Topics in SCHISMA Dialogues[1]

Joris Hulstijn, René Steetskamp, Hugo ter Doest, Stan van de Burgt and Anton Nijholt
University of Twente[2]
{joris | steets | terdoest | stan | anijholt}@cs.utwente.nl

Abstract

An important part of dialogue management in a dialogue system is *topic management*: The system has to keep track of topic aspects (current topic, context, obligations, ...) in order to be able to resolve ambiguities and to respond in a cooperative and sensible way.

In the SCHISMA project, we are developing a system that takes part in a conversation about theater performances and is able to make ticket reservations. In this paper we discuss several techniques for topic management that we developed and tested this year.

1 Introduction

In a dialogue system one of the main tasks is answering questions that the user states, either explicitly or implicitly. Since the question is often based on, or referring to, earlier utterances of either speaker, analyzing only the last user utterance will not suffice in most cases. This conjecture is verified in the SCHISMA project, a cooperative project of Twente University and KPN Research in the Netherlands. In this project the SCHISMA system is developed. It provides the user with information on theater performances, and optionally sells the user one or more tickets for a given performance (Hoeven et al.1995). The system started as a pure, human driven, *Wizard of Oz* (WoZ) system to which several additions were made as time went by and more subsystems became available. This way, the system evolves to the final system we envision. Currently, parts of the generator and the dialogue management modules are incorporated, while pre -processing modules will be incorporated later this year

In an earlier phase of the project a corpus of dialogues was collected, which serves as a basis for much of the work presented in this paper. A dialogue is considered here to be a sequence of turns of two speakers: the *user* and the *system*. A *turn* is an uninterrupted sequence of words by one speaker. A turn consists of one or more *utterances*: linguistically identifiable units. Typically, the typed utterances in our corpus are marked by punctuation and conjunctives.

Responding to a user utterance typically involves more information than can be found in the utterance itself. We argue that the response is mainly determined by the following parameters: *context*, *topic*, *utterance type* (UT), *expected response type* (ERT) and *dialogue phase*.

The *context* incorporates both information extracted from the dialogue and domain knowledge. We choose to organize the context around the notion of *topic*, the entity the dialogue is currently *about* (see section 4). This approach is compared to other approaches in section 5. In section 6 the implementation of the context and related modules is discussed.

Other parameters that guide the behavior of the dialogue manager are *utterance type* (UT) and *expected response type* (ERT). The utterance type indicates the grammatical type of a sentence or the lexical type of a constituent (e.g. declarative, nominal, interrogative, adverbial). The expected response type indicates what the user expects the system response to be about, based on the form of the utterance. For instance, a *when* question indicates that the user expects the system to respond with a time or date.

These parameters can be detected using syntactic and lexical cues from the user utterance, the previous topic and domain knowledge. How detection works is discussed in section 7.

SCHISMA is a mixed initiative system. The initiative shifts between various *phases* of the dialogue. Apart from the obvious *opening* and *closing* phases, we distinguish a *information*, a *se-*

[1] This paper has been presented at the TWLT11 workshop on *Dialogue Management in Natural Language Systems*, june 1996, Enschede, The Netherlands.

[2] Department of Computer Science, University of Twente, PO BOX 217, 7500 AE Enschede, The Netherlands

lection, a *reservation* and a *confirmation* phase. During the first two phases, information and selection, the user has the initiative: the user requests information about performances, which is subsequently provided for by the system. But during the reservation phase, the system takes more initiative. It asks the user for information items like name and address that is needed to make a reservation. After this, the system lists the reservation details (performance, price) and asks for confirmation.

The reservation phase presupposes that a particular performance is selected. So there is no obligatory sequential order among phases, just a logical one. It is very well possible for a user to go through the first three phases in one go, for instance by asking *I want to reserve a ticket for tonight.* When the information provided by an utterance is not enough to determine a unique performance, the system takes over initiative, asking the user for more details. So, the *phase* of the dialogue has a profound impact on the response behavior of the system.

The architecture of the SCHISMA dialogue manager has progressed from a *finite state-based* to an *utterance-based* architecture. One of the reasons for this development is that the huge number of states for a nontrivial dialogue automaton, makes manual construction unfeasible (Bos 1995); some general principles are needed to automatically generate the automaton. Once principles are used, there is no longer a conceptual need for finite-state techniques[1]. This paper describes the first step in determining useful principles for our domain. Some of them will be applicable to any information-dialogue. But some will be particular to our theater domain.

In section 2 we introduce the architecture of our system, followed by the principles of the grammar in section 3 and the role of context and topic in section 4. The implementation is explained in section 6. In section 7 we report on some of our ongoing corpus-based research into ways of determining utterance type, topic and expected response type. Section 8 summarizes our findings.

2 Architecture

The SCHISMA system basically consists of three modules: a *morphological analyzer*, a *parser* and a *dialogue manager*. These interact with a *database*. (figure 1) Static knowledge sources are depicted using round cornered boxes. Actual data-flow is depicted by arrows; conceptual influence by dashed arrows. In principle this cycle is repeated for each user utterance. Therefore, we call it an utterance-based architecture.

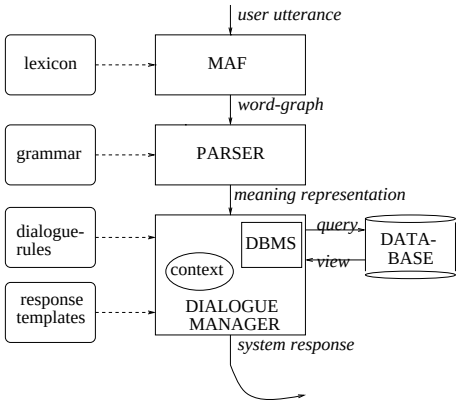

Figure 1: SCHISMA architecture

The user produces an utterance, which is scanned by the *morphological analyzer*, *MAF*. In this way, typing errors may be corrected and proper names can be recognized. In the theater domain we have a lot of names or titles that read like a normal constituent. Since we cannot rely on the user to use capitals and quotes consistently, we have to recognize these beforehand. We do not want to end up with *Twelfth Night* as an indication of date. The result of MAF is a word-graph; in most cases it turns out to be a partially annotated string.[2]

This annotated string is transformed by the *parser* into a meaning representation in the form of an *item list*. An *item list* is a list of *information items*: objects and attributes that are important in the domain. Typical items are `time`, `artist` or `price`. In the item list, items are stored with their values and flags indicating the status of the information (see section 6). Item lists represent the domain information that the parser has been able to extract and that is relevant for the dialogue manager. Information that is not in the lexicon or that cannot be parsed is neglected. The item list is a simplified version of the feature

[1] Compare (Aust and Oerder 1995) who reach similar conclusions.

[2] We accept word-graphs because we may want our system to be able to deal with spoken language input in a later stage. Speech recognition modules usually produce word-graphs.

structure that is produced by the parser.

The parser behaves according to a *unification based grammar*. We believe that the design of the grammar should focus on performance, not linguistic competence. Therefore, the grammar should ideally produce at most and at least one parse. See section 3.

The *dialogue manager* determines the appropriate response action based on the user-utterance and on the context. The dialogue manager also updates the context with information from the user utterance. The dialogue manager behaves according to a number of *dialogue rules*. These specify for each state of the dialogue what response actions should be taken. A response action may be a combination of three kinds of actions: a search in the database followed by a generated answer, a request for more user-information, or a control utterance like 'thanks' or 'good bye'. Some response actions require more than one turn. For instance, a reservation involves a sub-dialogue about the number of tickets. The dialogue manager keeps a stack of *actions pending* to plan and control actions. The shape of system responses is determined by a set of *response templates*.

3 Grammar

In developing a grammar and parser for the SCHISMA system, performance, contrary to competence, is the central issue. We consider linguistic insights as a means for domain-specific meaning extraction rather than a goal in itself. This type of language modeling is sometimes referred to as *semantic grammar*.[3]

Our parsing system should process any input. The grammar must not be too strict, or otherwise it may fail to parse. On the other hand it should extract as much domain-relevant information as can be recovered without the dialogue context. So, the parser should map its input onto preferably at least and at most *one* meaning representation.

A good example of the difficulties in parsing natural dialogue is (1). Traditionally it would not be considered grammatical. Yet we can extract valuable information from it. It is a question, indicated by the presence of the question-mark. It connects to a previous question, indicated by the conjunct *en* and contains a specification of a date, *op donderdag*. The information is indeed present in the feature structure produced by the parser. The `schisma` part of the feature structure will be turned into an item list and merged with the context.[4]

(1) en op donderdag ? (011)
and on thursday ?

```
Z:
[ cat:   Z
  head:    [ first:[ canbesubj:-
                     prep:OP ] ]
  schisma:[ mode:[ questionmark:+ ]
            time:[ week: [ thursday:+ ]
                   coordination: OP ] ] ]
```

The one-parse requirement has some implications. Ambiguity should be represented by underspecified values; not by sets of possible readings, as is traditionally done. The dialogue manager needs to interpret the meaning representation in context, filling in the variables. For example, in (1) the dialogue manager has to recognize the utterance as a continuation of a previous question, substituting the focus of that question with the new 'thursday' date.

If no alternative parses are allowed, introduction of lexical and structural ambiguity should be allowed only if it can be accounted for at a later stage in the parsing process. This can be done by applying both syntactic and semantic constraints. The syntactic constraints used at the moment are agreement and some word order constraints. Word order is mainly used to determine the utterance type. Semantic constraints are derived from the task domain. For instance, tickets can only be reserved by system and user, excluding an interpretation of (2) where *kaartjes* is subject.[5]

(2) kaartjes reserveren (322)
reserve tickets/tickets reserve

Semantic constraints are especially important for 'telegram style' utterances, like (2). We expect that experienced users of a keyboard inquiry system like ours will often use shortened utterances with little or no syntactic structure. We find a number of telegram style dialogues in our

[3] We consider a grammar semantic when both general syntactic and domain-specific semantic constraints are used interactively (Androutsopoulos and Thanisch 1994). Compare with the *conceptual parser* of (Sowa 1984).

[4] Except where indicated otherwise, the examples in this paper are taken from the corpus. The number between brackets indicates the line number. English translations are literal.

[5] The Dutch phrase is an imperative, using the infinitive. *kaartjes* is direct object.

corpus. In most cases a human interpreter could easily reconstruct the meaning from the context and the theater environment. The nature of the system implies that users will want information or tickets for a certain performance. This is build into the expectations of the system.

Semantic constraints are also important to deal with structural ambiguity. The following sentence, is structurally ambiguous. (example 3) The prepositional phrase *voor morgen* can modify *kaartjes*: tickets for tomorrows performance. But it might also modify the verb: reserve before tomorrow. Given our domain, only the first interpretation remains.

(3) Kan ik kaartjes voor morgen reserveren. (345)
Can I reserve tickets for/before tomorrow.

In the course of writing the SCHISMA grammar the following issues kept coming up: (1) exactly how much syntactic structure is needed, and (2) how can domain knowledge best be integrated with the grammar?

With respect to the first issue, the best way to find out is to experiment with different setups. For these experiments in writing a SCHISMA grammar, we use the left-corner parser developed at SIL (McConnel 1995, pcpatr). It is a straightforward implementation of PATR-II (Shieber 1986) and it can be easily incorporated in the SCHISMA system.

With respect to the second issue, we are planning corpus-based research into subcategorization patterns and corresponding semantic roles in our domain. Domain specific information can be applied in a *typed* unification-based grammar. Both linguistic and conceptual knowledge are represented in *type-hierarchies*. In (Steetskamp 1996) such a *semantic parser* for a fragment of the SCHISMA domain is described. It makes use of the head-corner parser generator for typed feature structures that was developed at our department (Moll 1995).

4 Context and Topic

Our corpus of information dialogues contains heavily context-dependent types of utterance. Answers depend on questions, anaphora depend on their antecedent and elliptical expressions depend on the previously uttered phrases. See examples (4), (5) and (6) respectively.

(4) S: Hoeveel kaartjes wilt u? (622)
How many tickets would you like.
U: 4

(5) U: Daar wil ik wel heen.(153)
wanneer spelen zij?
I'd like to go there
When are they performing?
S: Op 7 januari kunt U naar 'Cocktail'.
On the 7th of januari you can see 'Cocktail'

(6) U: En Othello?
and Othello?

We believe that the *information structure* of the context is just as important as the *information content* in resolving context dependencies. The information structure of the dialogue context is modeled using *topic*. The topic specifies what the dialogue at that point *is about*. With each topic some information items are naturally associated. For instance, the topic *performance* often co-occurs with a *date*, an *artist* or *group* and a performance *title* (see figure 2). So, the topic specifies a *frame* of information-items associated with the topic. Topic related ideas have been used widely in dialogue systems. Recently, (Veldhuijzen van Zanten 1996; Smith et al. 1995; Deemter et al. 1994). Classical work on dialogue systems and topic is done by Grosz and others at CRI. (Grosz 1977) They propose *conceptual spaces* associated with a topic in the same role as our frames. In the psychology and AI research traditions topic is often called *focus of attention*. Topic has the same role as the *backward looking center* of the centering approach. (Grosz and Sidner 1986; Grosz et al. 1995) In section 5 comparisons are made.

Some items associates with a topic, may function as a topic themselves. Such *subtopics* have a frame of their own. So, we get a hierarchical frame structure. The current version of the frame structure is isomorphic to the design of our database. A frame structure can be seen as an object oriented data model. Potential topics correspond to names of object classes, items correspond to attributes.

The decision to model the information structure of the context by the frame structure that models the task domain, reflects our assumption that the structure of the dialogue itself is largely determined by the task. Unlike types of dialogue that are about actions themselves, like instructional dialogues (Grosz 1977; Smith et al. 1995),

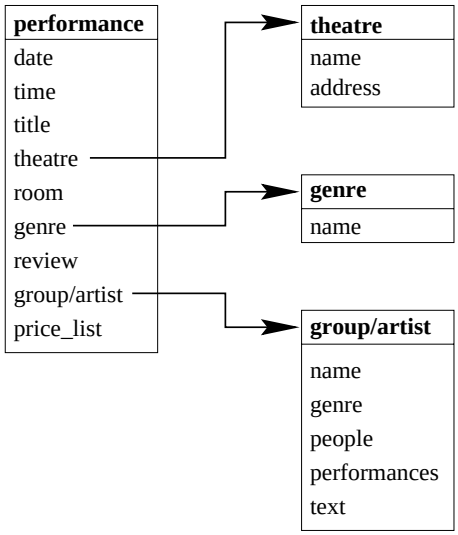

Figure 2: Fragment of Frame Structure

our task provides little *procedural* information. It does give logical dependencies between elements, which may have an impact on the order in which items are best dealt with.

Most actions for this type of information dialogues are of the form `ask_attribute`, `update_attribute` or `show_attribute`. At the moment we have no formal way of specifying other procedural knowledge. One way, would be to use AI-style *plans* (Allen et al. 1991). Some of the intuitions behind plans have already been coded in our dialogue phases. Similar considerations can be found in (Aust and Oerder 1994). They propose a limited dialogue control structure surrounding a *slot-filling* mechanism with little or no procedural preference.

5 Other Approaches to Topic

The notion of *topic* is problematic in the literature. Different research traditions use it in different ways. Our notion of topic compares best to the notion used by Mieke Rats in her analysis of a corpus of information dialogues[6]

> An entity T is the topic of an utterance U if U is intended to increase the addressee's knowledge about T, request information about T or otherwise get the addressee to act with respect to T. (Rats 1996, p 37)

Our use of topic differs from other well-known approaches. First, it is not a question-based approach (Kuppevelt 1995). According to Kuppevelt, topic is intimately related with the most salient question that needs answering at that point in the discourse: what question is it that makes the sentence relevant at this point? The question may have been asked explicitly, as in most information dialogues, or it may be raised implicitly. Topic is defined as the (type of) entity that would provide an answer to the most salient question.

We found the question-related notion of topic theoretically appealing. It gives a nice account of *relevance*. However, we found that it did not provide enough detail for successful annotation of the corpus. It proved to be very difficult to decide what the most salient question would be at a certain point. Moreover, wh-questions comprise over a quarter of the corpus, so they are important enough to introduce a separate theoretical notion to account for them. This is why we have *expected response type* among the parameters that determine the dialogue manager's behavior (section 7).

Second, our notion differs from the Prague school approach (e.g. (Hajicova et al. 1995)). The Prague school assumes a strong link between topic and contextually given information. New or contrastive information, on the other hand, is said to be in *focus*. Elements in focus often get the sentence accent and are generally placed at the tail-end of the sentence. Changing the word order (for instance by left dislocation or topicalization) changes the information structure.

The following example (7) (constructed) illustrates the differences between the notions of topic, expected response type and given information.

(7) S: U kunt naar de cabaret voorstellingen door Herman Finkers and Commil Foo.
You can see the comedy shows by Herman Finkers and Commil Foo.
U: Herman Finkers, wanneer is die in de schouwburg?
Herman Finkers, when is he in the theater?

The left dislocation in the user utterance, indicates that the fronted *Herman Finkers* is in focus. And indeed, picking Herman Finkers marks a contrast with other names from the selection. On the other hand the question clearly is *about* Herman Finkers. So, the current *topic* is *Herman Finkers* too. Apparently, the topic does not

[6] The definition is based on (Gundel 1985).

always correspond to contextually given information; topic (aboutness) and focus (informativity) represent orthogonal dimensions.

The *when* question sets the expected response type to be a *date*. According to Van Kuppevelt, a date would be the expected next topic. Suppose the system answers with a date, say *13th of may*, would we say that the answer was *about* Herman Finkers or about the 13th? The answer to this question is a matter of taste. Date is an attribute of performance. Answering the question gives a value to the attribute, thereby increasing knowledge about the performance. Therefore we concluded that expected response (salient question) and topic (aboutness) express different dimensions too.

A third approach to these issues is the *centering* approach[7]. Our notion of topic roughly corresponds to the *backward looking center*.[8] Centering is often applied in anaphora resolution. A list is kept of potential antecedents, ordered by a preference order based on syntactic function. The top of the list gives what is called the *forward looking center*: the most likely antecedent for anaphora in the next sentence. Since most anaphora are related to the topic, the forward looking center in a way predicts the next topic. For English the preference order is $subj < obj < other\ phrases$, so subject is preferred over object, which is preferred over other phrases. Since Dutch is a relatively free word order language, such a preference order based on syntactic function is not immediately available.[9] In section 7 we suggest topic detection heuristics based on syntactic cues and the information structure of the context.

6 Implementation

The dialogue manager has four tasks: determining the right response actions to a user utterance, updating the context, managing the interaction with the database and generating the actual system responses.

item list

The dialogue manager expects information in the form of an *item list*. An *item list* is a list of *item-value* pairs headed by a tag indicating the *utterance type* (UT) and a list of items indicating the *expected response type* (ERT). The utterance types are explained in section 7. An item occurs in the ERT when the utterance indicates that a value for that item is wanted by the user.

$ItemList ::= [UT : ERT; IV_1, ..., IV_n]$
where

UT	::=	`whq` | `decl` | ... | `misc`
ERT	::=	_ | $Item, (Items)^*$
IV_i	::=	$Item_i = Value \quad (1 \leq i \leq n)$
		| $Item_i \neq Value$
		| $Item_i =$ _
		| $UnaryItem$

Item-value pairs may indicate what value an item has (=) , what value an item will not have ($\neq$) or that the item has not yet been given a value (_). In that case, the value should be provided by the context. There are unary items indicating confirmation, denial, greetings and thanks. These may be called control items, contrasting domain items. We assume that each item occurs at most once. Here are some examples.

(8) Wanneer zijn er musicals? (820)
When are musicals on?
`[ whq; time; genre = musical ]`

(9) ik wil niet naar de 'groene vogel' (2369)
I don't want to go to the 'groene vogel'
`[ decl; _; title =/= 'groene vogel']`

The other parameters used to determine the best response action, topic and phase, are detected by the dialogue manager and stored in the context. A topic shift often occurs when an item-value pair is not compatible with the current topic. A phase shift is often indicated by an empty ERT (section 7). In the future, the set of control items will be extended with other syntactic and lexical cues, for instance the subject type and verb type, conjuncts or the presence of a question mark.

The input to the dialogue manager, an item list, is a simplified version of the feature structure produced by the parser. In this conversion some information will get lost. One of the main reasons for this loss is the assumption that all relevant information can be converted into item-value pairs. It means that all information that falls outside of the domain, even when it is parseable, is neglected.

[7] See (Grosz et al. 1995) for an introduction.
[8] The center is sometimes called *focus of attention*.
[9] Dutch is like German in this respect. (Strube and Hahn 1996) have formulated a preference order for German, based on the semantic functions of constituents.

context history

We need a representation of context that respects the potential topic structure or task frame. We think of this data-structure as a tree. Information along each branch must be consistent. It must be possible to move back and forward between branches, since it is possible in our system to come back to previously discarded topics.

The context data-structure is implemented as a list of nodes. Each node has two components: a *view* (the active records in the database) and a partially filled-in frame. The list forms the dialogue history, ordered linearly in time. For each context-node, the information shared by the records in view models the information conveyed by the dialogue so-far, which again corresponds to a partially filled in frame. Adding information means narrowing the view. When somebody selects `genre=dance` , the view is restricted to all performance records that have `dance` in their genre fields.

This view mechanism only allows a monotonic increase of information. Backtracking to previous context nodes is possible, but all information that has been added in between is lost from the view. This is undesirable, since some of the lost information may not be logically related to the discarded topic. Therefore, after backtracking, the view is recomputed from the other component of a context node: the frame. (see also example 10)

Items in the frame are stored with a flag that indicates their status. An item is *wanted* when it occurred in the ERT or when the system needs it as an argument for a response action. An item is *known* when the value was mentioned in conversation or follows uniquely from other items. An item can also be *unknown*, *partially known* or *known-not* to be of a certain value.

An item list is said to *match* the current context, if (1) all domain items in the item list occur in the frame of the current topic, and (2) for all sets of values for those items, there is at least one record in the view with corresponding values. For instance, when updating the view with restriction `artist='Jules Deelder'` would make the view empty, 'Jules Deelder' does not *match*. In that case backtracking is needed to a previous node on the history list.

backtracking

Here is an example of a dialogue with complicated topic-structure [10]. It shows that information increase is not always monotonic. Users may jump back and forward between alternatives.

(10)	U: What's on at the 18th of march?	1
	S: On the 18th you can see 'Deelder Denkt' or 'Indonesian Tales'.	2
	U: At what time does Deelder start?	3
	S: The performance by Jules Deelder starts at 8 o'clock.	4
	U: How much is it?	5
	S: We give student discount. Do you have a student card?	6
	U: No	7
	S: The ticket for 'Deelder Denkt' without discount is f30,-.	8
	U: Oh, that's expensive.	9
	U: How much is the Indonesians?	10
	S: One ticket for 'Indonesian Tales' without discount is f30,-.	11
	S: Give me Deelder then after all.	12

After mentioning the date, 18th of march, there are two alternatives. The user can freely switch between them. In sentences 3 and 4, the user asks information about Jules Deelder. This limits the topic to the Deelder performance. Via the calculation of prices, discount becomes a subtopic of that. However, in utterance 10 the user switches to the other alternative. Backtracking is needed because the phrase 'Indonesians' does not match with the discount, nor with the Deelder context. It does fit the '18th of march' context, producing a separate frame. In this new frame, the discount information seems to be lost. However, discount not only is a subtopic of a performance (via price) but also of the control topic *user*. Because of structure sharing between frames, the information is not lost. However, the *time* information does get lost in backtracking. This is as it should be because the 8 o'clock time is logically associated with the Deelder performance. So apparently there are two distinct realms: facts about performances and user-information.

generation

We use sentence templates for the generation of system utterances. It is well-known that automatic sentence generation produces a more natural flow of dialogue when topic-focus issues are

[10] The example is constructed, but based on utterances (003-058).

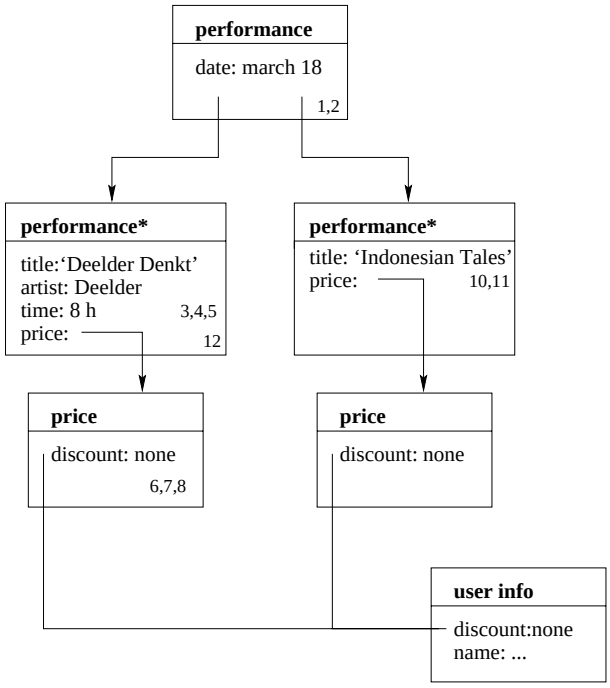

Figure 3: Topic Structure of (10)

taken into account (e.g. (Deemter et al. 1994) for spoken monologues). The templates are stored in a table that can be indexed by an expression that is combination of template type (`dec, whq, ynq`), control items, (`yes, thanks, bye`), items that are to be presented as *given* and items that are to be presented as *new*. Templates are constructed in such a way, that given items appear up front. *New* items may appear topicalized, or at the end of the sentence. The *new* items will be the results of the database search for items in ERT. The given items will normally be associated with the topic.

We plan to experiment with different response formats. For instance, we could choose to always use demonstratives or pronouns to refer to given items. We could choose, to always respond to a wh-question with its ERT only. The hypothesis is, that this would reduce the complexity of the dialogues. The most complex dialogues in our corpus, featuring nested sub-dialogues about discount, could have been avoided by a more clever use of sentence templates in the Wizard of Oz experiments.

7 Determining Topic, UT and ERT

This section deals with the following question: how can we automatically determine the *topic*, the *utterance type* and the *expected response type*, given the information in the feature structure produced by the parser, the previous utterance and the previous topic?

utterance type

Utterance type is the grammatical or lexical type of utterance. It can be detected by the parser using surface information. Utterance type is related to the communicative function, but not equal. For instance, in example (11) below a declarative sentence is used to request information about a performance. In fact, the majority of the declarative user utterances in our corpus have this function.

The set of utterance types in table 1 is developed by Andernach (Andernach 1996). Andernach uses so-called *cues* to represent surface information for automatic classification of utterances. The corpus has been manually annotated with these cues. Using an unsupervised learning algorithm, a partition of classes is found. Each class will correspond to a communicative function. Using supervised learning declarative descriptions of each class can be derived. These descriptions can be used to guide the response behavior of the dialogue manager. Utterance type is an important cue, as are the presence of a question mark and the semantic type of subject and verb. Utterance type can be detected using principles of Dutch word order. In a normal declarative sentence the first position is the subject position. Wh-questions also have a declarative word-order, but have a wh-word in first position. Yes/no questions have an inverted subject-verb order, as do topicalization constructions[11]. All cues can be detected using the parser.

expected response type

The ERT specifies what the type of response should be, according the form of the utterance. For instance, a *when* question expects a *time* to be supplied in the next response. The rules to determine topic and ERT are organized using utterance type. Example (11) is a declarative ut-

[11] See the Dutch grammar *ANS* for an analysis of Dutch word order and utterance types.(Geerts et al. 1984)

`whq`	fin verb on 2nd , wh-word on 1st
`decl`	fin verb on 2nd , no wh-word
`ynq`	fin verb on 1st, subject on 2nd
`imp`	fin verb on 1st, no subject
`prep`	prepositional phrases
`nom`	nouns, nounphrases, proper names
`adj`	adjectives, adverbs or numbers
`thank`	thanks
`greet`	greetings
`conf`	confirmation, negation (yes, no)
`excl`	interjection, emotives, exclamations
`misc`	miscellaneous

Table 1: Utterance types

terance, but it does have an ERT. It is the communicative function that is conventionally associated with the declarative *ik wil* construction that determines the ERT. Similarly, a yes/no question may expect, apart from a straight yes or no, additional information. (12) So all types of utterance may specify an ERT.

(11)Ik wil graag naar de voorstelling van Youp van 't Hek (446)
I'd like to go to the show by Youp van 't Hek
ERT : information, e.g. `date`

(12)Speelt Toneelgroep Amsterdam ook bij jullie? (141)
Does Toneelgroep Amsterdam play at your place?
ERT : `yes` + information, e.g. `date`

An empty ERT indicates that the syntactic form of the sentence does not normally demand a response of a specific type. This is the normal situation during the reservation phase, when the system has the initiative. In the other case an empty ERT often means a *phase-shift*. (see below)

topic

There is a number of sources in the syntactic form of a user-utterance that indicate the current topic or introduce a new topic. We have no proper algorithm for detecting the topic. We do have a number of observations, that make it plausible that such a detection algorithm is feasible.

First, there are general observations regarding the way topics are referred to. With respect to topic structure, user utterances can be divided in four groups, ordered by the relative complexity of the expression used to refer to the topic. Compare the results of (Rats 1996, ch5).

1. topic introduction: a new topic is introduced using definite descriptions, proper names, (fragments of) titles or complex referring phrases. Performances may be selected using date.
2. topic shift: a different topic is introduced. This happens when items from the utterance do not *match* the frame associated with the current topic.
3. topic narrowing: a subtopic of the previous topic is introduced. Uses the same mechanism as (2.)
4. topic continuation: the topic remains the same. The topic is referred to using shortened descriptions, demonstrative pronouns, personal pronouns, *ie* (it, he) or the topic is simply left out.

Turning these observation around, we obtain a powerful heuristics:

- when items are described using definite descriptions, proper names (fragments of) titles or other complex referring phrases, such items are likely to be the new topic
- when items are described using shortened descriptions, demonstrative pronouns, personal pronouns, or simply left out, the old topic is likely to be the new topic.

Second, we have collected a number of detailed *heuristics* from corpus observation. By way of example, the heuristics for declarative utterances and for wh-questions are discussed.

Our corpus contains 2414 utterances, of which 978 are user utterances.[12] Of a total of 130 declarative utterances, 79 contained the word *wil* (want) and 111 contained the word *ik* (I). Apparently most declarative sentences are of the form *ik wil X* (I want X). Often X refers to the new topic. The interjections *graag* (like to), *toch* (after all) and *nog* (still) occurred a lot. The polite form *ik zou (graag) X willen* (I would like to (have) X) also occurred. (8 times) A small part of declaratives consists of answers to system-questions. Most of these are of the form *ik heb* Y (I have Y) as an answer to a question about discount.

There are basically two classes, depending on the type of X: utterances requesting information about a performance and utterances requesting tickets or a reservation for a performance. When

[12]Opening and closing phrases by the system are all tagged separately. This explains the odd proportion.

X is related to a performance title, an artist, group or date, we assume that the user wants information. The topic will normally be a performance. An X related to tickets, reservation or price, indicates that the user wants a reservation, the topic being a reservation action.

A particular performance or set of alternative performances can be introduced as topic, using the items *genre*, *artist/group*, *title* or *date*. When the user selected a performance by one or two of these items, the system ought to respond with the other items from the triple (*group/artist* | *title* | *date*). So when the topic is a performance, the ERT is the list of those items from the frame, that can be inferred. When all of these are given already, the system still has the possibility of listing the text of a review or newspaper clippings.

There are 278 Wh-questions, which is more than a quarter of all user utterances. Wh-questions often ask for values of attributes of the current topic. The topic is likely to remain the same. Some question words directly determine the expected response type. *Wie* (who) expects a person. But others, like *welke* (which) and *wat* (what) need to be combined with other phrases. *Welke NP* expects an answer about or of type NP (example (13)). *Wat* is more difficult. Usually it combines with the first subcategorized daughter of the verb. For instance, in example (14) the expected answer is a price, because price normally is the object of *cost*.

(13)Welke opera's worden gespeeld? (1349)
Which operas are performed?
topic: set of opera's
ERT: opera

(14)Wat kost een kaartje voor die opera? (265)
What does a ticket cost for that opera?
topic: 'ticket for opera'
ERT: price

The exact syntactic category does not matter for determination of the topic and ERT. When the wh-phrase is the subject, the topic is often given by the object of the verb. When the wh-phrase is direct object or adjunct, the topic is given by the subject of the verb. In both cases it is the first nounphrase that comes after the wh-phrase.

8 Summary

This paper explains some of the choices that have been made in the design of the SCHISMA dialogue manager. We cannot yet conclude that these choices are a success; the work is still very much in progress.

The design can be characterized as an *utterance-based* dialogue manager: the syntactic structure and content of the user utterance, together with the structure and content of the context, determine an appropriate response action. The parameters that determine the state of the dialogue and therefore the behavior of the dialogue manager are *phase*, *utterance-type*, *expected response type* and a detailed data-structure that keeps track of the *context*.

The design of the grammar should have the parser produce at least and at most one parse. The art is to limit structural and lexical ambiguity, using syntactic and semantic constraints. Domain information can be applied in the grammar using type-hierarchies and a typed unification grammar.

The context data-structure is organized around the notion of *topic*: what the dialogue is about. With a topic several information items are naturally associated. This results in a frame structure, that functions as the design of the database. Items are slots in a frame. values are filled in as the dialogue progresses. Consistency is checked using a database view.

Organizing the context around topics in this way has a number of advantages: context dependent utterances like utterances containing anaphora, answers to questions or ellipsis, can be more easily understood. Backtracking and correcting previous choices becomes possible, without discarding all information that has been uttered in the meantime. It remains possible to come back to previously discarded topics. Response templates are indexed using combinations of given and new items.

Our use of the notion of topic differs from other well known accounts. First, it differs from a question-based approach (Kuppevelt 1995). Questions do have influence on the topic, but are better dealt with using a separate notion: *expected response type*. Second, it differs from the Prague School approach. Topics are 'old' or given, but not always. Finally, it differs from the centering approach. We do not use a priority list based on the syntactic function of constituents. Rather we use the predefined frame structure as-

sociated with the task and a number of heuristics derived from our corpus to determine the parameters topic, expected response type and utterance type.

References

Allen, J., Kautz, H., Pelavin, R., and Tenenberg, J. (1991). *Reasoning about plans.* USA: Morgan Kaufmann Publishers, Inc. IN 516.35 r023.

Andernach, T. (1996). A machine learning approach to the classification of dialogue utterances. In *Proceedings of New Methods in Natural Language processing-2.* Bilkent, Turkey.

Androutsopoulos, G. D. R., and Thanisch, P. (1994). Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering.*

Aust, H., and Oerder, M. (1994). Database query generation from spoken sentences. In *IVTTA'94.* Philips, Aachen. Kyoto, Japan.

Aust, H., and Oerder, M. (1995). Dialogue control in an automatic inquiry system. In Andernach, et al, editors, *Corpus-Based Approaches to Dialogue Modelling*, TWLT9. University of Twente.

Bos, R. (1995). Using finite state automata in dialogue modelling. Master's thesis, Dept of Computer Science, University of Twente.

Deemter, K. v., Landsbergen, J., Leermakers, R., and Odijk, J. (1994). Generation of spoken monologues by means of templates. In Boves, L., and Nijholt, A., editors, *Speech and Language Engineering*, TWLT8.

Geerts, G., Haeseryn, W., de Rooij, J., and van den Toorn, M., editors (1984). *Algemene Nederlandse Spraakkunst.* Wolters-Noordhoff, Groningen.

Grosz, B. J. (1977). The representation and use of focus in dialogue understanding. Technical report, SRI International.

Grosz, B. J., Joshi, A., and Weinstein, S. (1995). Centering: A framework for modeling the local; coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Grosz, B. J., and Sidner, C. L. (1986). Attentions, intentions and the structure of discourse. *Computational Linguistics*, 12:175–204.

Gundel, J. (1985). Shared knowledge and topicality. *Journal of Pragmatics*, 9:83–107.

Hajicova, E., Sgall, P., and Skoumalova, H. (1995). An automatic procedure for topic-focus identification. *Computational Linguistics*, 21(1):81–94.

Hoeven, G. F. v. d., et al. (1995). Schisma a natural language accessible theatre information and booking system. In *Proceedings of the First International Workshop on Applications of Natural Language to Data Bases.* Versailles, France.

Kuppevelt, J. v. (1995). Discourse structure, topicality and questioning. *Journal of Linguistics*, 31:109–149.

McConnel, S. (1995). *PC-PATR Reference Manual.* Summer Institute for Linguistics, Dallas, Texas. alpha test version, october 30 edition.

Moll, M. (1995). Head-corner parsing using typed feature structures. Master's thesis, University of Twente, Enschede.

Rats, M. (1996). *Topic Management in Information Dialogues.* PhD thesis, Katholieke Universiteit Brabant, Tilburg.

Shieber, S. M. (1986). *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes 4. Stanford, CA: CSLI.

Smith, R., Hipp, D., and Biermann, A. (1995). An architecture for voice dialog systems based on prolog-style theorem proving. *Computational Linguistics*, 21(3):281–320.

Sowa, J. (1984). *Conceptual Structures.* The Systems Programming Series. New York: Addisson-Wesley Publishing Company.

Steetskamp, R. (1996). Semantics in the SCHISMA domain. Master's thesis, University of Twente, Enschede.

Strube, M., and Hahn, U. (1996). Functional centering. In *ACL '96.*

Veldhuijzen van Zanten, G. (1996). Pragmatic interpretation and dialogue management in spoken-language systems. In *TWLT11.*