. These experiments have been performed on one train/test partition only.

Table 3 shows the results for the POS and GS tasks of combining diferent base learners (systemcombi). For POS we used TiMBL, ME, RIP and WIN as the components, for GS, the outputs of RIP were not available, so we only used the three remaining components[9]. These ensembles were only tested on one partition of the 10CV split, so we provide the corresponding single system results as a comparison. We see that for POS majority voting already shows a performance increase. For GS this is not the case–we see a big drop for voting here, because the first and second runner up are much worse than TiMBL. As we can see in the table, this is not a problem for stacking. For POS stacking produces a considerable accuracy increase, and for GS it at least approximates the best single component. The arbiter method produces an improvement even larger for POS (up from 83.2% for ME to 86.4% for UNKNOWN and up from 98.1% to 98.6% for KNOWN) and considerable for GS as well (from 92.8% for a single TiMBL to 93.4% on GS-DATA, and from 81.6% to 83.6% for GS-TEST). The bottom row of the table shows the accuracy of an oracle system, which would always suggest the correct category if one of the components would propose it. As we can see, there is still room for improvement[10]. In future work we should also look at the issue how the number of components influences the performance of the ensemble.

A second ensemble that is relatively easy to construct for our data sets, is an ensemble based on feature variation. Each component uses a default TiMBL learner, on the basis of a different feature subset (featurecombi). We constructed nine components for each ensemble by hand, trying to ensure variation in the set. This is similar to the experiments of Bay (1998), who uses random feature subsets for each component. The various feature subsets, and the results, are shown in Table 4. As we can see, most of the components are much simpler than the full feature

[9]These are again the systems with default settings.

[10]Note however, that in theory the oracle is only an upper bound for voting, not for stacking.

	POS-UNKNOWN		GS-TEST	
Single components				
	`ddaasssch`		`pppfsss`	
	111111111	82.6	1111111	81.6
	110000000	44.8	0001000	47.0
	001100000	41.6	0011000	61.1
	111100000	53.4	0001100	63.0
	000011100	62.9	1110000	28.6
	000000011	38.7	0000111	32.1
	000011111	72.4	0011100	76.3
	110011100	72.7	0111110	81.7
	001111100	68.5	0010100	35.0
Ensembles				
majority		81.6		80.6
oracle		95.5		95.0
stacked (IB1)		84.8		82.5
stacked (MVDM-k9)		85.3		82.6
arbiter (MVDM-k9)		86.9		82.7
stacked+systems (MVDM-k9)		86.6		82.5

Table 4: Feature combination ensemble results for POS-UNKNOWN and GS-TEST. These experiments have been performed on one train/test partition only. The **stacked+systems** entry (bottom row) includes all systems from Table 3 and all feature subsets components.

representation, and hence do not perform very well by themselves. Neither does the majority voting ensemble perform well. As an illustration of the power of stacking, however, this experiment is sufficient. Both stacked and arbiter version produce better results than the systemcombi ensemble for POS. For GS, the stacked version is better than systemcombi, but, disappointingly, the arbiter version is not. Finally, if we put the components of systemcombi and featurecombi together, the results further improve upon both simple stacked ensembles for POS.

7 Evolution Of Modular Ensemble Systems

In Section 6 four dimensions of variation have been identified that can be used to cause variation among classifiers, and hence possibly improve the performance of an ensemble. Two of these dimensions (bias and feature set) were shown to be effective on two of our data sets. Ensemble methods are a very active area of research and these and other variations are shown to work well time after time (see Dietterich (2000) and the references therein). What is striking however, is that so far:

- Few attempts have been made to optimize the divergence in an ensemble directly in order to get better performance[11]. In most research on combined methods, an ensemble is constructed by making a diverse set in some ad-hoc fashion–as we have done–, and then combining these components (usually by using voting).

- Moreover, the four dimensions of variation have mainly been studied in isolation. This in spite of the fact that a simultaneous variation in bias *and* output coding *and* data set *and* feature set might have much more far-reaching effects.

In this section we sketch the outline of a method that can exploit all of the above mentioned dimensions of modularization, while at the same time explicitly optimizing the composition of the ensemble using Genetic Algorithms. It is a derivative of the method used for feature and

[11] We are only aware of work in this direction in the Neural Networks field (e.g. Moriarty and Miikkulainen, 1997; Yao, 1999), but have not yet encountered such work in symbolic Machine Learning.

parameter optimization in Section 5. We simply encode the whole ensemble as a large vector of parameters.

A first population of ensembles is generated with random settings for each component, all components are trained on the same task, a 'stacked' second level algorithm learns to combine the outputs of the ensemble, and the combiner's test score is used as a fitness value. Subsequent generations are formed by cross-over and mutation from the fittest ensembles. Using the score of the whole ensemble as a fitness measure ensures the selection of good 'team-players' as constituents, even though we have no good understanding how quality (accuracy) and specialization (de-correlation of errors) contribute to the global solution. This should typically lead to automatic modularization of the task. As shown in Table 5, in our first experiments, this method has a higher accuracy than both the best individual system, and the 'hand-designed' systemcombi and featurecombi systems for the GS task. For POS results are better than systemcombi but the same (stacked version) or slightly worse (arbiter version) than featurecombi.

However, these experiments are only starting to scratch the surface of what is possible with these methods. We hope that additional performance gains can be realized when more dimensions of variation are included in the optimized ensembles. But, it remains to be seen whether GA's are an appropriate optimization method for this application.

	POS-UNKNOWN		GS-TEST	
Single components				
	`ddaasssch`		`pppfsss`	
	011011011	71.7	0110000	26.0
	110100110	65.5	1001110	70.0
	001011111	76.8	1101000	57.3
	111111001	70.8	0010111	44.9
	010111111	80.4	0110010	33.0
	010111100	72.4	0001111	70.2
	100111111	74.7	0111111	81.5
	011000010	52.3	1111100	77.9
	001100010	41.7	1000001	18.8
Ensembles				
handmade-featurecombi stacked		85.3		82.6
handmade-featurecombi arbiter		86.9		82.7
GA-stacked		85.3		83.3
GA-arbiter		86.5		84.4
GA-**feat+param**-stacked		85.8		83.4
GA-**feat+param**-arbiter		86.4		84.0

Table 5: GA optimized feature modular ensembles for POS-UNKNOWN and GS-TEST. These experiments have been performed on one train/test partition only. Each component uses a default TiMBL learner, on the basis of a different feature subset. For the **feat+param** entries (bottom two rows) the GA optimized both the feature subset as well as the TiMBL parameters. The stacked and arbiter systems all use TiMBL with MVDM-k9 as the second level learner.

8 Conclusions And Future Work

In this paper we have studied various supervised machine learning techniques and compared their performance on a set of well-known NLP tasks such as Grapheme to phoneme conversion, Part of speech tagging, and Word sense disambiguation. The methods included in the study are: Neural networks, Memory-based learning, Rule induction, Decision trees, Maximum Entropy Modeling, Winnow Perceptrons, Naive Bayes Method, and Support Vector Machines.

We have run the algorithms on the NLP data sets under indentical conditions, and the results of these experiments are in line with previously obtained results (Daelemans et al., 1999). In

particular, Memory-based learning performs well across our NLP benchmark tasks. However, on particular tasks, there are strong competitors: On WSD, Support Vector Machines show an outstanding performance. This algorithm, however, still has trouble scaling up to large NLP tasks, and requires future research on more sophisticated task decomposition strategies. On POS, Maximum Entropy modeling is slightly better than Memory-based learning, but this is mitigated after tuning TiMBL's parameters.

Furthermore, we have also shown how important feature and parameter tuning is, and discussed how it can be done using GA's or more traditional search methods. Further work is needed, however, to evaluate the full potential of this approach.

The comparisons clearly demonstrate how the techniques perform very differently on the different data sets, and how some of them have serious limitations regarding large input space. We have thus also investigated ensemble systems, combinations of different classifier systems, and their performance on the same NLP benchmark tasks. We have provided arguments and empirical evidence that stacking is superior to voting in ensemble systems. In particular, using stacking, it seems a good idea (if there is enough training data) to always use ensembles rather than the best single algorithm.

Using stacked and arbiter combination we have been able to build effective ensembles (beating every single component system) from natural collections of components, such as divergent feature sets, and different learners participating in a benchmark. In future work, the remaining two dimensions of variation (i.e. data set modularization, and output coding modularization will be investigated as well.

We have proposed a new framework for constructing optimized modular ensemble classifiers using GA's. The first evaluations are very promising. In future work, we will continue in this direction. In particular, we plan to investigate the development of intermediate representations that work well for stacking. The research will also include investigations on advanced hardware and especially, on parallel computing, in order to tackle the notorious scaling problem caused by memory and time limits when dealing with large NLP data sets.

A strong motivation behind our theoretical and empirical investigations of the integration of machine learning approaches to NLP tasks is the improvement that these new methods may offer to speech and language applications. Some of the biggest problems in speech and language technology deal with knowledge acquisition, flexible interaction management, and robust speech processing, all areas where the complexity of the task requires adaptive methods that can classify new items and new situations into approriate classes efficiently and with sufficient accuracy. The results on modularity, combination, and evolutionary optimization will thus have important heuristic value for systems that learn (Jokinen, 2000).

The research on appropriate representations and combination of different learning methods into ensembles can also support building of hybrid systems that integrate symbolic and stochastic processing as well as supervisedly and unsupervisedly learned representations for improved performance. Our methodological studies can thus be related to real-world language and speech applications which serve as reference points against which various solutions can be further evaluated and compared.

Word	Accuracy								
	TiMBL	FAMBL	C4.5	ME	RIP	WIN	NB	NN	SVM
accident	87.1	85.2	69.0	87.8	88.8	88.3	–	87.7	**90.8**
amaze	**99.7**	**99.7**	99.1	93.1	98.8	99.1	97.2	99.4	99.1
band	87.0	86.7	81.7	82.8	85.8	81.4	85.3	83.5	**89.9**
behaviour	95.5	96.3	95.4	96.1	**96.7**	95.3	93.3	95.7	94.5
bet-n	**76.9**	66.3	70.0	71.3	62.5	70.0	70.0	59.4	75.0
bet-v	79.0	78.0	69.0	84.0	77.0	72.0	71.0	84.0	**86.0**
bitter	56.3	55.8	45.8	57.9	52.1	37.4	59.5	50.5	**67.4**
bother	83.4	77.4	72.3	77.7	77.1	77.4	79.1	77.4	**84.3**
brilliant	52.1	51.7	44.8	55.4	52.9	17.3	55.6	47.3	**60.4**
bury	48.2	50.9	34.1	50.9	48.2	37.9	50.3	40.3	**51.8**
calculate	80.0	74.6	75.4	76.8	**81.8**	74.3	81.4	80.0	79.3
consume	62.7	65.5	51.8	67.3	70.9	63.6	64.6	62.7	**71.8**
derive	65.5	59.0	55.5	**70.3**	65.2	65.9	61.0	63.8	67.2
excess	83.8	84.8	83.5	85.5	82.4	**85.9**	84.5	81.4	85.5
float-a	60.0	52.0	74.0	62.0	66.0	70.0	**80.0**	54.0	72.0
float-n	66.7	64.4	47.8	60.0	50.0	46.7	66.7	54.4	**72.2**
float-v	50.0	47.7	36.2	51.2	41.2	44.6	50.0	37.3	**52.7**
generous	43.6	43.0	37.6	51.5	40.6	44.5	**52.1**	40.3	50.9
giant-a	90.3	**92.6**	**92.6**	**92.6**	92.3	**92.6**	90.0	91.9	91.3
giant-n	80.0	77.4	75.8	77.9	80.8	76.3	78.7	76.3	**82.9**
invade	52.5	50.0	37.5	60.0	48.8	41.3	58.8	48.8	**62.5**
knee	**78.3**	71.1	68.5	74.9	71.7	69.2	71.9	65.8	77.5
modest	58.3	59.8	55.6	**65.6**	64.4	38.5	61.2	57.6	63.4
onion	95.0	**97.5**	80.0	80.0	80.0	82.5	82.5	92.5	95.0
promise-n	67.3	67.4	61.3	**73.6**	72.3	67.4	69.2	72.6	73.2
promise-v	89.2	87.3	85.9	87.6	87.3	85.5	88.6	89.4	**92.4**
sack-n	**83.3**	72.5	65.8	71.7	75.8	58.3	77.5	73.3	**83.3**
sack-v	**99.5**	98.4	96.3	96.3	96.3	96.8	97.4	96.3	97.9
sanction	80.0	72.3	71.8	74.6	76.4	61.8	80.9	72.7	**82.7**
scrap-n	82.5	77.5	62.5	77.5	68.8	77.5	78.8	47.5	**87.5**
scrap-v	87.5	92.5	92.5	85.0	**95.0**	**95.0**	92.5	85.0	87.5
seize	60.0	61.5	59.7	62.9	55.0	55.3	58.5	58.5	**63.2**
shake	69.6	68.7	62.6	69.1	67.3	62.2	68.2	67.4	**76.2**
shirt	86.4	85.7	87.2	80.2	87.9	84.6	80.2	83.9	**89.5**
slight	91.9	91.2	90.2	89.5	**92.2**	90.2	90.0	87.8	90.9
wooden	97.6	97.3	97.6	95.4	**97.8**	95.7	97.0	93.5	97.0
best one	5	3	1	4	5	3	2	0	19
avg. rank	3.9	4.8	7.1	4.3	5.0	6.6	4.6	6.4	2.3

Table 6: Generalization accuracies (10CV) for the WSD task. The bottom two rows summarize the table by listing how many times an algorithm was the best one, resp. what its average ranking per word is.

References

Abney, S. (1996). Statistical methods and linguistics. In Klavans, J. L. and Resnik, P., editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pages 1–26. MIT Press, Cambridge, MA.

Aha, D. W., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Baayen, R. H., Piepenbrock, R., and van Rijn, H. (1993). *The CELEX lexical data base on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA.

Bay, S. (1998). Combining nearest neighbor classifiers through multiple feature subsets. In *Proc. 17th Intl. Conf. on Machine Learning*, pages 37–45.

Berger, A., Della Pietra, S., and Della Pietra, V. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1), March 1996.

Blake, C., Keogh, E., and Merz, C. (1998). UCI repository of machine learning databases.

Brill, E. and Wu, J. (1998). Classifier combination for improved lexical disambiguation. In *Proceedings of the Seventeenth International Conference on Computational Linguistics (COLING-ACL'98), Montreal, Canada*, pages 191–195.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974.

Campbell, C. (2000). Algorithmic approaches to training support vector machines: A survey. In *Proceedings of ESANN2000*, pages 27–36. D-Facto Publications, Belgium.

Carlson, A., Cumby, C., Rosen, J., and Roth, D. (1999). SNoW User's Guide. Technical Report UIUC-DCS-R-99-210, University of Illinois at Urbana-Champaign.

Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, Lake Tahoe, California.

Cost, S. and Salzberg, S. (1993). A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

Daelemans, W. (1996). Experience-driven language acquisition and processing. In Van der Avoird, M. and Corsius, C., editors, *Proceedings of the CLS Opening Academic Year 1996-1997*, pages 83–95. CLS, Tilburg.

Daelemans, W., Van den Bosch, A., and Zavrel, J. (1999). Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (2000). TiMBL: Tilburg memory based learner, version 3.0, reference manual, technical report ILK-0001. Technical report, ILK, Tilburg University.

Dietterich, T. and Bakiri, G. (1991). Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proc. of the 9th National Conference on Artificial Intelligence (AAAI-91)*, pages 572–577.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Kittler, J. and Roli, F., editors, *Proc. of the First International Workshop on Multiple Classifier Systems (MCS 2000)*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer Verlag, Berlin.

Escudero, G., Márquez, L., and Rigau, G. (2000). A comparison between supervised learning algorithms for word sense disambiguation. In *Proc. of CoNLL-2000*. ACL.

Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

Johansson, S. (1986). *The tagged LOB Corpus: User's Manual*. Norwegian Computing Centre for the Humanities, Bergen, Norway.

John, G., Kohavi, R., and Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, San Mateo, CA. Morgan Kaufmann.

Jokinen, K. (2000). Learning Dialogue Systems. In *Proceedings of the LREC 2000 Workshop "From*

Spoken Dialogue to Full Natural Interactive Dialogue", pages 13–17.

Kilgarriff, A. and Rosenzweig, J. (2000). Framework and results for english senseval. *Computers and the Humanities, special issue on Senseval*, 34(1–2).

Kool, A., Zavrel, J., and Daelemans, W. (2000). Simultaneous feature selection and parameter optimization for memory-based natural language processing. In *submitted*.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning*, 2:285–318.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1994). *Machine learning, neural and statistical classification*. Ellis Horwood, New York.

Mooney, R. J. (1996). Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 82–91.

Moreira, M. and Mayoraz, E. (1998). Improved pairwise coupling classification with correcting classifiers. In *Proceedings of the 10th European Conference on Machine Learning*, pages 160–171.

Moriarty, D. E. and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5:373–399.

Ng, H. T. and Lee, H. B. (1996). Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proc. of 34th meeting of the Assiociation for Computational Linguistics*.

Prechelt, L. (1998). Early stopping – but when? In *Neural Networks: Tricks of the trade*, Lecture Notes in Computer Science 1524, Springer Verlag, Heidelberg, pages 55–69.

Quinlan, J. (1993). C4.5*: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Ratnaparkhi, A. (1997). A linear observed time statistical parser based on maximum entropy models. Technical Report cmp-lg/9706014, Computation and Language, http://xxx.lanl.gov/list/cmp-lg/.

Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *Proc. of AAAI*.

Roth, D. (2000). Learning in natural language: Theory and algorithmic approaches. In *Proc. of CoNLL'00*. ACL.

Stanfill, C. and Waltz, D. (1986). Toward memory-based reasoning. *Communications of the* ACM, 29(12):1213–1228.

Tjong Kim Sang, E., Daelemans, W., Déjean, H., Koeling, R., Krumolowski, Y., Punyakanok, V., and Roth, D. (2000). Applying system combination to base noun phrase identification. In *Proceedings of COLING 2000*, Saarbrücken, Germany.

Van den Bosch, A. (1997). *Learning to pronounce written words: A study in inductive language learning*. PhD thesis, Universiteit Maastricht.

Van den Bosch, A. (1999). Instance-family abstraction in memory-based learning. In *Proc. of the 16th International Conference on Machine Learning (ICML'99), Bled, Slovenia*, pages 39–48.

Van Halteren, H., Zavrel, J., and Daelemans, W. (1998). Improving data driven wordclass tagging by system combination. In *Proceedings of the Seventeenth International Conference on Computational Linguistics (COLING-ACL'98), Montreal, Canada*, pages 491–497.

Vapnik, V. (1982). *Estimation of Dependencies Based on Empirical Data*. Nauca, Moskow, 1979, Enlish translation: Springer Verlag, New York.

Weiss, S. and Kulikowski, C. (1991). *Computer systems that learn*. San Mateo, CA: Morgan Kaufmann.

White, A. and Liu, W. (1994). Bias in information-based measures in decision tree induction. *Machine Learning*, 15(3):321–329.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

Zell, A., Mamier, G., Vogt, M., et al. (1995). *SNNS, Stuttgart Neural Network Simulator, User Manual*. University of Stuttgart, version 4.1 edition. Technical report 6/95.

Twente Workshops on Language Technology

The TWLT workshops are organised by the PARLEVINK project of the University of Twente. The first workshop was held in Enschede, the Netherlands on March 22, 1991. The workshop was attended by about 40 participants. The contents of the proceedings are given below.

Proceedings Twente Workshop on Language Technology 1 (TWLT 1)
Tomita's Algorithm: Extensions and Applications
Eds. R. Heemels, A. Nijholt & K. Sikkel, 103 pages.

Preface and Contents

A. Nijholt *(University of Twente, Enschede.)* (Generalised) LR Parsing: From Knuth to Tomita.

R. Leermakers *(Philips Research Labs, Eindhoven.)* Recursive Ascent Parsing.

H. Harkema & M. Tomita *(University of Twente, Enschede & Carnegie Mellon University, Pittsburgh)* A Parsing Algorithm for Non-Deterministic Context-Sensitive Languages.

G.J. van der Steen *(Vleermuis Software Research, Utrecht)* Unrestricted On-Line Parsing and Transduction with Graph Structured Stacks.

J. Rekers & W. Koorn *(CWI, Amsterdam & University of Amsterdam, Amsterdam)* Substring Parsing for Arbitrary Context-Free Grammars.

T. Vosse *(NICI, Nijmegen.)* Detection and Correction of Morpho-Syntactic Errors in Shift-Reduce Parsing.

R. Heemels *(Océ Nederland, Venlo)* Tomita's Algorithm in Practical Applications.

M. Lankhorst *(University of Twente, Enschede)* An Empirical Comparison of Generalised LR Tables.

K. Sikkel *(University of Twente, Enschede)* Bottom-Up Parallellization of Tomita's Algorithm.

The second workshop in the series (TWLT 2) has been held on November 20, 1991. The workshop was attended by more than 70 researchers from industry and university. The contents of the proceedings are given below.

Proceedings Twente Workshop on Language Technology 2 (TWLT 2)
Linguistic Engineering: Tools and Products.
Eds. H.J. op den Akker, A. Nijholt & W. ter Stal, 115 pages.

Preface and Contents

A. Nijholt *(University of Twente, Enschede)* Linguistic Engineering: A Survey.

B. van Bakel *(University of Nijmegen, Nijmegen)* Semantic Analysis of Chemical Texts.

G.J. van der Steen & A.J. Dijenborgh *(Vleermuis Software Research, Utrecht)* Lingware: The Translation Tools of the Future.

T. Vosse *(NICI, Nijmegen)* Detecting and Correcting Morpho-syntactic Errors in Real Texts.

C. Barkey *(TNO/ITI, Delft)* Indexing Large Quantities of Documents Using Computational Linguistics.

A. van Rijn *(CIAD/Delft University of Technology, Delft)* A Natural Language Interface for a Flexible Assembly Cell.

J. Honig *(Delft University of Technology, Delft)* Using Deltra in Natural Language Front-ends.

J. Odijk *(Philips Research Labs, Eindhoven)* The Automatic Translation System ROSETTA3.

D. van den Akker *(IBM Research, Amsterdam)* Language Technology at IBM Nederland.

M.-J. Nederhof, C.H.A. Koster, C. Dekkers & A. van Zwol *(University of Nijmegen, Nijmegen)* The Grammar Workbench: A First Step Toward Lingware Engineering.

The third workshop in the series (TWLT 3) was held on May 12 and 13, 1992. Contrary to the previous workshops it had an international character with eighty participants from the U.S.A., India, Great Britain, Ireland, Italy, Germany, France, Belgium and the Netherlands. The proceedings were available at the workshop. The contents of the proceedings are given below.

Proceedings Twente Workshop on Language Technology 3 (TWLT 3)

Connectionism and Natural Language Processing.

Eds. M.F.J. Drossaers & A. Nijholt, 142 pages.

Preface and Contents

L.P.J. Veelenturf *(University of Twente, Enschede)* Representation of Spoken Words in a Self-Organising Neural Net.

P. Wittenburg & U. H. Frauenfelder *(Max-Planck Institute, Nijmegen)* Modelling the Human Mental Lexicon with Self-Organising Feature Maps.

A.J.M.M. Weijters & J. Thole *(University of Limburg, Maastricht)* Speech Synthesis with Artificial Neural Networks.

W. Daelemans & A. van den Bosch *(Tilburg University, Tilburg)* Generalisation Performance of Back Propagation Learning on a Syllabification Task.

E.-J. van der Linden & W. Kraaij *(Tilburg University, Tilburg)* Representation of Idioms in Connectionist Models.

J.C. Scholtes *(University of Amsterdam, Amsterdam)* Neural Data Oriented Parsing.

E.F. Tjong Kim Sang *(University of Groningen, Groningen)* A connectionist Representation for Phrase Structures.

M.F.J. Drossaers *(University of Twente, Enschede)* Hopfield Models as Neural-Network Acceptors.

P. Wyard *(British Telecom, Ipswich)* A Single Layer Higher Order Neural Net and its Application to Grammar Recognition.

N.E. Sharkey & A.J.C. Sharkey *(University of Exeter, Exeter)* A Modular Design for Connectionist Parsing.

R. Reilly *(University College, Dublin)* An Exploration of Clause Boundary Effects in SRN Representations.

S.M. Lucas *(University of Essex, Colchester)* Syntactic Neural Networks for Natural Language Processing.

R. Miikkulainen *(University of Texas, Austin)* DISCERN: A Distributed Neural Network Model of Script Processing and Memory.

The fourth workshop in the series has been held on September 23, 1992. The theme of this workshop was ”Pragmatics in Language Technology”. Its aim was to bring together the several approaches to this subject: philosophical, linguistic and logic. The workshop was visited by more

than 50 researchers in these fields, together with several computer scientists. The contents of the proceedings are given below.

Proceedings Twente Workshop on Language Technology 4 (TWLT 4)

Pragmatics in Language Technology

Eds. D. Nauta, A. Nijholt & J. Schaake, 114 pages.

The fifth workshop in the series took place on 3 and 4 June 1993. It was devoted to the topic "Natural Language Interfaces". The aim was to provide an international platform for commerce, technology and science to present the advances and current state of the art in this area of research.

Proceedings Twente Workshop on Language Technology 5 (TWLT 5)

Natural Language Interfaces

Eds. F.M.G. de Jong & A. Nijholt, 124 pages.

D. Speelman *(University of Leuven)* A Natural Language Interface that Uses Generalised Quantifiers.
R.-J. Beun *(IPO, Eindhoven)* The DENK Program: Modeling Pragmatics in Natural Language Interfaces.
W. Menzel *(University of Hamburg)* ASL: Architectures for Speech and Language Processing
C. Huls & E. Bos *(NICI, Nijmegen)* EDWARD: A Multimodal Interface.
G. Neumann *(University of Saarbrücken)* Design Principles of the DISCO system.
O. Stock & C. Strapparava *(IRST, Trento)* NL-Based Interaction in a Multimodal Environment.

The sixth workshop in the series took place on 16 and 17 December 1993. It was devoted to the topic ”Natural Language Parsing”. The aim was to provide an international platform for technology and science to present the advances and current state of the art in this area of research, in particular research that aims at analysing real-world text and real-world speech and keyboard input.

Proceedings Twente Workshop on Language Technology 6 (TWLT 6)
Natural Language Parsing: Methods and Formalisms
Eds. K. Sikkel & A. Nijholt, 190 pages.

Preface and Contents
A. Nijholt *(University of Twente)* Natural Language Parsing: An Introduction.
V. Manca *(University of Pisa)* Typology and Logical Structure of Natural Languages.
R. Bod *(University of Amsterdam)* Data Oriented Parsing as a General Framework for Stochastic Language Processing.
M. Stefanova & W. ter Stal *(University of Sofia / University of Twente)* A Comparison of ALE and PATR: Practical Experiences.
J.P.M. de Vreught *(University of Delft)* A Practical Comparison between Parallel Tabular Recognizers.
M. Verlinden *(University of Twente)* Head-Corner Parsing of Unification Grammars: A Case Study.
M.-J. Nederhof *(University of Nijmegen)* A Multi-Disciplinary Approach to a Parsing Algorithm.
Th. Stürmer *(University of Saarbrücken)* Semantic-Oriented Chart Parsing with Defaults.
G. Satta *(University of Venice)* The Parsing Problem for Tree-Adjoining Grammars.
F. Barthélemy *(University of Lisbon)* A Single Formalism for a Wide Range of Parsers for DCGs.
E. Csuhaj-Varjú & R. Abo-Alez *(Hungarian Academy of Sciences, Budapest)* Multi-Agent Systems in Natural Language Processing.
C. Cremers *(University of Leiden)* Coordination as a Parsing Problem.
M. Wirén *(University of Saarbrücken)* Bounded Incremental Parsing.
V. Kubon & M. Platek *(Charles University, Prague)* Robust Parsing and Grammar Checking of Free Word Order Languages.
V. Srinivasan *(University of Mainz)* Punctuation and Parsing of Real-World Texts.
T.G. Vosse *(University of Leiden)* Robust GLR Parsing for Grammar-Based Spelling Correction.

The seventh workshop in the series took place on 15 and 16 june 1994. It was devoted to the topic ”Computer-Assisted Language Learning” (CALL). The aim was to present both the state of the art in CALL and the new perspectives in the research and development of software that is meant to be used in a language curriculum. By the mix of themes addressed in the papers

and demonstrations, we hoped to bring about the exchange of ideas between people of various backgrounds.

Proceedings Twente Workshop on Language Technology 7 (TWLT 7)
Computer-Assisted Language Learning
Eds. L. Appelo, F.M.G. de Jong, 133 pages.

Preface and Contents

L.Appelo, F.M.G. de Jong *(IPO / University of Twente)* Computer-Assisted Language Learning: Prolegomena

M. van Bodegom *(Eurolinguist Language House, Nijmegen, The Netherlands)* Eurolinguist test: An adaptive testing system.

B. Cartigny *(Escape, Tilburg, The Netherlands)* Discatex CD-ROM XA.

H.Altay Guvenir, K. Oflazer *(Bilkent University, Ankara)* Using a Corpus for Teaching Turkish Morphology.

H. Hamburger *(GMU, Washington, USA)* Viewpoint Abstraction: a Key to Conversational Learning.

J. Jaspers, G. Kanselaar, W. Kok *(University of Utrecht, The Netherlands)* Learning English with It's English.

G. Kempen, A. Dijkstra *(University of Leiden, The Netherlands)* Towards an integrated system for spelling, grammar and writing instruction.

F. Kronenberg, A. Krueger, P. Ludewig *(University of Osnabruek, Germany)* Contextual vocabulary learning with CAVOL.

S. Lobbe *(Rotterdam Polytechnic Informatica Centrum, The Netherlands)* Teachers, Students and IT: how to get teachers to integrate IT into the (language) curriculum.

J. Rous, L. Appelo *(Institute for Perception Research, Eindhoven, The Netherlands)* APPEAL: Interactive language learning in a multimedia environment.

B. Salverda *(SLO, Enschede, The Netherlands)* Developing a Multimedia Course for Learning Dutch as a Second Language.

C. Schwind *(Universite de Marseille, France)* Error analysis and explanation in knowledge based language tutoring.

J. Thompson *(CTI, Hull, United Kingdom/EUROCALL)* TELL into the mainstream curriculum.

M. Zock *(Limsi, Paris, France)* Language in action, or learning a language by whatching how it works.

The eighth workshop in the series took place on 1 and 2 December 1994. It was devoted to speech, the integration of speech and natural language processing, and the application of this integration in natural language interfaces. The program emphasized research of interest for the themes in the framework of the Dutch NWO programme on Speech and Natural Language that started in 1994.

Proceedings Twente Workshop on Language Technology 8 (TWLT 8)
Speech and Language Engineering
Eds. L. Boves & A. Nijholt, 176 pages.

Preface and Contents

Chr. Dugast *(Philips, Aachen, Germany)* The North American Business News Task: Speaker Independent, Unlimited Vocabulary Article Dictation

P. van Alphen, C. in't Veld & W. Schelvis *(PTT Research, Leidschendam, The Netherlands)* Analysis of the Dutch Polyphone Corpus.

H.J.M. Steenken & D.A. van Leeuwen *(TNO Human factors Research, Soesterberg, The Netherlands)* Assessment of Speech Recognition Systems.

J.M. McQueen *(Max Planck Institute, Nijmegen, The Netherlands)* The Role of Prosody in Human Speech Recognition.

L. ten Bosch *(IPO, Eindhoven, the Netherlands)* The Potential Role of Prosody in Automatic Speech Recognition.

P. Baggia, E. Gerbino, E. Giachin & C. Rullent *(CSELT, Torino, Italy)* Spontaneous Speech Phenomena in Naive-User Interactions.

M.F.J. Drossaers & D. Dokter *(University of Twente, Enschede, the Netherlands)* Simple Speech Recognition with Little Linguistic Creatures.

H. Helbig & A. Mertens *(FernUniversität Hagen, Germany)* Word Agent Based Natural Language Processing.

Geunbae Lee et al. *(Pohang University, Hyoja-Dong, Pohang, Korea)* Phoneme-Level Speech and natural Language Integration for Agglutinative Languages.

K. van Deemter, J. Landsbergen, R. Leermakers & J. Odijk *(IPO, Eindhoven, The Netherlands)* Generation of Spoken Monologues by Means of Templates

D. Carter & M. Rayner *(SRI International, Cambridge, UK)* The Speech-Language Interface in the Spoken Language Translator

H. Weber *(University of Erlangen, Germany)* Time-synchronous Chart Parsing of Speech Integrating Unification Grammars with Statistics.

G. Veldhuijzen van Zanten & R. op den Akker *(University of Twente, Enschede, the Netherlands)* More Efficient Head and Left Corner Parsing of Unification-based Formalisms.

G.F. van der Hoeven et al. *(University of Twente, Enschede, the Netherlands)* SCHISMA: A natural Language Accessible Theatre Information and Booking System.

G. van Noord *(University of Groningen, the Netherlands)* On the Intersection of Finite Staate Automata and Definite Clause Grammars.

R. Bod & R. Scha *(University of Amsterdam, the Netherlands)* Prediction and Disambiguation by Means of Data-Oriented Parsing.

The ninth workshop in the series took place on 9 June 1995. It was devoted to empirical methods in the analysis of dialogues, and the use of corpora of dialogues in building dialogue systems. The aim was to discuss the methods of corpus analysis, as well as results of corpus analysis and the application of such results.

Proceedings Twente Workshop on Language Technology 9 (TWLT 9)

Corpus-based Approaches to Dialogue Modelling

Eds. J.A. Andernach, S.P. van de Burgt & G.F. van der Hoeven, 124 pages.

J.H. Connolly, A.A. Clarke, S.W. Garner & H.K. Palmén *(Loughborough University of Technology, UK)* Clause-internal structure in spoken dialogue.

J. Carletta, A. Isard, S. Isard, J. Kowtko, G. Doherty-Sneddon & A. Anderson *(HCRC, Edinburgh, UK)* The coding of dialogue structure in a corpus.

J. Alexandersson & N. Reithinger *(DFKI, Saarbrücken, Germany)* Designing the dialogue component in a speech translation system - a corpus-based approach.

H. Aust & M. Oerder *(Philips, Aachen, Germany)* Dialogue control in automatic inquiry systems.

M. Rats *(ITK, Tilburg, the Netherlands)* Referring to topics - a corpus-based study.

H. Dybkjær, L. Dybkjær & N.O. Bernsen *(Centre for Cognitive Science, Roskilde, Denmark)* Design, formalization and evaluation of spoken language dialogue.

D.G. Novick & B. Hansen *(Oregon Graduate Institute of Science and Technology, Portland, USA)* Mutuality strategies for reference in task-oriented dialogue.

N. Fraser *(Vocalis Ltd, Cambridge, UK)* Messy data, what can we learn from it?

J.A. Andernach *(University of Twente, Enschede, the Netherlands)* Predicting and interpreting speech acts in a theatre information and booking system.

The tenth workshop in the series took place on 6-8 December 1995. This workshop was organized in the framework provided by the Algebraic Methodology and Software Technology movement (AMAST). It focussed on algebraic methods in formal languages, programming langauges and natural languages. Its aim was to bring together those researchers on formal language theory, programming language theory and natural language description theory, that have a common interest in the use of algebraic methods to describe syntactic, semantic and pragmatic properties of language.

Proceedings Twente Workshop on Language Technology 10 (TWLT 10)

Algebraic Methods in Language Processing

Eds. A. Nijholt, G. Scollo & R. Steetskamp, 263 pages.

Vladimir A. Fomichov *(Moscow, R)* A Variant of a Universal Metagrammar of Conceptual Structures. Algebraic Systems of Conceptual Syntax.

Theo M.V. Jansen *(Amsterdam, NL)* The Method of ROSETTA, Natural Language Translation Using Algebras.

C.M. Martín-Vide, J. Miquel-Verges & Gh. Paun *(Tarragona, E)* Contextual Grammars with Depth-First Derivation.

Pál Dömösi & Jürgen Duske *(Kussuth University H, University of Hannover, G)* Subword Membership Problem for Linear Indexed Languages.

C. Rico Perez & J.S. Granda *(Madrid, E)* Algebraic Methods for Anaphora Resolution.

Vincenzo Manca *(Pisa, I)* A Logical Formalism for Intergrammatical Representation.

The eleventh workshop in the series took place on 19-21 June 1996. It focussed on the task of dialogue management in natural-language processing systems. The aim was to discuss advances in dialogue management strategies and design methods. During the workshop, there was a separate session concerned with evalutation methods.

Proceedings Twente Workshop on Language Technology 11 (TWLT 11)

Dialogue Management in Natural Language Systems

Eds. S. LuperFoy, A. Nijholt and G. Veldhuijzen van Zanten, 228 pages.

Preface and Contents

David R. Traum *(Université de Genève, CH)* Conversational Agency: The TRAINS-93 Dialogue Manager.

Scott McGlashan *(SICS, SW)* Towards Multimodal Dialogue Management.

Pierre Nugues, Christophe Godéreaux, Pierre-Olivier and Frédéric Revolta *(GREYC, F)* A Conversational Agent to Navigate in Virtual Worlds.

Anne Vilnat *(LIMSI-CNRS, F)* Which Processes to Manage Human-Machine Dialogue?

Susann LuperFoy *(MITRE, USA)* Tutoring versus Training: A Mediating Dialogue Manager for Spoken Language Systems.

David G. Novick & Stephen Sutton *(Portland, USA)* Building on Experience: Managing Spoken Interaction through Library Subdialogues.

Latifa Taleb *(INRIA, F)* Communicational Deviation in Finalized InformativeDialogue Management.

Robbert-Jan Beun *(IPO, NL)* Speech Act Generation in Cooperative Dialogue.

Gert Veldhuijzen van Zanten *(IPO, NL)* Pragmatic Interpretation and Dialogue Management in Spoken-Language Systems.

Joris Hulstijn, René Steetskamp, Hugo ter Doest, Anton Nijholt & Stan van de Burgt *(University of Twente, NL & KPN Research, NL)* Topics in SCHISMA Dialogues.

Gavin Churcher, Clive Souter & Eric S. Atwell *(Leeds University, UK)* Dialogues in Air Traffic Control

Elisabeth Maier *(DFKI, D)* Context Construction as Subtask of Dialogue Processing – the VERBMOBIL Case.

Anders Baekgaard *(CPK, DK)* Dialogue Management in a Generic Dialogue System.

Wayne Ward *(Carnegie Mellon University, USA)* Dialog Management in the CMU Spoken Language Systems Toolkit.

Wieland Eckert *(University of Erlangen, D)* Understanding of Spontaneous Utterances in Human-Machine-Dialog.

Jan Alexandersson *(DFKI, D)* Some Ideas for the Automatic Acquisition of Dialogue Structure.

Kristiina Jokinen *(Nara Institute of Science and Technology, JP)* Cooperative Response Planning in CDM: Reasoning about Communicative Strategies.

Elizabeth Hinkelman *(Kurzweil Applied Science, USA)* Dialogue Grounding for Speech Recognition Systems.

Jennifer Chu-Carroll *(University of Delaware, USA)* Response Generation in Collaborative Dialogue Interactions.

Harry Bunt *(Tilburg University, NL)* Interaction Management Functions and Context Representation Requirements.

Peter Wyard & Sandra Williams *(BT, GB)* Dialogue Management in a Mixed-Initiative, Cooperative, Spoken Language System.

Rolf Carlson *(KTH, SW)* The Dialog Component in the Waxholm System.

Laila Dybkjær, Niels Ole Bernsen & Hans Dybkjaer *(Roskilde University, DK)* Evaluation of Spoken Dialogue Systems.

Vincenzo Manca *(Pisa, I)* A Logical Formalism for Intergrammatical Representation.

TWLT 12 took place on 11-14 September 1996. It focussed on 'computational humor' and in particular on verbal humor. TWLT12 consisted of a symposium (Marvin Minsky, Douglas Hofstadter, John Allen Paulos, Hugo Brandt Corstius, Oliviero Stock and Gerrit Krol as main speakers), an essay contest for computer science students, two panels, a seminar organized by Salvatore Attardo and Wladyslaw Chlopicki and a two-day workshop (Automatic interpretation and Generation of Verbal Humor) with a mix of invited papers and papers obtained from a Call for Papers.

Proceedings Twente Workshop on Language Technology 12 (TWLT 12)

Computational Humor: Automatic Interpretation and Generation of Verbal Humor

Eds. J. Hulstijn and A. Nijholt, 208 pages.

Tony Veale & Mark Keane Bad Vibes Catastrophes of Goal Activation in the Appreciation of Disparagement Humour and General Poor Taste.

Kim Binsted & Graeme Ritchie Speculations on Story Pun.

Dan Loehr An Integration of a Pun Generator with a Natural Language Robot.

Cameron Shelley, Toby Donaldson & Kim Parsons Humorous Analogy: Modeling 'The Devils Dictionary'.

Michal Ephratt More on Humor Act: What Sort of Speech Act is the Joke?

TWLT 13 took place on 13-15 May 1998. It was the follow-up of the Mundial workshop, that took place in München in 1997. Both the Mundial workshop as TWLT13 focussed on the formal semantics and pragmatics of dialogues. In addition to the three-day workshop in Twente, with invited and accepted papers, on 18 May a workshop titled 'Communication and Attitudes' was organized at ILLC/University of Amsterdam.

Proceedings Twente Workshop on Language Technology 13 (TWLT 13)

Formal Semantics and Pragmatics of Dialogue (Twendial'98)

Eds. J. Hulstijn and A. Nijholt, 274 pages.

TWLT14 was held on 7-8 December 1998. It focussed on the role of human language technology in the indexing and accessing of written and spoken documents, video material and/or images, and on the role of language technology for cross-language retrieval and information extraction. The workshop consisted of a series of accepted papers.

Proceedings Twente Workshop on Language Technology 14 (TWLT 14)
Language Technology in Multimedia Information Retrieval
Eds. D. Hiemstra, F.M.G. de Jong and K. Netter, 194 pages.

Marten den Uyl, Ed S. Tan, Heimo Müller & Peter Uray *(SMR Amsterdam, Vrije Universiteit Amsterdam, Joanneum Research)* Towards Automatic Indexing and Retrieval of Video Content: the VICAR system

Anton Nijholt *(University of Twente)* Access, Exploration and Visualization of Interest Communities: The VMC Case Study (in Progress)

Joanne Capstick, Abdel Kader Diagne, Gregor Erbach & Hans Uszkoreit *(DFKI, Saarbrücken)* MULINEX: Multilingual Web Search and Navigation

Klaus Netter & Franciska de Jong *(DFKI, Saarbrücken and University of Twente)* OLIVE: speech based video retrieval

Franciska de Jong *(University of Twente)* Twenty-One: a baseline for multilingual multimedia

---

TWLT15 was held on 19-21 May 1999. It focussed on the interactions in Virtual World. Contributions were invited on theoretical, empirical, computational, experimental, anthropological or philosophical approaches to VR environments. Invited talks were given by Russell Eames *(Microsoft)*, Lewis Johnson (USC), James Lester (North Carolina State University), Pierre Nugues (ISMRA-Caen) and Stephan Matsuba (VRML Dream Company)

Proceedings Twente Workshop on Language Technology 15 (TWLT 15)

Interactions in Virtual Worlds

Eds. Anton Nijholt, Olaf Donk and Betsy van Dijk, 240 pages.

Rainer Kuhn & Sigrun Gujonsdottir *(University of Karlsruhe)* Virtual Campus Project – A Framework for a 3D Multimedia Educational Environment

James C. Lester *(North Carolina State University)* Natural Language Generation in Multimodal Learning Environments

Stephen N. Matsuba *(The VRML Dream Company, Vancouver)* Lifelike Agents and 3D Animated Explanation Generation Speaking Spaces: Virtual Reality, Artificial Intelligence and the Construction of Meaning

Pierre Nugues *(ISMRA-Caen)* Verbal and Written Interaction in Virtual Worlds – Some application examples

Anton Nijholt *(University of Twente)* The Twente Virtual Theatre Environment: Agents and Interactions

Sandy Ressler, Brian Antonishek, Qiming Wang, Afzal Godil & Keith Stouffer *(National Institute of Standards and Technology)* When Worlds Collide –Interactions between the Virtual and the Real

Lakshmi Sastry & D. R. S. Boyd *(Rutherford Appleton Laboratory)* EISCAT Virtual Reality Training Simulation: A Study on Usability and Effectiveness

Frank Schaap *(University of Amsterdam)* "Males say 'blue,' females say 'aqua,' 'sapphire,' and 'dark navy'" The Importance of Gender in Computer-Mediated Communication

Boris van Schooten, Olaf Donk & Job Zwiers *(University of Twente)* Modelling Interaction in Virtual Environments using Process Algebra

Martijn J. Schuemie & Charles A. P. G. van der Mast *(Delft University of Technology)* Presence: Interacting in Virtual Reality?

Jarke J. van Wijk, Bauke de Vries & Cornelius W. A. M. van Overveld *(Eindhoven University of Technology)* Towards an Understanding 3D Virtual Reality Architectural Design System

Peter J. Wyard & Gavin E. Churcher *(BT Laboratories, Ipswich, Suffolk)* Spoken Language Interaction with a Virtual World in the MUeSLI Multimodal 3D Retail System

J. M. Zheng, K. W. Chan & I. Gibson *(University of Hong Kong)* Real Time Gesture Based 3D Graphics User Interface for CAD Modelling System

---

TWLT16/AMiLP 2000 is the second AMAST workshop on Algebraic Methods in Language Processing. Like its predecessor, organized in 1995 at the University of Twente in the TWLT series, papers were presented on formal language theory, programming language theory and natural language theory. A common theme in these papers was the use of mathematics, in particular the use of an algebraic approach. AMiLP 2000 was held in Iowa City, Iowa, USA, from May 20–22 2000, just before the AMAST 2000 conference.

Proceedings Twente Workshop on Language Technology 16 (TWLT 16)

Algebraic Methods in Language Processing (AMiLP2000)

Eds. D. Heylen, A. Nijholt, and G. Scollo, 274 pages.

Denys Duchier *(Universität des Saarlandes)* A Model Eliminative Treatment of Quantifier-Free Tree Descriptions

Theo M.V. Janssen *(University of Amsterdam)* An Algebraic Approach to Grammatical Theories for Natural Languages

Aravind K. Joshi *(University of Pennsylvania)* Strong Generative Power of Formal Systems

Jozef Kelemen, Alica Kelemenov & Carlos Martin-Vide *(Silesian University & Rovira I Virgili University)* On the Emergence of Infinite Languages from Finite Ones

Stephan Kepser *(University of Tübingen)* A Coalgebraic Modelling of Head-Driven Phrase Structure Grammar

Hubert Dubois & Hélène Kirchner *(LORIA-UHP & CNRS)* Objects, Rules and Strategies in ELAN

Jens Michaelis & Uwe Mönnich & Frank Morawietz *(Universität Tübingen)* Algebraic Description of Derivational Minimalism

Rani Nelken & Nissim Francez *(Dept. of Compute Science, Technion, Israel)* A Calculus for Interrogatives Based on Their Algebraic Semantics

Gheorghe Păun *(Institute of Mathematics of the Romanian Academy)* Molecular Computing and Formal Languages: A Mutually Beneficial Cooperation

G. Reggio, M. Cerioli & E. Astesiano *(University of Genova)* An Algebraic Semantics of UML Supporting its Multiview Approach

James Rogers *(University of Central Florida)* wMSO Theories as Grammar Formalisms.

Teodor Rus *(University of Iowa)* Algebraic Definition of Programming Languages.

Karl-Michael Schneider *(University of Passau)* Tabular Parsing and Algebraic Transformations

Edward Stabler & Ed Keenan *(University of California)* Structural Similarity

Thomas L. Cornell *(Cymfony Net, Inc.)* Parsing and Grammar Engineering with Tree Automata.

TWLT 17, *Interacting Agents*, was co-organised with the Centre for Evolutionary Language Engineering (CELE) and as such is also the first workshop in the CELE Workshops on Evolutionary Language Engineering (CEvoLE) series. Together with TWLT18/CEvoLE2 these workshops are jointly titled "Learning to Behave". The workshops investigate human-agent interaction and knowledge both on the level of agents communicating with the external environment and on the level of the internal agent processes that guide the modelling and understanding of the external sensory input in the brain.

TWLT 17 focussed on the interaction of an agent with the environment. This covers many topics such as the use of conversational strategies, natural language and non-verbal communication, turn-taking, protocols, cross-media references, the effect of context, affect and emotion in agents. A special session is devoted to theatre applications. TWLT 17 was held in Enschede from October 18–20 2000.

Proceedings Twente Workshop on Language Technology 17 (TWLT 17)

Learning to Behave, Workshop I: Interacting Agents

Eds. A. Nijholt, D. Heylen and K. Jokinen, 205 pages.

Preface and Contents

Nadia Magnenat-Thalmann, Sumedha Kshirsagar *(MIRALab, CUI, University of Geneva)* Communication with Autonomous Virtual Humans

Sabine S. Geldof *(AI-Lab, Vrije Universiteit Brussel)* Co-habited Mixed Reality and Context

Zsófia Ruttkay, Jeroen Hendrix, Paul ten Hagen, Alban Lelièvre, Han Noot & Behr de Ruiter *(Centre for Mathematics and Computer Science, Amsterdam)* A Facial Repertoire for Avatars

Kristina Höök *(SICS, Kista, Sweden)* Evaluating Interactive Characters – going beyond body language

Frédéric Kaplan *(Sony Computer Science Laboratory, Paris)* Talking AIBO : First Experimentation of Verbal Interactions with an Autonomous Four-legged Robot

Jan-Thorsten Milde *(Universität Bielefeld)* The Instructable Agent Lokutor

Patrizia Paggio & Bart Jongejan *(Center for Sprogteknologi, Copenhagen)* Unification-Based Multimodal Analysis in a 3D Virtual World: the Staging Project

Boris van Schooten *(Parlevink Group, University of Twente, Enschede)* A Specification Technique for Building Interface Agents in a Web Environment

Bernard Spanlang, Tzvetomir I Vassilev *(Department of Computer Science, University College, London)* Efficient Cloth Model for Dressing Animated Virtual People

Luc van Tichelen and Alex Schoenmakers *(Lernout & Hauspie Speech Products, Ieper)* A Conversational Agent for a Multi-Modal Information Kiosk

Job Zwiers, Betsy van Dijk, Anton Nijholt & Rieks op den Akker *(Parlevink/CTIT, University of Twente, Enschede)* Design Issues for Intelligent Navigation and Assistance Agents in Virtual Environment

Marc Cavazza *(University of Teesside)* Mapping Dialogue Acts to Narrative Functions for Interactive Storytelling

Ricardo Imbert & Angélica de Antonio *(Universidad Politénica de Madrid)* The Bunny Dilemma: Stepping Between Agents and Avatars

Patrizia Palamidese *(CNR-CNUCE, Pisa)* Composing and Editing Structured 3D Stories

Rachel Price, Chraig Douther, Mervyn A. Jack *(Center for CIR, University of Edinburgh)* Using Choreographed Electronic Motion to Create Mood in Virtual Environments

Nikitas M. Sgouros *(University of Piraeus)* Empowering the Audience in Virtual Performances

Elisabeth André *(DFKI, Germany)* Adding Lifelike Synthetic Charaters to the Web

Roel Vertegaal *(Queens University, Kingston)* Agents as Attention-based Interfaces

---

The proceedings of the workshops can be ordered from Leerstoel TKI, Department of Computer Science, University of Twente, P.O. Box 217, NL-7500 AE Enschede, The Netherlands. E-mail orders are possible: bijron@cs.utwente.nl. Each of the proceedings costs NLG. 50,=.