Topics in SCHISMA Dialogues

Joris Hulstijn, René Steetskamp, Hugo ter Doest, Stan van de Burgt and Anton Nijholt
{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 theatre perfomances and is which able to make reservations for these performances. In this paper we discuss several techniques for topic management that we developped and tested this year.

1 INTRODUCTION

In a (spoken) dialogue system one of the main tasks is answering questions that the user states, either explicitly or implicitely. As the question is often based on, or referring to, earlier utterances of either interlocutor, analysing 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 developpe: this system provides the user with information on theatre 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 uninterupted 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.

As we noted above, answering a question involves more information than can be found in the last utterance alone. In fact, we argue that the response is mainly determined by the following parameters: *(information) context*, *topic*, *utterance type* (UT), *expected response type* (ERT) and *dialogue phase*.

The *(information) context* incorporates both information extracted from the dialogue and domain knowledge. We choose to organise the context around the notion of *topic*, the entity the dialogue is currently *about* (see section 4).

Other parameters that guide the behaviour of the dialogue manager are *utterance type* (UT) and *expected response type* (ERT). The utterance type is roughly the grammatical type of sentence. (e.g. declarative, nominal, ...). The expected response type indicates what the user is expects the system response to be about. For instance, a *when* question from the user will prompt the system to respond with a notion of time or date.

Like topic, these parameters can be detected using syntactic cues from the user utterance, together with domain knowledge and the previous topic. This is discussed more thoroughly in section 6.

We distinguish several *phases* of the dialogue. Apart from the obvious *opening-* and *closing* phases, we distinguish a *information*, a *selection*, a *reservation* and a *confirmation* phase. During the first two phases, the information and selection phases, the user has the initiative.

For example, first the user requests information about performances, which is subsequently provided for by the system. During the reservation phase, the system takes more initiative, ask-

ing the user for information like name and adress in order to be able 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. Note that there is no obligatory sequential order, just a logical one. So, 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.* Now if the information provided by that 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 behaviour 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 theatre 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 5. In section 6 we report on some of our ongoing corpus-based research into ways of determining utterance type, topic and expected response type. Section 7 raps up with conclusions.

2 ARCHITECTURE

The SCHISMA system basically consists of three modules: a *morphological analyzer*, a *parser* and a *dialogue manager* (figure 1).

The user produces an utterance, which is scanned by the *morphological analyzer*, *MAF*, and transformed by the *parser* into a meaning representation in the form of an *item-list*. An *item list* is a list of *information items*[2], combined with

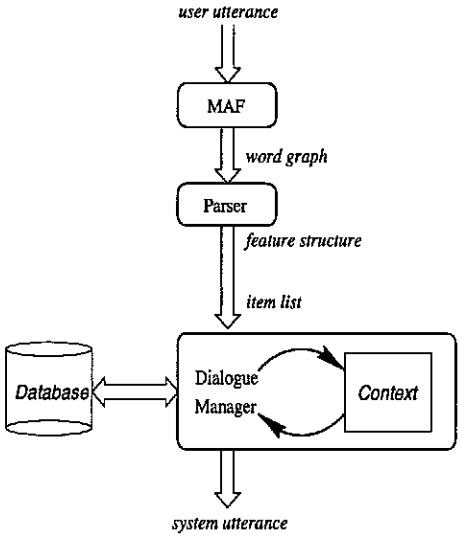

Figure 1: SCHISMA architecture

their values and flags indicating the status of the information (see section 5). The parser uses 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 one parse. In order to achieve this the grammar is based on four principles, which are explained in section 3.

The *dialogue manager* determines the appropriate response action based on the meaning representation of the user-utterance and on the context. The dialogue manager also updates the context with the information from the user utterance. 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'.

Item-lists are used to represent the information that the parser has been able to extract from the user-utterance that is relevant for the dialogue manager. Information that is not recognized by the lexicon or that cannot be parsed is neglected. Think of an item-list as a simplified version of the feature structure produced by the parser.

3 GRAMMAR

In developing a grammar and parser for the SCHISMA system, performance (contrary to competence) is the central issue. In our approach linguistic knowledge and theory is applied where necessary: we consider linguistics as a means for (domain-specific) meaning extraction rather than

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

[2] The term *information item* or shortly *item* is used for both objects and features or attributes that are important in the domain. Typical items are `time`, `artist` or `price`.

a goal in itself. In literature this type of language modeling is sometimes referred to as *semantic grammar*[3].

Our parsing system should process any input, i.e. tackle the no-answer problem[4], and extract as much domain-relevant information as can be recovered without the dialogue context. So, the parser should map its input onto preferably *one* expression, representing the context-independent part of the meaning that the user wants to communicate. See for instance the feature structure produced by an adjunct like example (1). The `schisma` part will be turned into an item-list. The context will reveal the function of the adjunct, for instance as the continuation of a previous question.[5].

(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 may seem rather trivial. However it has some implications that are less trivial and have clear consequences for the system as a whole. First, we need a dialogue manager that interprets the context-independent expression in context of the dialogue (see section 4). Second, if no alternative parses are allowed, introduction of lexical and structural ambiguity is allowed only if it can be accounted for at a later stage in the parsing process. Third, the parser should indicate exactly where contextual parameters still need to be filled in (for instance for anaphoric expressions or ellipsis).

We have translated the above findings into the following principles for the SCHISMA grammar development process.

P1 Do not introduce more (syntactic and semantic) alternatives in the lexicon than can be accounted for during parsing.

P2 The grammar should be as structural unambiguous as possible. Structural ambiguity is allowed only if, at a later stage during parsing, out of the introduced alternatives a selection can be made (using syntactic and/or semantic constraints). Note the correspondence to **P1**.

P3 Syntactic constraints guide the building of the meaning representation (semantic disambiguation).

P4 Semantic constraints guide syntactic processing (syntactic disambiguation).

In the course of writing the SCHISMA grammar the following two questions kept coming up.

Q1 What *structure* of utterances is important for determining the meaning of the input (and thus, what structure is unimportant)?

Q2 How can semantic knowledge about the domain be used to guide syntactic processing?

With respect to Q1, the best way to find out is to experiment with different set-ups. For these experiments in writing a SCHISMA grammar, we use a left-corner parser for PATR-II, 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.

As for Q2, we are planning corpus-based research into subcategorization patterns and corresponding semantic roles in our domain. We realize that the grammar not only combines words, but also the concepts behind it (Sowa 1984). 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

[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) (see principles P3, P4 below).

[4]When the grammar is too strict, the grammar may fail to parse, producing a *no answer*.

[5]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.

on their antecedent and elliptical expressions depend on the previously uttered phrases. See examples (2), (3) and (4) respectively.

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

(3) 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"

(4) U: En Othello?
and Othello?

We believe that the *information structure* of the context is just as important as its *information content* in resolving such dependencies (Chafe 1976). The information structure of the dialogue context is modelled 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[6].

Some items function as *subtopic*. Subtopics have a frame of their own. So, we get a hierarchical frame structure. The current frame structure is isomorphic to the design of our database.

So, the frame-structure that specifies the relations between potential topics, subtopics and attributes is our way of incorporating domain-knowledge in the system. However, a frame-structure cannot express all that needs to be known. For instance, the fact that a reservation presupposes a performance to have been selected, cannot be represented in the frame.

There are three other points in the architecture where domain-knowledge plays a role: in the grammar during unification (see principle P4), in the procedure that determines whether an item-list *matches* a frame (section 5) and in the procedure that specifies the actions needed to make a reservation. At the moment we have no formal way of specifying this kind of procedural knowledge. One way, would be to have both user- and system *plans* (Allen et al. 1991), to guide the system's behaviour and help understand the user's behaviour. Some of the intuitions behind plans are coded in our dialogue phases.

[6]Topic-focus related ideas have been used widely in dialogue systems. Recently, (Veldhuijzen van Zanten 1996; Smith et al. 1995; Deemter et al. 1994; Rats 1995). See section 4.1 for comparisons.

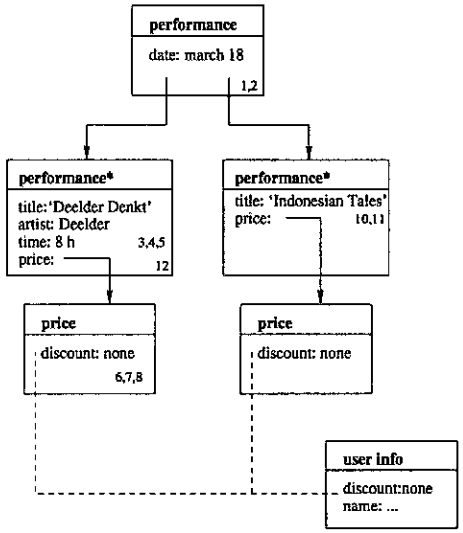

Figure 2: Fragment of Frame Structure

4.1 Other Approaches to Topic

The notion of *topic* is problematic in the literature. Different 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[7] (Rats 1996).

> 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). Kuppevelt uses a notion of topic based on *aboutness*. 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 our information dialogues, or it may be there implicitly. Topic is defined as the (type of) entity that would provide an answer to that question.

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

We found that the question-related notion of topic, although theoretically appealing, did not provide enough detail for successful automatic application to the corpus. Moreover, wh-questions comprise over a quarter of the corpus, so they are important enough to introduce a separate notion of topic to account for them. This is why we have *expected response type* among the parameters that determine the dialogue manager's behaviour (section 6).

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. Example (5) (constructed) illustrates the differences.

(5) 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 theatre?

The left dislocation in the user utterance, indicates that the fronted *Herman Finkers* is in focus. So, that fits nicely: 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. So, topic (aboutness) and focus (informativity) represent separate dimensions. Now, the *when* question sets the expected response type to be a *date*. According to Van Kuppevelt, a date would be the expected next topic. So, suppose the system answers with a date, say *13th of may*. Would we say that the answer was *about* Herman Finkers or about the date? To solve this, we concluded that expected response and aboutness express different dimensions too. The topic of the answer remains em Herman Finkers.

A third approach to these issues is the *centering* approach[8]. Our notion of topic roughly corresponds to the *backward looking center*[9]. Centering is most often used for anaphora resolution. A list is kept of potential antecedents, ordered by a preference order based on syntactic function. The list gives what is called the *forward looking center*. 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 subjects are preferred over objects or other phrases. Since Dutch is a relatively free-word order language, a preference order is not immediately available[10]. In section 6 we suggest a preferred next topic based on the syntactic structure of the utterance.

Two questions remain: how to determine the topic, given user utterance and previous topic and how to implement a context datastructure that supports topic theory? The first question is dealt with in section 6. The implementation of the dialogue manger is explained in the following section.

5 Implementation

The dialogue manager has to keep track of the parameters mentioned in the introduction. The most important of those parameters is the context. So how is the context updated with information from the user-utterance?

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*[11] (ERT).

$$item\_list ::= [\mathtt{UT} : \mathtt{ERT}; \mathtt{IV}_1, ..., \mathtt{IV}_n]$$

where

$$\begin{aligned} \mathtt{UT} &::= \mathtt{DEC} \mid ... \mid \mathtt{MIS} \\ \mathtt{ERT} &::= \_ \mid Item, (Items)^* \\ \mathtt{IV}_i &::= Item_i = Value \\ &\quad \mid Item_i = \_ \\ &\quad \mid Item_i != Value \quad (1 \leq i \leq n) \end{aligned}$$

Item-value pairs may indicate what value an item has (=) , what value an item will not have (!=) or that the item has been given no value (_). In that case, the value should be provided by the context. There are 'unary' items indicating confirmation, denial, greetings and thanks. We assume that each item occurs at most once.

[8] See (Grosz et al. 1995) for an introduction.
[9] The center is sometimes called *focus of attention*.
[10] 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.
[11] For a list of utterance types, see table 1.

Item names are taken to be equal to the names of the slots in the database. Values are strings or numbers[12]. Here are some examples.

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

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

In general there are two kinds of items: control items, like utterance type or `thanks` and domain-items like `title` or `date`. Control-items merely convey the form of utterance, domain items convey its content. There are no items indicating the expected topic or the phase of the dialogue. These are stored and determined within the dialogue manager. 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 6). In the future, the set of control items may be extended with other cues, for instance the subject type and verb type or the presence of a question mark.

Note, that the input to the dialogue manager actually is a feature structure produced by the parser. To get an item-list, the feature structure needs to be converted first. Although this conversion usally is straightforward, some information may get lost. One of the main reasons for this loss is the simplifying assumption that all relevant information can be converted into item-value pairs.

context datastructure

We need a representation of the context that respects the topic structure. Some of the characteristics of our topic structure are that it loosely resembles a tree, that the information along each branch must be consistent and that it must be possible to come back to to previously discarded topics. The topic-structure described here corresponds to a fragment of the frame-structure introduced in section 4. It is, as it were, a filled-in frame-structure.

The *context* datastructure is a list of which the elements have two components: a *view* (the active records in the database) and a set of item-value pairs. The information common to the records in view corresponds to the information that has been conveyed in the dialogue. 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 (figure 3). The view is stored in a priority-list of active records from the database. All records in view are moved to the head of the list. Records that are no longer in view remain in the list outside the view. In this way, the view can be stored as an integer value: the number of records in view counted from the head of the list.

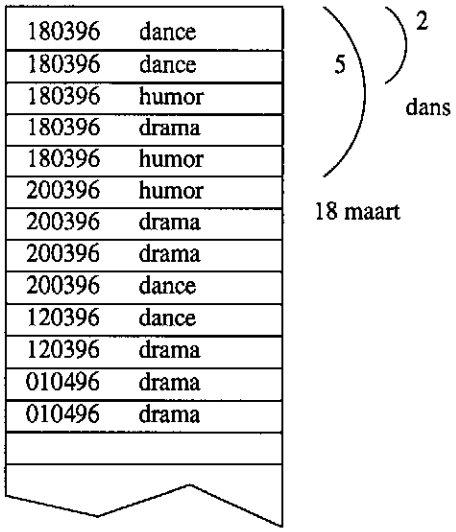

Figure 3: Database View

Note that the view mechanism only allows a monotonic increase of information. Backtracking 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 needs to be computed from scrap, using the list of item-value pairs (see also example 8).

Item-value pairs are stored with a flag that indicates its status. An item is *wanted* when it occurred in the ERT or when the system needs it in order to make a reservation (e.g. user address). An item is *given* when the value is present in the item-list. An item is *mentioned* when the item has come up in conversation but the value has not been supplied in the utterance.

A list of item-value pairs is said to *match* the current context, if (1) all *mentioned* items occur in the frame of the current topic, and (2) for all *given* items, there is at least one record in the view with corresponding values. When the view turns out to be empty, we say the item-list does

[12]For times and dates we use the representation formalism developed by (Aust and Oerder 1994).

not match[13].

Suppose that after updating the view with restriction `artist='Jules Deelder'`, the view has become empty, that means that 'Jules Deelder' did not *match* in the last context. In that case we would need to backtrack to a previous node on the context list, where 'Jules Deelder' would in fact match.

backtracking

Here is an example of a dialogue with complicated topic-structure [14].

(8)	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 are 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 open. 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. Discount is a subtopic of that. However, in utterance 10 the user switches to the other alternative. This will happen, 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 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*. Since this kind of user information is available in all contexts, 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 associated with the Deelder

[13]The matching procedure would be the place to extend the system with a domain inference module.

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

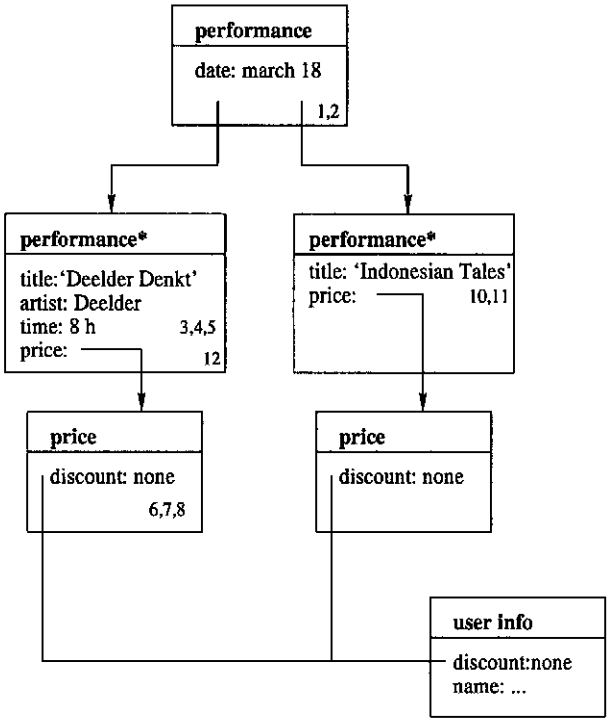

Figure 4: Topic Structure of (8)

performance. So apparantly 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 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 refer to the topic.

We plan to experiment with different 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[15].

6 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* (ERT), 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 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 (9) below a declarative sentence is used to request information about a performance. In fact, the majority of the declarative user utterances in our corpus are of this type.

The set of utterance types in table 1 is developed by (Andernach 1996). Andernach uses so-called *cue-patterns* representing surface information for automatic classification of utterances. The utterance type is an important cue, as are the presence of a question mark and the semantic type of subject and verb. The corpus has been manually annotated with these cues. Utterance type can be detected using principles of Dutch word order. In a normal declarative sentence the 1^{st} position is the subject position. Wh-questions have a declarative word-order, but have a wh-word in 1st position(SVO). Yes/no questions have SOV as do topicalization constructions[16].

expected response type

The ERT specifies what the form of the user utterance suggests the type fo the response should be. 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 (9) is a declarative utterance, which does have an ERT. It is the communicative function that is conventionally associated with the declarative *ik wil*-construction that

[15]The most complex dialogues in our corpus, featuring nested sub-dialogues, could have been avoided by a more clever use of sentence templates.

[16]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
DEC	fin verb on 2nd , no wh-word
YNQ	fin verb on 1st, subject on 2nd
IMP	fin verb on 1st
PRE	prepositional phrases
NOM	nouns, nounphrases, proper names
ADJ	adjectives, adverbs or numbers
THA	thanks
GRE	greetings
CON	confirmation/negation (yes, no)
XCL	interjection, emotives, exclamations
MIS	miscellaneous

Table 1: Utterance types

determines the ERT. Similarly, a yes/no question may expect, apart from a straight yes or no, additional information. (10) Therefore all utterances can specify an ERT.

(9) 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 : `date`

(10) Speelt Toneelgroep Amsterdam ook bij jullie? (141)
Does Toneelgroep Amsterdam play at your place?
ERT : `yes + 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.

Secondly, we have collected a number of detailed *templates*. The templates look like grammar rules. Each template represents a class of utterances of the same form. For each template the expected new topic and the expected response type are indicated. By way of example, the templates for declarative utterances and forv wh-questions are presented in detail. Similar templates exist for the other utterance types.

declaratives

Our corpus contains 2414 utterances, of which 978 are user-utterances.[17] 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), where X then 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) PS willen* (I would like to (have) X) also occurred. (8 times) 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. A small class consists of answers to a system-question. (see figure 5)

However, it is difficult to judge if the user wants information or already wants to reserve tickets. This can often be decided on the basis of the type of X. Is it a performance title, an artist, group or date, we assume that the user wants information. Is it related to tickets, reservation or price, we assume the user wants a reservation.

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

The following notation is used: () indicates optional elements. | and / indicate alternatives. Words in capitals represent syntactic categories of the grammar.

wh-questions

There are 278 Wh-questions, which is more than a quarter of all client utterances. Some question words directly determine the expected response type. *Wie* expects a person. But others, like *welke* and *wat* need to be combined with other phrases. *Welke NP* expects an answer about or of type NP (example (11)). *Wat* is more difficult. Usually it combines with the expected object of the verb. For instance, in example (12) the expected answer is a price, because price normally is the object of *cost*.

(11) Welke opera's worden gespeeld? (1349)
Which operas are performed?
ERT: opera

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

The exact syntactic category does not matter for determination of the topic and ERT. If 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 PS that comes after the WHP.

WH1 WHP V PS1 (PS2) ?
topic: PS1 ¡ 'topic
ERT: type(WHP)

WH2 wat V PS1 (PS2) ?
topic: PS1 ¡ 'topic
ERT: type(1st subcat(V) $\neq$ PS1).

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

D1 Ik wil (graag) naar **genre** | **group/artist** | **title** | **date**
I want /like to go to ...
topic = **performance**
ERT = if not given **group/artist** | **title** | **date**
else _

D2 Ik wil / zou (graag) / **information** over **genre** | **group/artist** | **title** | **date** (willen) (hebben)
I want to/ like to have **information** *on ...*
topic = **performance**
ERT = if not given **group/artist** | **title** | **date**
else **review**

D3 Ik wil / zou (graag) (N) **tickets** (voor **group/artist** | **title** | **date**) (willen)(reserveren)
I want to/like to have/reserve (N) **tickets** *(for ...*
topic = **tickets** < **performance**
ERT = _

D4 Ik heb PS1
I have PS1
topic = PS1 ¡ 'topic
ERT = _

Figure 5: Declarative Sentence Formats

7 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 behaviour of the dialogue manager are *phase*, *utterance-type*, *expected response type* and a detailed datastructure that keeps track of the *context*.

The design of the grammar is based on for principles, that help to have the parser produce one parse only. The idea is to keep the number of alternative parses down by limiting alternative lexical entries, by disallowing mere structural ambiguity and by combining syntactic and semantic constraints from the domain. Domain information can be applied in the grammar using type-hierarchies and a typed unification grammar.

The context datastructure is organized around the notion of *topic*: what the dialogue is about at that point. 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 more easily be 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.

Our use of the notion of topic 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*. We differ from the Prague School (Hajicova et al. 1995) in that word-order is not used as the main cue to discern the topic. Rather we formulated a number of standard sentence formats that utterances in our corpus can follow. For each of these formats, the predicted effect on topic and response type was indicated. The results come out remarkably similar to the preference order used in centering theory (Grosz et al. 1995). Word order does make a difference in the selection of sentence templates that are used in sentence-generation. Templates are indexed using combinations of given and new items.

REFERENCES

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

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

Chafe, W. L. (1976). Givenness, contrastiveness, definiteness, subjects and topics, and point of view. In Li, C., editor, *Subject and Topic,* 20–55. Academic Press.

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

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, PO BOx 217, 7500 AE Enschede.

Rats, M. (1995). Referring to topics. In *Corpus-Based Approaches to Dialogue Modelling,* TWLT9.

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: Center for the Study of Language and Information.

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, The Netherlands.

Strube, M., and Hahn, U. (1996). Functional centering. In *ACL '96: Proceedings of the 34th Meeting of the Association for Computational Linguistics.*

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