Discourse Representation and Discourse Management for a Natural Language Dialogue System

Lars Ahrenberg, Arne Jönsson and Nils Dahlbäck
Department of Computer and Information Science
Linköping University
S - 581 83 Linköping

ABSTRACT

In this paper we discuss some requirements on natural language interfaces that are needed for supporting connected dialogue, and propose methods for meeting these requirements in the context of a system that has been developed at our department over the last couple of years. On the basis of data collected in Wizard-of-Oz simulations we argue that the dynamic discourse representation needed for a given application can be structured in terms of a tree of dialogue objects (moves, initiative-response units) and a score board, i.e. a list of parameters each of which keeps some information relevant to the system's interpretation and generation tasks. We also argue that the domain knowledge needed for supporting reference resolution can, and should be obtained from data of this kind.

1 Introduction

In this paper we discuss some important requirements on natural language interfaces (henceforth NLIs) that are needed for supporting connected dialogue, and propose methods for meeting these requirements in the context of a system (LINLIN - LInköping Natural Language INterface) that has been developed at our department over the last couple of years.[1] Naturally there are many other requirements that a NLI must meet - as regards linguistic coverage and overall design - but these will not be discussed here (see e.g. Rettig & Bates (1988) for an overview of relevant requirements).

In section 2 we give a short overview of the system, in section 3 we report some findings on referential and coherence relations in interface dialogues that are exploited in the system, in section 4 we describe how the system monitors and controls the dialogue, and, finally, in section 5, we summarize and briefly mention some ideas for customizing the system to an application.

2 The system

The LINLIN system is characterized by the following features, some of which were determined already at the start of the project and some of which have emerged in the course of its development:

- Linguistic knowledge and domain knowledge are integrated in the same knowledge-base, allowing for simultaneous development (Wroblewski & Rich, 1989) and interleaving of syntactic and semantic processes (Ahrenberg, 1989a).
- The system is modular in the sense that different kinds of knowledge can be manipulated separately by the system designer.
- An object-oriented knowledge representation language is used throughout making it

possible to use inheritance also in the expression of linguistic generalizations (Flickinger et al., 1985; Gazdar, 1987).

- Unification is used as the main operation for the derivation of syntactic and semantic descriptions of utterances, while chart parsing techniques are employed for parsing and surface generation (Shieber, 1988; Wirén, 1988). The linguistic knowledge may be compiled into PATR-II format (Shieber, 1984), extended with disjunctive and negative values as well as some notational conveniences (Andersson, 1989).
- Semantic interpretation is object-oriented (Hirst, 1987) and concerns the linking of linguistic objects (parts of utterances) to objects (instances, classes, properties) of the universe of discourse, of which the system may have independent knowledge. A pilot system demonstrating the approach to semantic interpretation is described in Ahrenberg (1989b).
- A sublanguage approach (Grishman & Kittredge, 1987) is assumed on all levels of description from vocabulary to discourse pragmatics. The system may be viewed as a shell which can be customized to different applications. Such customization is concerned with the different parts of the knowledge-base only, while the processing modules remain intact. Studies aimed at the recovery of specific properties of man-machine communication in natural language have been performed (Jönsson & Dahlbäck, 1988; Dahlbäck & Jönsson, 1989)

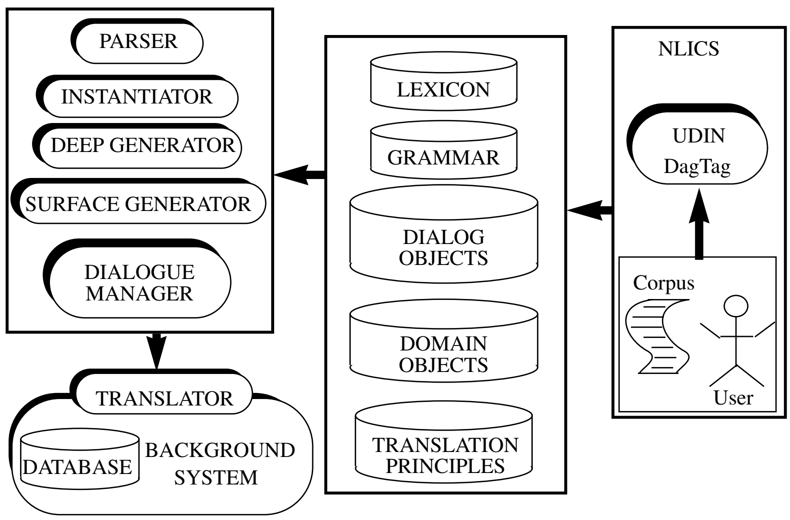

Figure 1. Overview of LINLIN

The system has four major parts (Figure 1):

- The background system with its interface and data model. This will not be elaborated more upon here.
- The processing modules: the parser, the instantiator, the deep- and surface generators, the dialogue manager and the translator. These are regarded as domain independent general processors which do not change between applications.
- The static knowledge bases used by the processing modules: they comprise general and domain-specific vocabulary (the lexicon), knowledge of syntactic constructions and their semantic impact (grammar), knowledge of general and domain-specific concepts (domain objects) and knowledge of the communicative events and structures possible in a dialogue (dialogue objects).

- Finally, we have the customization tools, which consist of an editor for the knowledge-base and a system for analysing a corpus of dialogues (Ahrenberg & Jönsson, 1989; Jönsson & Ahrenberg 1990).

At the top of the conceptual hierarchy a distinction is made between linguistic and non-linguistic objects. The latter concept has the concept of a dialogue-event as a specific subconcept. Thus, all knowledge is integrated in the same structure and expressed in the same way.

Linguistic objects are either syntactic or morphological. Word forms are special in being both syntactic and morphological, i.e. they are the terminals of syntactic structures and the roots of morphological structures. In the current implementation syntactic and morphological objects differ in how internal structure is expressed, as there are built-in functions that handle morphological analysis and synthesis directly in terms of the structural properties assigned to word forms and morphs, while structural information assigned to syntactic objects are first compiled into a form that better suits the parser.

Domain objects, dialogue objects and the dynamic knowledge structures of the system are presented in more detail below.

The system is controlled by the dialogue manager, which receives user input and controls message passing between other modules of the system (Figure 2). In section 4 the working of the dialogue manager is explained in more detail.

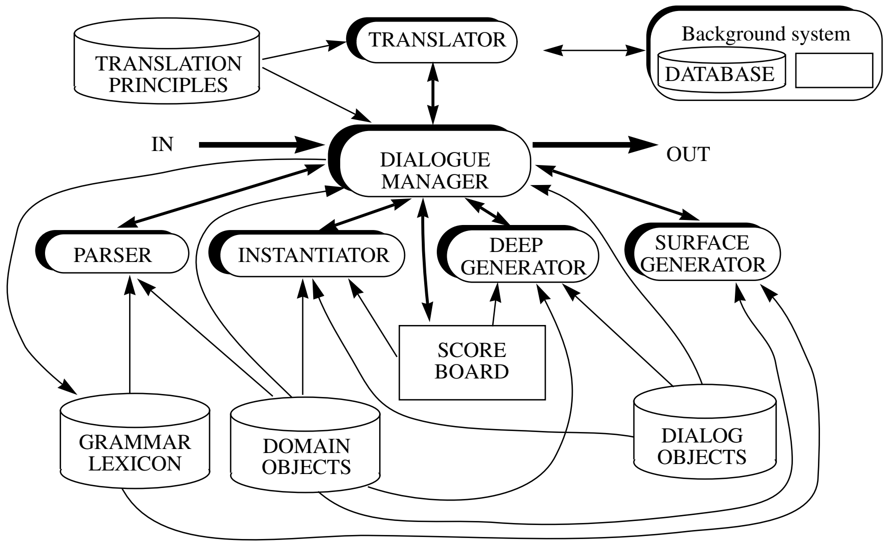

Figure 2. Control and information flow in LINLIN.

3 Discourse representation for connected dialogue

3.1 Coherence in dialogue

In this section we will discuss in some detail our work on two important features that make a dialogue connected: anaphoric, referential relations and coherence relations between moves made by the two alternating participants of the dialogue. We discuss these issues from the point of view of the static knowledge needed. As mentioned above a general assumption underlying our work is that a NLI must be customized to each specific application. One consequence of this is that our aim is not to develop any general, all-encompassing computational theory of discourse, but rather, for a given (type of) application, and a module of the system, to find the simplest possible model that will satisfy the requirements of that application. We will, however, occasionally make reference to such general theories and compare our findings with their predictions.

3.2 Domain objects and referential relations

One of the hardest problems of interpretation is to determine the objects and concepts referred to in an utterance. Here we limit ourselves to two important classes of referring expressions: anaphoric pronouns and definite descriptions. In the former case, it has for a long time been an accepted truth within AI and computational linguistics that simple algorithms working only on the immediate linguistic context will not suffice for the task of reference resolution. Often cited examples of this have been the usage of pronouns in the task-oriented dialogues collected by Grosz (1977), where in the extreme case the pronoun 'it' was used to refer to the pump being assembled, the last mention of which in the dialogue was some 20 minutes earlier. Some recent theoretical and empirical work has, however, made it possible to question the validity of these assumptions, at least for some types of discourse, including typical application domains for NLI's such as database retrieval and advisory systems. To take one example, Guindon (1988) analysed the use of pronouns in a number of Wizard-of-Oz dialogues between users and an advisory system to a statistical package.[2] She found that the structure of the dialogue in the case of typed input was simpler than in the case of spoken face-to-face dialogue. Since pronouns were not very common anyway, and since their antecedents could be found in the local linguistic context, Guindon argues that it is unnecessary to develop complicated mechanisms for pronoun resolution for these kind of applications. The use of anaphoric pronouns in our own corpus give support to this claim. Elsewhere we argue that the reason for the differences obtained in Grosz' and Guindon's dialogues is the different types of *common ground* (Clark & Marshall, 1981) available in the different cases (Dahlbäck, 1990). To put it simple; in the case of the task oriented dialogues, the pump is not to be found some 50 or 100 utterances away in the previous dialogue, it is just there in front of you, and the pronoun 'it' is not a case of anaphor but exophor. In the terms of Clark and Marshall, the common ground required for the successful use of such a referring expression is *physical co-presence*, not *linguistic co-presence*.

A definite NP is either coreferential with another NP, or is related to some previously mentioned discourse referent. The former case, by Hawkins (1979) called 'strict anaphoric' is often taken as the most common or prototypical case. But Fraurud (1989) found that in her corpus of newspaper articles only 2% of all definite NP's refer to entities introduced with an indefinite NP, and even if other types of previous explicit mention is included in the same category, it still only comprises 15% of the total sample. Other types of definite NP's are thus more frequent. Two of the most common of these are Hawkins' 'associative anaphoric' and 'larger situation' uses of definite NP's. In the former case, the previous mention of a referent is seen as triggering a number of possible associated referents (e.g. *a car: the engine*), in the latter the discourse situation or context as such triggers these associations (e.g. when discussing a lecture mentioning 'the teacher' or 'the blackboard').

In line with our sublanguage approach, we believe that the knowledge underlying the use of definite descriptions such as those above is to a large extent domain specific. Our method for obtaining this knowledge for the different domains is to use Wizard-of-Oz simulations, and then analyse the resulting dialogues. In a sense, we are performing a kind of protocol analysis, but of a different kind than the one used in the well-known work of Ericsson & Simon (1984) in that we search for the knowledge implied by the language used rather than the knowledge which is expressed explicitly. In work more similar in spirit to ours, Regoczei and Plantinga (1987) studied dialogues between experts and knowledge engineers as a means of uncovering the experts' underlying conceptual knowledge. Our work differs from theirs in two respects. First, their interest is in expert systems, our's is in NLI's. Second, they use one specified knowledge representation system (the conceptual graphs of Sowa, 1984), whereas we have decided not to use any specific knowledge representation scheme when analysing the dialogues. Instead we have worked with as few theoretical commitments as possible, in order not to enforce some specific theoretical straitjacket on the implementation of such a module. In most, if not all, cases, we can capture the necessary knowledge using three types of conceptual relations: subconcept/superconcept relations (IS-A), instance-concept relations and whole-part relations, which seem to be part of any minimal vocabulary for describing relations between different concepts.

We have made a preliminary analysis of the dialogues described in Jönsson and Dahlbäck (1988) and Dahlbäck and Jönsson (1989) from this point of view. Our aim has been to find the concepts used and the explicit or implicit relations between these. The general impression emerging from this analysis is that the conceptual structures are surprisingly simple. The conceptual structure is often rather ‘flat’, and a small set of conceptual links seem sufficient for describing the links between the concepts used, as these emerge e.g. in the use of definite expressions violating the prototypical given-new sequence. Furthermore, the conceptual relations are not always what you would find using an association test or a dictionary (cf. Fraurud, *op.cit.*). Even if we grant that there may exist some ‘standard’ conceptual relations, we sometimes find relations that are ‘non-standard’, though very natural in the specific domain in which they occur. To take one concrete example, in the context of obtaining information about charter holidays to Greece the concept ‘room’ has a whole-part relation to the concept ‘shower’, something which you cannot find in a standard dictionary definition, or even characterization of the concept ‘room’. The latter would instead have parts such as ‘floor’ and ‘ceiling’, concepts which are never mentioned in these dialogues. Even if we regard ‘room’ as polysemous between ‘room’ and ‘room-in-a-hotel’, the former concept never occurs as such in these dialogues, since it is part of the mutual knowledge that room in this context denotes ‘room-in-a-hotel’ and not ‘room’. Similarly, since it is part of the common understanding that Greek islands offer good facilities for summer holidays, but not winter holidays, users often, given a mentioned hotel, speak of 'the beach', but they do not speak of 'the pist(s)'. Using the work of Clark and Marshall (1981) mentioned above, we can say that the system and users share knowledge of the discourse domain, and share it as a part of their mutual knowledge. In this case, the use of potentially ambiguous expressions is possible. It is for this and other similar reasons that we believe that NLI's will need to be customized for the different specific application domains. A corollary of this is that for domains where such a mutual knowledge does not exist within the potential user community, a NLI might not be of much use. In our corpus we have one possible such case, an advisory system for the selection of wines to a planned dinner. The dialogue below illustrates the problem.[3]

EXAMPLE 1 U9> Which medium dry wines can I have with moose steak?
S10> Red wines are not described as dry, but in terms of sweetness and harshness.
U11>Certainly, they are!

Not all examples are as clear as this one, but the problem is not isolated the clear cut ones. In a sense, the problem is more pronounced with the more subtle cases, since misunderstandings risk being undetected for a longer part of the dialogue.

3.3 Dialogue objects and topical coherence

There are many models of dialogue, too numerous even to be mentioned here. Most of them have in common, though, that they recognize constituents of the dialogue of some sort. Models may differ in the number of such units they employ, in how these units are characterized and in the importance given to them in comparison with other factors such as speaker intentions or structure of the activity in which the dialogue is embedded, but they are usually present in some way. We refer to constituents of a dialogue by the name of *dialogue objects*.

Dialogue objects are divided into three main classes on the basis of structural complexity. There is one class corresponding to the size of a dialogue, another class corresponding to the size of a discourse segment (Grosz & Sidner, 1986; Allen, 1987: 398f), and a third class corresponding to the size of a single speech act, or dialogue move. Thus, a dialogue is structured in terms of discourse segments, and a discourse segment in terms of moves and embedded segments. Utterances are not analysed as dialogue objects, but as linguistic objects which function as vehicles of one or more moves.

These classes of dialogue objects are those that we have found necessary and sufficient for analysing and managing the dialogues of our corpus so far; there is no claim that they are applicable to all types of dialogue, and even less so, to any type of discourse. The simplicity of the schema is quite striking and becomes even more striking when one looks at the structure of the discourse segments.

While there is, in the AI-literature, fairly large agreement on the usefulness of the notion of discourse segment, there are no simple algorithms for detecting segment structure. Cue words help sometimes, but are usually absent; intonation is definitely absent when input is made via keyboard. Grosz & Sidner (1986) have argued that what characterizes a discourse segment is a definite purpose, called by them the discourse segment purpose, or DSP, so that the segment structure is more or less isomorphic to the DSP-structure. However, unless the discourse follows a pre-determined plot or plan, as is the case with task-oriented dialogues (Grosz, 1977; Litman & Allen, 1987), the DSP-structure is no more available than the segment structure at a given moment of the discourse. For this reason we favour a structural approach to discourse representation, which means that the segment structure is primary. This then has to be inferred from the structure and content of utterances.

In fact, one can argue that if the segment structure can be detected on this basis, then there is no need to refer to an independent DSP-structure at all. In the case of queries directed to a data-base there may be all sorts of plans that a user follows: he may be interested in obtaining information about some item that he intends to order, he may be comparing items for a survey or he may be demonstrating a feature of the system for a friend. Instead of bogging the system down with uncertain guesses at domain plans, we require of the system only that it has knowledge of small-range discourse plans which essentially contain an initiative (a move that opens the segment), and an appropriate, eliminatory response (a move that closes the segment). For this reason we refer to the discourse segments of this type of dialogue as *initiative-response units*, or *IR-units*, for short. The eliminatory response need not be adjacent to the initiative - the immediate response may instead start a clarification sub-dialogue or be an interruption of some sort, - but it must follow at some point. Yet the similarity of this scheme to the analysis of conversational structure in terms of adjacency pairs (Levinson, 1983: 303f) is obvious and interesting. A list of the most common IR-units used in the analysis of our corpus is found in table 1.

TABLE 1. Some important Initiative-Response units in a corpus of interface dialogues.

IR-unit	Type of Initiative	Types of Closing Response
Q/A	Q = Information Request	A = Answer
Q/AS	Q = Information Request	AS = Assertion that implies failure to answer
D/ACK	D = Directive	ACK = Acknowledgement (after execution)
AS/ACK	AS = Assertion	ACK = Acknowledgement (of information)
AS/AS	AS = Assertion (of interest, want)	AS = Assertion (forwarding interest)[4]

For this scheme to be possible, the system must merely be able to differentiate between these two basic classes of move, and it must, for each subclass of initiative have means to identify it as such, and a repertoire of eliminatory and immediate response moves that are applied under well-defined conditions.

An initiative is usually connected with a user intention of some sort, at least for a human observer. A query may be interpreted as a request for information, a command as a request for action and so on. The best response would accordingly be a move that supplies the requested information, or performs the requested action, but the best responses are not the only ones that are appropriate and eliminating. Assertions that the system is unable to find the information or forbidden to perform the action have the same effect.

We take IR-units to have many of the properties associated with discourse segments in the literature. They provide a structure constraining the search for referents of pronouns and simple definite descriptions, i.e. objects that have been accessed in the current segment are considered as indices for such indexicals before objects belonging to other segments, and they provide domains over which factors such as topic, modality and temporal perspective remain constant. Also, their simplicity provides transparency and regularity for the user and still a measure of flexibility of design.

Many structural approaches to dialogue modelling recognize complex units that consist of sequences of discourse segments, but do not comprise the whole dialogue. This is true also of some models intended for computational use, such as that of Wachtel (1986), where four levels are used, called conversation, dialogue, exchange (corresponding to the notion of IR-unit used here) and move, respectively. The feature characterizing the intermediate level in Wachtel's model is that of having a common topic, i.e. an object whose properties are discussed over a sequence of exchanges. However, the notion of topic is not as simple as that. In fact, a sequence of segments may hang together in a number of different ways; e.g. by being about one object for which different properties are at issue. But it may also be the other way around, so that the same property is topical, while different objects are talked about. As an illustration consider the dialogue of example 2 below. There, price is introduced as a topical property in utterance U2 and recovered in utterance S11 where the question of U2 receives an answer. Also in the following IR-units consisting of moves U12/S13 and U14/S15, respectively, price is the topical property although the set of items of interest is changing. Only in move U16 a new topical property is introduced, while the topical object is picked up from the previous IR-unit.

EXAMPLE 2 S1>Welcome to Vingfritid[3]
U2>Hi. How much is a holiday-trip to Greece?
S3>Where do you want to go?
U4>To which places do you organize trips?
S5>You may go to any of the islands Calymnos, Corfu, Kos, Crete, Patmos, Rhodos and Samos.
U6>We would like to go to Corfu.
S7>What class of hotel do you prefer?
U8>The highest.
S9>For how long do you want to stay?
U10>For two weeks.
S11>Please wait... Hilton Hotel***** costs SEK 5575 per person.
U12>How much is the cheapest class?
S13>Please wait... Royal Hotel*** costs SEK 2245 per person.
U14>Is there any trip which costs SEK 4000 per person?
S15>Please wait... Diwani Hotel**** costs SEK 4195 per person.
U16>How far is it to the beach from Diwani Hotel?

A more complex case, quite common in data-base querying, is when a set of objects are specified intensionally by the user as potentially interesting, say “small, cheap cars” (cf. example 3 below) and a number of segments following that specify this topic in different ways.

Thus, rather than using a single, vaguely characterized notion of topic, we find it better to distinguish a number of topical parameters. Three such parameters that we have used in cases such as those illustrated above are CurrentObject, CurrentAttribute and CurrentSet. Each of these parameters may remain constant over a pair of adjacent IR-units, but they may also change. What is interesting is that they do not change their values at the same points in the dialogue, as illustrated in the dialogue of example 1. As there is no reason to give any one of these parameters a specific priority in defining dialogue objects on the basis of “constant topic” and we want to keep the structure of dialogue objects strictly hierarchical, we register them simply as properties of IR-units. A question is then, of course, how we know what value to give to a topical parameter, when a new IR-unit is started. A preliminary heuristic, which seems to work well, is the following: If there is explicit information in the initiating move of what the value should be, then use that information. If there is no explicit information, assume the value of the previous IR-unit. As a case in point consider utterance U6 of example 1. In this move the user gives the value for an attribute of the CurrentSet which is important to the system and which is assumed to hold when the system answers questions such as U12 and U14 and which is kept until it is explicitly changed.

Another property relating to topic is the topical domain. We make a basic distinction between *task-related* initiatives, which are directed to the background system, and *discourse-related* initiatives, which seek clarification, in one way or other, of previous moves. The same distinction is made for the corresponding responses and for IR-units. Below we will often indicate a task-related move with the subscript T, and a discourse-related move with the subscript D. Other classifications relating to the same property are system-related initiatives (subscript S) that are concerned with information **about** the background system rather than from it, and parameter-related initiatives (subscript P) that are taken by the system when it needs a value for a parameter in order to perform some system-internal task. Utterances S3, S7 and S9 of example 1 can be analysed as parameter-related initiatives. The topical domain of a move is correlated with a definite knowledge-base, or part of a knowledge-base, of the system. Thus, a task-related move should result in (or from) a data-base query, while a discourse-related move involves the discourse representation and a parametric move involves an attribute of the CurrentSet.

Other properties of moves concern the identity of initiators and addressees and, in the case of questions and answers, the property or argument questioned. There are also contextual properties for moves and IR-units. For instance, a discourse-related IR-unit can only occur in the context of a previous move. Illustrations of dialogue object specifications are given in Figure 3 (a question) and in Figure 4 (an IR-unit).

4 Dialogue management and dynamic knowledge bases

In the course of a dialogue the system keeps a representation of the dialogue structure in the form of a tree of dialogue objects, which is updated for every new move. The reasons for maintaining a representation in the form of a tree that covers the whole dialogue are primarily the following: (i) trees are computationally simple structures; (ii) the dialogues that we have in our corpus can be analysed in this way, i.e. there seem to be two ways in which IR-units relate to one another: they are parallel but temporally ordered as in the case of a sequence of task-related units, or one IR-unit is subordinated to another one by seeking or giving information that complements the information given in the initiative or response of the dominating IR-unit; (iii) the open constituents of the dialogue, i.e. the right frontier of the tree (cf. Webber, 1990 and the stacks of open segments used in many models, e.g. by Grosz & Sidner, 1986) provides a structure that constrains the search for indices of indexical expressions; (iv) it is possible to refer to dialogue objects that are not on the right frontier by means of expressions such as *your previous answer*, and even to objects that are not salient as in *the car we talked about before*.

Class	(VALUE Q)
IR-Unit	(TYPE Q/A)
TopicalDomain	(VALUE <IR-Unit TopicalDomain>)
Initiator	(VALUE <IR-Unit Initiator>)
Addressee	(VALUE <IR-Unit Addressee>)
CurrentObject	(VALUE <IR-Unit CurrentObject>)
CurrentAttribute	(VALUE <IR-Unit CurrentAttribute>)
CurrentSet	(VALUE <IR-Unit CurrentSet>)
Q-Object	(VALUE (or <CurrentObject> <CurrentSet>))
Q-Aspect	(TYPE Path)
Q-Answer	(TYPE (or Thing Description))

Figure 3. Description of a question[5]

Class	(VALUE Q/A)
TopicalDomain	(VALUE T)
TopicalBase	(VALUE Database)
Initiator	(VALUE User)
Addressee	(VALUE System)
Initiative	(TYPE Q)
Response	(TYPE A)
CurrentObject	(TYPE Thing)
CurrentAttribute	(TYPE Attribute)
CurrentSet	(TYPE Description)

Figure 4. Description of a task-related Initiative-Response unit.

Another dynamic knowledge structure is the Scoreboard. It is a structure used by various modules of the system, in particular the Instantiator and the Deep Generator when they encounter problems with indexical expressions. Some of the attributes of the Scoreboard are identical to attributes that are found on moves and IR-units in the dialogue tree, but others are different, see figure 5. The value of an attribute on the Scoreboard is often a function that examines the dialogue tree in order to solve a referential problem.

The module responsible for building and updating these dynamic representations is the dialogue manager (DM). In fact, as explained in section 2, the responsibilities of the DM is to control the system as a whole, and for this purpose a third dynamic structure, the Action Plan Stack is used. An illustration of the structures that the DM manipulates is given in Figure 5.

One important feature of the DM is its distributed control. The actions of the action plan are distributed on the nodes of the dialogue tree that are still open. We can say that the nodes in the dialogue tree are responsible for their own correctness. Initially the node representing the dialogue as a whole creates an instance of an IR-unit whose type is unknown and integrate this IR-unit into the tree. The IR-unit in turn creates an instance of a user move, i.e. a move that waits for a user input to be parsed and instansiated. This user move is not integrated into the tree until the parser and instantiator are finished and it is then the DM that links it to a suitable node. For a normal task-related initiative this node is the IR-unit where it was instantiated, but in some cases the user does not follow this straightforward behaviour and so the DM must find another node to connect it to by traversing the right frontier of the tree.

The feature of distributed control means that if, say, a user move does not succeed in parsing the input, the user move node creates an instance of a clarification request IR-unit, i.e. a Q_D/A_D, and inserts this node to its right. This IR-unit consists of two parts, one part for prompting the user with a clarification request and another to interpret the user input.

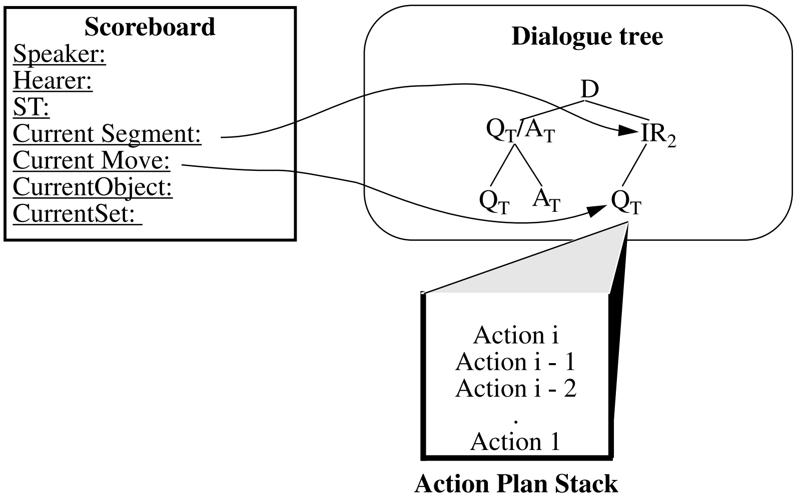

Figure 5. The internal structures used by the dialogue manager

In example 3 we have an illustration of another type of subordinated IR-unit.

EXAMPLE 3 U1> I would like a car which is between three and five years old
S2> What brand?
U3> Saab or BMW
S4> The following alternatives found: BMW 318/320, SAAB 99/90
U5> How many cars are there to choose from?
S6> The 1985 model of BMW318/320
The 1985 model of SAAB 90/99

The dialogue fragment consists of three IR-units where one IR-unit, S2-U3, is embedded in another, U1-S4. The resulting dialogue tree is shown in Figure 6. The clarification request S2 is created from an instance of an IR-unit for clarifications Q_P/A_P, which is connected to the IR-unit, AS_T/AS_T, who initiated the first user move-node, the one which interpreted U1. This is done when the node AS_T/AS_T finds that the given information is not enough for accessing the background system. The user answers the clarification request properly and the answer is returned from the Q_P/A_P-node to the node AS_T/AS_T which initiated the clarification request. Thereafter an answer is given. This is the left subtree of Figure 6.

.

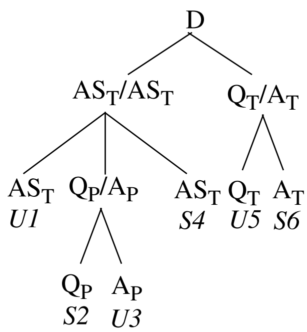

Figure 6. Dialogue tree for example 3.

After the initial segment, the user initiates a new question. This question starts a new segment, but with the same salient objects. This is achieved by copying the information from one segment to the other, as explained above.

The Action Plan Stack is responsible for the execution of the different actions. There is one stack for every node and execution proceeds by executing the stack top of the current node.

When a node is created, its process description is placed on the stack connected to that node. The process descriptions, or action plans as we call them, are quite simple. For instance the plan for a move which should generate an answer to the user consist of a sequence of three process calls: (deep-generate(<Answer>)) (surface-generate(<Answer>)) (up), where the first activates the Deep Generator, the second activates the Surface Generator and the last returns control to the mother node.

We can have such simple action plans because of the distributed control. To return to utterance U1 of example 2, which shows the common failure that the user move does not contain enough information for a successful data base access, the action plan is modified as follows. The user move is created from an instance of an IR-unit which originally had the action plan (create-move(<Initiative>)) (access(<TopicalBase>)) (create-move(<Response>)) (up) on its stack, where the arguments refer to the as yet unknown values of the named attributes of the IR-unit's description. When the utterance is interpreted and control is given back to the node that created the user move, the type has been identified as AS_T/AS_T and the plan is refined to: (access(-Database)) (create-move(AS_T)) (up). Now, when the stack is popped and the function access(Database) is executed, the Translator will notify the DM that there is some missing information. The DM inspects the message from the Translator and creates an instance of a clarification request, Q_P/A_P, which is inserted under the node AS_T/AS_T as a right sister of the first AS_T. When the new daughter node is created, an action for integrating information from it is added to the action plan of the mother. Then control is given to the new IR-unit, which has the task to create a question describing the failure and then wait for a user move.

5 Conclusions and future work

Most of the modules described in section 2 have been implemented as independent systems, but there is as yet no implementation of the full system. We are currently working on an implementation using a relational data base as background system and second-hand cars as the domain.

One of our requirements on LINLIN is that it should be designed for customization to various applications. While the processing modules are the same in every application, the knowledge-bases should be updated with new knowledge. This is not only the case for linguistic knowledge and domain knowledge, but also for the dialogue objects needed. For instance, in a system where you can both order things and make inquiries about the things you can order, you need to recognize order moves as distinct from requests of information. The domain and the complexity and variation of the dialogue will probably also affect the kinds of indexicality that the system has to cope with and even the ways in which reference resolution is done. Thus, the attributes of the Scoreboard and the access functions used by it are changeable.

For this reason we propose a model for NLI's that uses a “shell” which, in the same way as an expert system shell, must be updated with specific knowledge pertaining to vocabulary, grammar and object descriptions. This customization, as we call it, is performed by a language engineer whose work it is to get information from the customer on various aspects of the application and use this information to update a basic domain-independent grammar and lexicon with application-dependent words and rules, and also update, revise or construct domain and dialogue object descriptions.

A problem is though that for many applications the knowledge needed as regards the dialogue structure and also all the various lexical and grammatical updates, is not known in advance. Therefore, as suggested by Kelley (1983) who conducted simulations in two different runs with actual users to build a NLI, we see a role for Wizard-of-Oz simulations also in the context of application design.

NOTES

1. This work is part of the project “Analysis and generation of natural language text”, funded by the Swedish National Board for Technical Development (STU).
2. A Wizard-of-Oz study is a study where a subject interacts with a computer system on the assumption that it has certain properties, such as understanding human language, though in fact it doesn't. Instead the system is simulated by a person operating another usually much simpler system.
3. This dialogue, as all examples in the paper, is an English translation of a dialogue from our corpus of Swedish dialogues collected in Wizard-of-Oz simulations.
4. The distinction between an assertion of information interest and an indirect request for information is sometimes difficult to draw. If the embedded clause is specific enough to be translated into a database query, we classify the move as Q, otherwise as AS. Thus, the input *I would like to know the top speed of all BMW's* is a Q, while *I would like to know something about BMW's* is an AS.
5. The illustrative nature of this figure should be emphasized; it does not reflect an actual representation that we use. It illustrates some demands on such a representation, however, e.g. the need to distinguish actual values from type restrictions and the need for structure-sharing between moves and the IR-units of which they are part.

REFERENCES

Ahrenberg, L. 1989a. On the integration of linguistic knowledge and world knowledge in natural language understanding. In Ö. Dahl & K. Fraurud (eds.) *Papers from the First Nordic Conference on Text Comprehension in Man and Machine*, Institute of Linguistics, University of Stockholm, pp. 1-11.

Ahrenberg, L. 1989b. A Constraint-Based Model for Natural-Language Understanding and a Pilot Implementation. Research Report LiTH-IDA-R-89-22, Department of Computer and Information Science, Linköping University.

Ahrenberg, L. and Jönsson, A. 1989. An interactive system for tagging dialogues. *Literary and Linguistic Computing* **3**(2), 66-70.

Allen, J. 1987. *Natural Language Understanding*. Menlo Park, Benjamin/Cummings.

Andersson, L. 1989. A Study in Linguistically Motivated Extensions to Unification-Based Grammar Formalisms, Master's thesis, LiTH-IDA-Ex-8918, Department of Computer and Information Science, Linköping University.

Clark, H.H. & Marshall, C.R. 1981. Definite reference and mutual knowledge. In Joshi, A., Webber, B. & Sag, I. (eds.) *Elements of discourse understanding*. Cambridge: Cambridge University Press.

Dahlbäck, N. 1990. Discourse situations and language structure - towards a taxonomy of dialogue situations. Unpublished manuscript. Department of Computer and Information Science, Linköping University, Linköping, Sweden.

Dahlbäck, N. & Jönsson, A. 1989. Empirical studies of discourse representations for natural language interfaces. In *Proceedings of the 4th Conference of the European Chapter of the Association for computational Linguistics (E-ACL ‘89)*, Manchester, England, pp. 291-298.

Ericsson, A. & Simon, H.1984. *Protocol analysis*. Cambridge: MIT Press.

Flickinger, D., Pollard, C. J. and Wasow, T. 1985. Structure sharing in lexical representation. In *Proceedings of the 23rd Annual Meeting of the ACL*, Chicago, pp. 262-267.

Fraurud, K. 1989. Towards a non-uniform treatment of definite NP's in discourse. In Ö. Dahl & K. Fraurud (eds.) *Papers from the First Nordic Conference on Text Comprehension in Man and Machine*, Institute of Linguistics, University of Stockholm, pp. 75-87.

Gazdar, G. 1987. Linguistic applications of default inheritance mechanisms. In Whitelock, P., Wood, M. M., Somers, H. L., Johnson, R. and Bennett, P. (eds.) *Linguistic Theory & Computer Applications*, Academic Press, pp. 37-67.

Grishman, R. & Kittredge, R. (eds.) 1986. *Analyzing language in restricted domains*. Hillsdale, N.J.: Erlbaum.

Grosz, B. 1977. The representation and use of focus in dialogue understanding. Unpublished Ph.D. thesis. University of California, Berkeley.

Grosz, B. and Sidner C. 1986. Attention, Intention and the Structure of Discourse. *Computational Linguistics* 12(3): 175-204.

Guindon, R. 1988. A multidisciplinary perspective on dialogue structure in user-advisor dialogues. In Guindon, R. (ed.) *Cognitive science and its applications for human-computer interaction.*Hillsdale, N.J.: Erlbaum, pp. 163-200.

Hawkins, J. A. 1978. *Definiteness and Indefiniteness*. London, Crown-Helm.

Hirst, G. 1987. *Semantic Interpretation and the Resolution of Ambiguity*. Cambridge University Press.

Jönsson, A. and Ahrenberg, L. 1990. Extensions of a descriptor-based tagging system into a tool for the generation of unification-based grammars. To appear in *Research in Humanities Computing* 1.

Jönsson, A. & Dahlbäck, N.1988. Talking to a computer is not like talking to your best friend. In *Proceedings of the 1st Scandinavian conference on artificial intelligence (SCAI' 88)*, Tromsø, Norway, pp. 53-68.

Kelley, J. F. 1983. An Empirical Methodology for Writing User-Friendly Natural Language Computer Applications, *Proceedings of the CHI'83*, pp.193-196

Levinson, S. C. 1983. *Pragmatics*. Cambridge University Press.

Litman, D. J. and Allen, J. F. 1987. A plan recognition model for subdialogues in conversations. *Cognitive Science* **11**(2), 163-200.

Regoczi, S. & Plantinga, E. 1987 Creating the domain of discourse: ontology and inventory. *International Journal of Man-Machine Studies*, **27**, 235-250.

Rettig, M. and Bates, M. 1988. How to choose natural language software. *AI Expert*, July 1988: 41-49.

Shieber, S. 1984. The Design of a Computer Language for Linguistic Information. *Proceedings of the 10th International Conference of Computational Linguistics/22nd Annual Meeting of the ACL*, Stanford, California: 362-366.

Shieber, S. 1988. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, John von Neumann Society for Computing Sciences, pp. 614-619.

Sowa, J. 1984. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley.

Wachtel, T. 1986. Pragmatic sensitivity in NL interfaces and the structure of conversation. In *Proceedings of the 11th International Conference of Computational Linguistic*s, University of Bonn, pp. 35-42.

Webber, B. L. 1990. Structure and Ostension in the Interpretation of Discourse Deixis.Report MS-CIS-90-58, University of Pennsylvania, Dept of Computer and Information Science.

Wirén, M. 1988. On Control Strategies and Incrementality in Unification-Based Chart Parsing, Licentiate thesis, Thesis No 140, Department of Computer and Information Science, Linköping University.

Wroblewski, D. A. and Rich, E. A.1989. LUKE: An Experiment in the Early Integration of Natural Language Processing. *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas 1989: 186-191.