The Cactus UseT Architecture: an Overview of Relevant Aspects

W. Pasman, V1.0, 3/9/2

Introduction

This document discusses techniques and methods that are important for the definition of the architecture for the UseT work package in the Cactus Impulse project [Lagendijk02]. We hope that the contents suffice to enable other UseT participants to understand and participate in architecture discussions.
Numerous areas are important for UseT: personal call assistant systems, electronic assistants, language technology, context awareness, intelligent agents, action planning, trust, consistency and security. In this report we restrict ourselves to the sensing part of the i-DEA, and exclude issues within the i-DEA itself and the actions that the i-DEA can take. It will take another report to elaborate on issues inside the i-DEA.

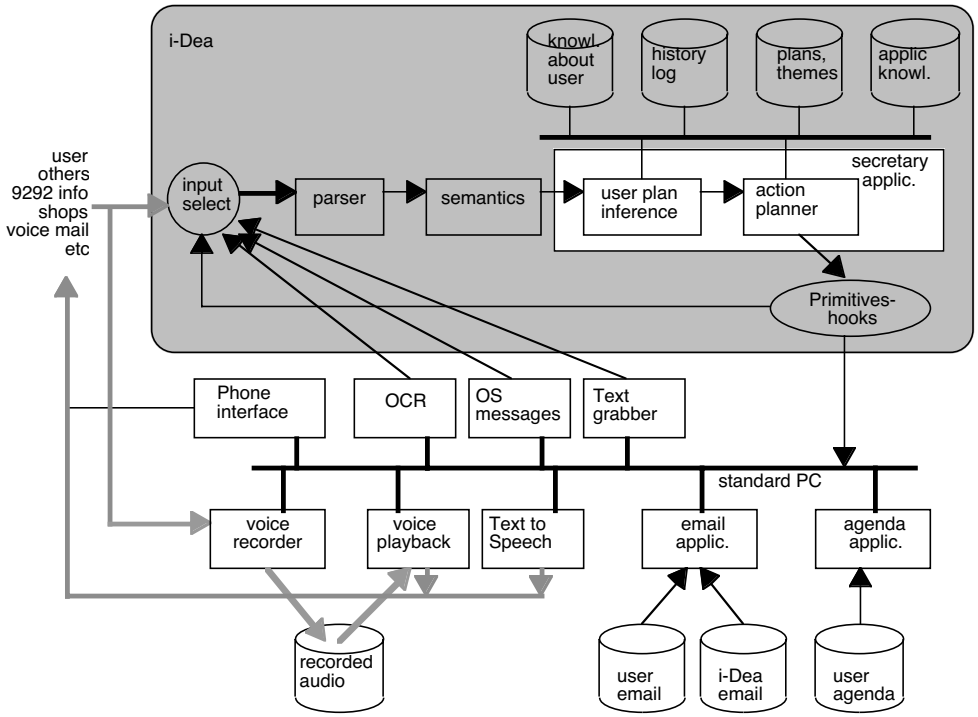

Figure 1. Sketch of i-Dea components in a tentative architecture. The 'secretary application' and the surrounding knowledge bases are of central interest for Cactus, the other components are required for demo purposes.

To give an architecture idea, Figure 1 shows a possible architecture for the i-DEA and surrounding infrastructure that the i-DEA should be able to use. Elements such as multimodality are discussed later in this report. The secretary application, the user plan inference and the surrounding knowledge bases are the central research topic

for the Cactus project, the other components are needed to get a working system but are of limited research value.
Depending on the expected functionality of the Cactus system, the knowledge representation may have to contain knowledge and rules to cover a large number of areas. Maybe these knowledge areas can be separated into nice modular packages. Traditional AI systems don't do this, but a trend to such modularity is promoted by current intelligent agent technology. To get an idea about the required knowledge, we discuss in Chapter 1 both existing electronic personal call assistants and requirements for real secretaries.
Chapter 2 focuses on the 'semantics' block in Figure 1. Here the user input is converted into a format that allows logic reasoning. Because of various ambiguities and the importance of common sense to interpret natural language, natural language processing (**NLP**) has a strong history of knowledge representation and common-sense reasoning. To give a background in NLP, Appendix A gives a brief sketch of the usual parts in an NLP analysis pipeline. It should be kept in mind that in humans, language understanding has nothing of such a nice structured, modular approach; instead parsing, inference and semantics seem to run in parallel. Apart from their usefulness to process natural language, many discourse and story representation mechanisms are powerful enough to represent the standard user interface actions as well, and thus seem a good basis for the i-DEA.
If separate semantic analysis is done for other input modalities, a modality integration step can be inserted between the semantics and the plan inference step, resulting in a single semantic representation integrating all inputs. This is discussed in Chapter 3.
The final semantic representation is then fed into the secretary application, which needs to estimate the user plans, extrapolate them and take appropriate actions. Chapter 3 discusses techniques for inference of the user's plans. Action planning is not discussed in this report.

There are a number of questions about the proposed architecture: is it feasible within the limited manpower available within Cactus, is there (commercial or prototype) software available that we can build on, and what concessions should we make and which parts are essential to make Cactus a success? The answers to these questions have impact far beyond the UseT.3 (proof of concept) and therefore we hope that this document gives a common ground to discuss the issues.

1. Personal Call Assistants

This chapter discusses existing work on electronic and human personal call assistants (PCA), and requirements that are known for secretary work that can be applied to such a system.

Electronic PCAs

Currently, commercially successful electronic PCAs are based on phoneme-level recognition (appendix A) and very simple interfaces. More advanced NLP techniques and open dialogue are not used, probably because NLP technology is not yet sufficiently reliable and foolproof to use in commercial services [Allen00]. KPN's Eileen system tries to alleviate this problem by putting a human operator between the electronic services and the human user of the system.

Fully electronic PCAs

Several attempts have been done to develop an electronic personal assistant reachable via a telephone, email and/or fax. There are a number of companies currently offering such services on a commercial basis, for instance Conita [Conita01], Executive Services International [Executive02], Cisco [Cisco02] and Webley [Webley02]. Those services focus on enabling and enhancing communication between a customer and a business representative. They mainly integrate voice, fax and email communication, so that conversion is done between the medium that a customer used to address the representative and the medium that the representative is currently using or has available. These systems often also contain functionality like agenda management and conference booking. Executive's system can also take the initiative to make calls when it detects that the user is running late. Several patents [Irribarren96, Ueda98] have been claimed for such systems.
These electronic systems have a very limited speech interface, if any at all. Following accepted current practice, the world wide web consortium is working on a Voice Extensible Markup Language (VoiceXML) that aims at a standard definition language to define user interaction via voice and the corresponding actions from the computer. This language is very 'menu'-oriented: the computer always keeps control of the dialogue, asks very specific questions including the acceptable answers, and at every point there is only a limited number of acceptable user responses. For instance the computer may ask 'do you want tea or coffee' and the user may then respond with 'tea' or 'coffee', while other responses will be rejected.

Eileen

The Dutch telecom, KPN, is running field tests with personal call assistants using a real human operator interfacing between the services and the user's requests [YPCA02]. Their 'Mabel' prototype [Kardol99] provided two types of services: communication: speech phone, email, fax and SMS, and information: ANP current news, stocks, travel info, radio, Schiphol airport info, teletekst, TV programs, travel

news and weather forecasts. The users used the system mainly on the road, once the users were at home or at work they switched back to the conventional means of communication with a PC. This caused peak hours around 8 and 18 hour. Communication services were 48% of the total use, information request 52% (Figure 2). Heavy users considered voice phone, email and SMS the most important communication services, and ANP current news, teletekst and traffic information were considered the most important information sources. Stock ratings were also checked frequently, but were not considered important (maybe because the stock ratings were not important for the professional activities of the users?).

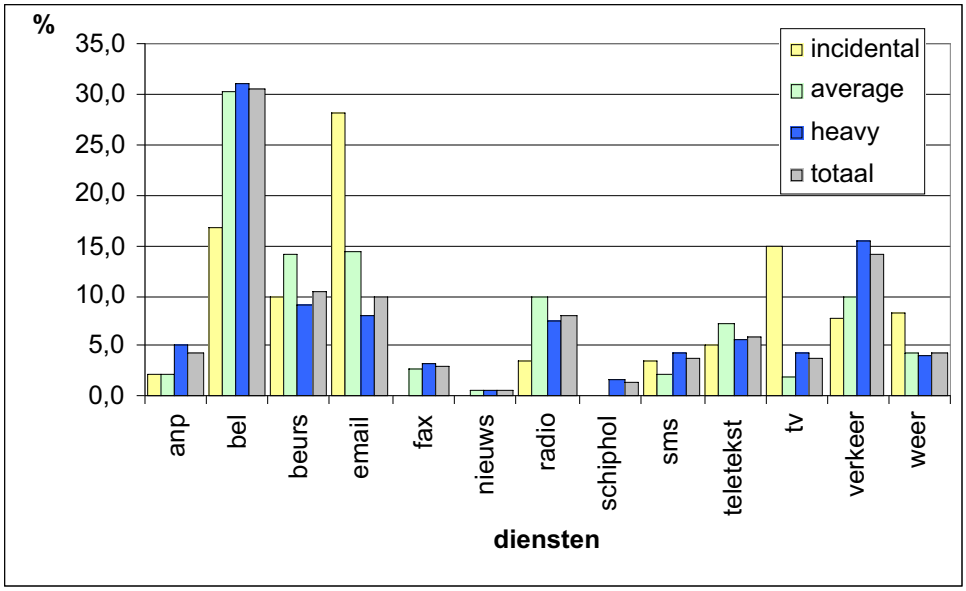

Figure 2. Actual use of various services in Mabel. From [Kardol99].

Most users of Mabel find only a few applications important, and probably supporting only the a small amount of important services does not warrant a complex speech interface as proposed earlier for Cactus. But the actual use of the system is not so black/white: in fact the weather forecast is used more often than the ANP news but probably the users find the ANP news more important for their job as the boss is paying these services. But this data does show that for Cactus a critical decision has to be made between a simple system providing only communication means, recent news and teletekst access, or a complex system more focusing on high usability and flexibility. For Cactus, only the latter seems to provide the challenges required for innovating research.

Another reason for promoting flexibility over simplicity is that it would be nice if the assistant could handle more complex requests like 'can you arrange a stay in Amsterdam for me between June 4 and 8' or 'our printer gives stripes on the paper, can you have someone look after it'. The Mabel (and Eileen) secretary is restricted to very simple information lookup, and is unable to handle such requests. It may be a good idea to subject such extended functionality to a Wizard of Oz test, to see what functionality would be appreciated and/or used often, and to see how to model such an extended functionality.

Kardol's report is also interesting for long lists of potential services, suggested by subjects that participated in their user tests. Surprisingly they all stay within the standard information-request scope, apparently no one proposed open requests. Finally, an interesting recommendation is that the interface of the computer system supporting the operator should be improved in order to keep an open dialogue between client and operator possible. It is mentioned as a first point to improve. It seems that a menu structure and closed dialogue tends to fixate both the dialogue both of the operator and the minds of the users.

Real secretary

Table 1 list the jobs of a real secretary. It is not clear to what extend these jobs can be replaced by an electronic secretary, and how desirable and useful that would be, but it gives an idea of the typical jobs of a human secretary, and how much more complex they are compared with the current electronic secretary attempts. And clearly the tasks of a real secretary would become even more complex when he was expected to follow his boss' tracks, anticipate his plans etc. It is clear that some clear and realistic goals of the expected tasks and performance of the i-DEA system have to be set (which will not be done in this document).

Table 1. Jobs of a real secretary, freely adopted from [MDCS00]. Issues marked with * are for advanced secretary levels, associated with high levels of trust and expertise level.

•	Receive and screen visitors. Respond to inquiries on variety of matters such as pending issues, protocols, rules and procedures.
•	Schedule and arrange meetings and conferences, and notify interested parties. Arrange travel, transportation and hotel for staff.
•	Edit documents to process information, handle correspondence, reports, etc.
•	Produce official documents: inform people about decisions, send orders, make a neat document from quick notes and instructions, reply to complaints, etc.
•	Transcription: for instance record minutes, transcribe taped conference and interviews.
•	Proof-read and correct documents for spelling, punctuation, format, syntax and content. Check the proper format, include necessary portions and related documents. File copies where necessary.
•	Keep databases up to date, such as agenda, address book, and administration.
•	Sort mail, sort to urgency, file as appropriate and make short abstracts where necessary.
•	Determine need for supplies, equipment, repair and maintenance services
•	Operate standard office equipment such as calculators, copiers, phones, fax, etc.
*	Briefing on matters to be considered before staff meetings and on problems and issues affecting the supervisor's area of responsibility.
*	Transfer of privileged legal and other information among staff and authorised persons
*	Maintaining calendars for others, ensuring no scheduling conflicts occur, and notify them of imposed deadlines.
*	Make recommendations for improving efficiency and economy of existing operations.
*	Assemble and summarize information from files, newspapers, journals, documents, and other sources
*	Establish forms, procedures, and standards for correspondence.

Table 2 indicates a number of knowledge areas that a secretary should master. Similar requirements probably have to be imposed on an electronic secretary.

Table 2. Knowledge areas expected for a real secretary, freely adopted from [MDCS00].

- Knowledge of office practices, procedures, machines and equipment
- Knowledge of correct English usage, spelling, punctuation, including knowledge about terminology in the expertise area that is dealt with
- Knowledge of organization and composition of business letters, minutes, reports, charts, and numerical and tabular materials
- Knowledge of techniques of receiving callers, making appointments, giving information, and explaining instructions and guidelines
- Knowledge of organisation and maintenance of filing systems related to the work
- Knowledge of administrative hearing, rules and procedures
- Knowledge of application of instructions and guidelines to specific problems arising in the work area
- Knowledge on terminology and syntax used, and of the content, organization, and format of documents and correspondence
- Ability to skilfully handle software, hardware and applications used in office environment.
- Ability to follow complex instructions
- Ability to apply instructions and/or guidelines as appropriate in order to support activities
- Ability to work with deadlines and work priorities, and to determine these
- Ability to communicate effectively
- Ability to select and apply references such as dictionaries, English usage manuals, legal references, procedures manuals and computer guides
- Ability to use diplomacy and discretion in giving out information and in referring and directing callers and visitors
- Ability to perform mathematical calculations
- Ability to transcribe documents from written, oral, or computer generated formats
- Ability to interpret instructions and guidelines in order to make decisions and take necessary actions.

Personal Secretary

We listed the requirements for a human secretary, but a normal secretary does not trace the user's step to infer his plans. A **personal** assistant thus has to have additional abilities, as listed in Table 3. The table lists also a number of optional abilities that could help acceptance of a personal call assistant.

For an electronic personal secretary other issues may play as well, for instance, how should it keep the expertise knowledge up to date. Traditionally a secretary would probably take some course every year but for an electronic system other methods may be desirable.

Table 3. Extra knowledge required for a personal secretary. Areas marked with * are optional.

- Ability to communicate effectively given the current user's context, taking into account his preferences, tastes and particular abilities and requirements of his current situation.
- Inferring the user's intentions, plans and goals, reducing the need to get explanations and instructions

* Ability to infer which knowledge the user is missing

* Knowledge about how and when to try to learn new things to the user

* Knowledge to communicate such that security and privacy are respected

2. Semantics

An important but tricky step in the language processing pipeline is the conversion of the user's utterances and actions into a semantic structure, that is describing the meaning of the user's utterances. In conventional interfaces, what the user wants is clear, as every button, menu and command has a very well-defined meaning and it is simply assumed that if the user clicks then he wants the selected action. With natural language as interface language, ambiguity and uncertainty comes in, and it is up to the semantic module in Figure 1 to consult the context and knowledge bases in order to disambiguate the user's request. Also, with natural language the possibilities potentially grow beyond the command-only level, for instance the user may give suggestions, preferences, hints or ask for help. Finally, natural language interfaces may require additional communication, when the system decides to ask additional questions to resolve an ambiguity, of volunteers to give information.
There is problem with the idea of semantics itself, which is referring to the meaning of an utterance. There is no sec 'meaning' (see [Schank80] for some superficial problems, or [Barker99], course notes on semantics for fundamental problems), and we probably are better off with focusing on the task requirements. So here we use 'semantics' in the sense of distilling and representing task-relevant information. Such an approach is also essential in order to limit the scope of the dialogue and the amount of common-sense knowledge required [Allen00].
We will first discuss basic representation mechanisms. Then we will discuss extensions of the basic representations, needed to represent natural language peculiarities.

Basic Representation

This section describes the technical methods to represent the meaning of natural language sentences. We will first discuss some methods that make an explanation down to the word-level of an utterance, then we will discuss other methods that keep more distance from the words.
The representation should allow reasoning about it, so that rules can be applied to refine the first, rough representation of the user's utterances. For efficiency, it would be nice if the representation can also be used for inferring the user's plans and goals (next section), and in the application itself.

Word-level semantics

There are two knowledge representation systems that look very different: semantic nets which are a graphical representation, and predicate logic which is a formula notation. We discuss them and conclude that there is in fact not much difference.

Semantic Nets

A traditional approach is to convert the parse trees into a semantic net. A semantic net is a graph, with 'concepts' at the nodes and 'relation' labels at the vertices. The

concept nodes of the graph contain objects, entities, activities, etc. Figure 3 shows an example showing how a semantic net can be used to represent a sentence.

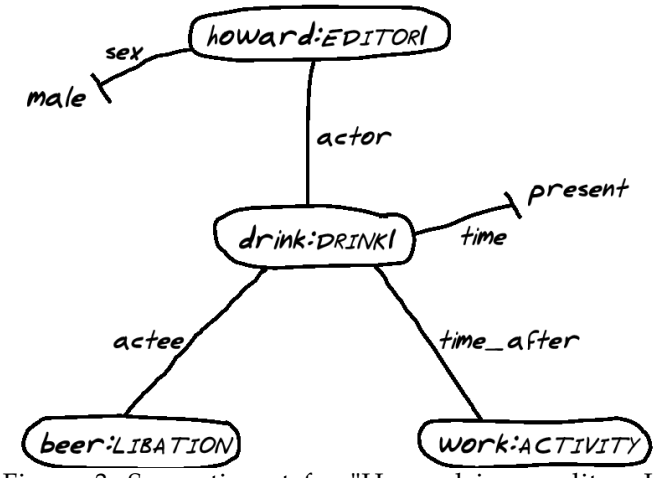

Figure 3. Semantic net for "Howard is an editor. He drinks beer after work". From [Barker99].

There are two types of semantic nets: inheritance networks and propositional semantic networks [Shapiro02]. Inheritance networks are to represent knowledge about objects in a hierarchical way, for instance, 'birds are animals', 'birds can sing' and 'tweety is a canary' and 'birds have heads' (Figure 4).

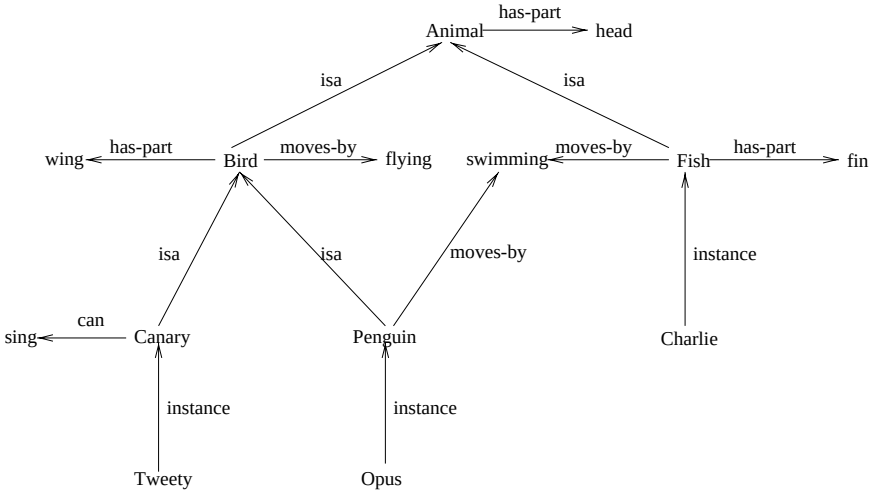

Figure 4. Inheritance network. From [Shapiro02].

The main limitation of inheritance networks is that they describe only information about relations, not about beliefs. Furthermore only information about the nodes is available, not about the relations (the labels along the links). Thus, the labels along the links require meta-knowledge not in the network.
Propositional semantic networks solve these kinds of problems by having nodes represent beliefs as well as individuals, categories and properties of the nodes (Figure 5). Nevertheless, in this representation (from [Shapiro02]) it seems that the problem is merely moved to the next level, as we now have labels like "rel", "source" and "member" along the links, which are not explained as before. Furthermore, it seems

that the only thing that happened is that the 'isa', 'class' and 'source' relations merely have moved from the nodes to the links, for instance 'source' and 'prop' still have to be paired, as an M!-node with for instance 'source' and 'arg1' seems not to make sense.

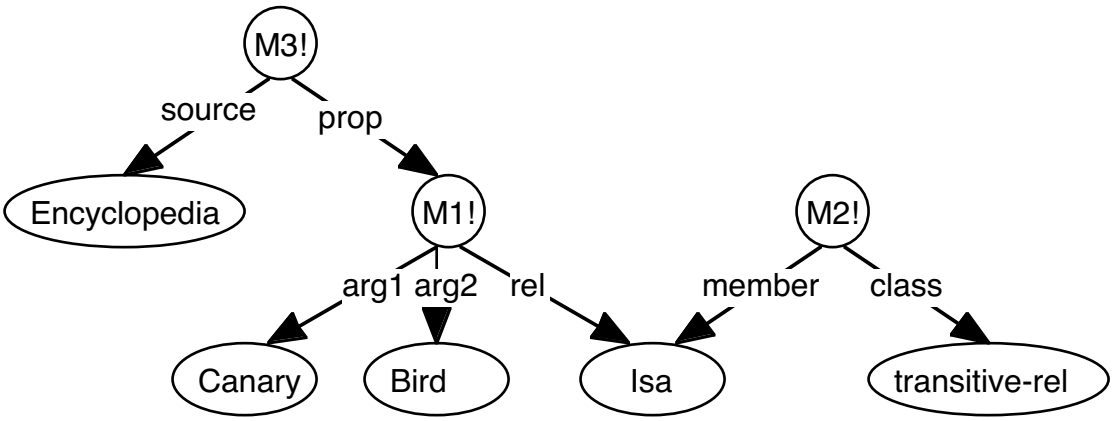

Figure 5. Propositional semantic net denoting (M1) A Canary is a Bird (M2) 'Isa' is a transitive relation (M3) Proposition M1 is from an encyclopedia.

The most straightforward solution seems to remove the labels from the links altogether, although we did not yet encounter such a proposal (Figure 6). A simple method to put traditional semantic nets into a format without link labels is to have two (label-less) links going out for each argument: one for the argument type and one for its value.

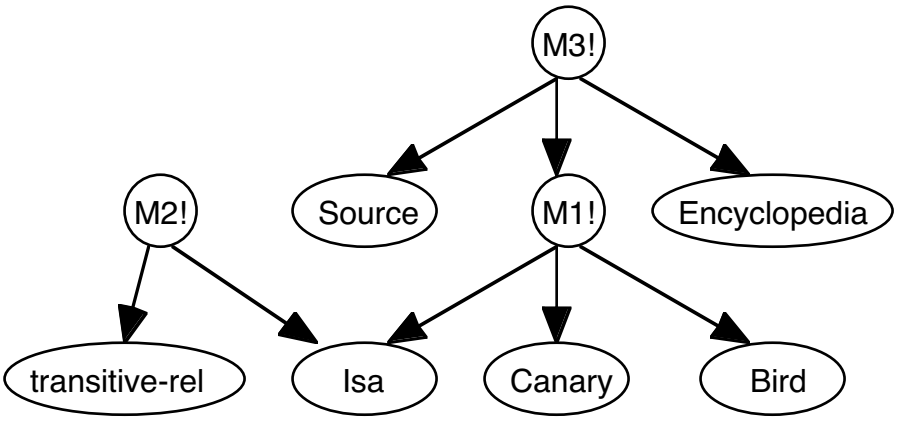

Figure 6. Alternative propositional semantic net, where all link labels have been eliminated.

Such graphs can easily converted to predicate-logic formulas (discussed below), by making a predicate for each node, having as arguments the immediate children of the node.

Predicate logic

Another popular approach in NLP is to convert the sentence into predicate logic. For instance, "Jan hit the cat" could be transferred into Person(Jan1) ∧ Cat(Cat1) ∧ Hit(Jan1,Cat1). When the primitive vocabulary is restricted, representations will get very CD-like, for instance "Jan gave Lynn a book" might be translated as

ATRANS(E1)∧ Actor(Jan1, E1) ∧ Object(Book1,E1) ∧ Recipient(Lynn,E1) ∧ Donor(Jan1,E1) where E1 is the event in question (events are discussed in 'temporal aspects' below). Standard predicate-logic formulas can easily be converted to graphs, for instance as in Figure 6.

Conclusion
Graphical notations and formula notation are of equal strength: pictures can be converted into graphs and the other way around. It seems to be a matter of taste which representation to use. What is essential is the way the predicates are defined and the reasoning is set up over the knowledge base. Unfortunately, there seems to be no standard 'format' to express knowledge, even at the predicate level. For instance 'John has a car' could be represented as Car(owner:John), as Owns(John,Car), or as Buyer(John,E1) & Bought(Car,E1). It is not clear which consequences are there to these alternatives. More knowledge is required about the various 'format' choices for predicates.

Sentence-level semantics
Previous section discussed representations that go down to the word level of the utterance. There are also methods that keep more distance from the words, such as frames, scripts and dialog acts.

Frames and Scripts
The original term 'Frame' apparently comes from Minsky [Minsky75], who did not give a detailed technical description but merely sketched how a data structure could contain stereotyped information in order to represent known and familiar situations in a coherent way.
The ideas behind frames and scripts are very similar to semantic network constructions, but they also link procedural information into the net. For instance, a frame may contain a default description for a cube in front view, but also additional links for what to do when a part of the cube is occluded (e.g., try the 'table' frame , or try the frame for a cube seen more from the right).
Scripts ([Schank80]) list typical activities involved with a certain activity. For instance a restaurant script contains information about being seated, getting the menu card, paying etc.
Once the right script has been found, actions are 'explained' as being part of the script. Furthermore, future steps can be predicted or even suggested to the user [Rich00]. A problem with the script approach is that it is difficult to detect which script is running, and that slight variants of the script may render the detection even impossible. Adding many variants on the scripts would defeat the purpose of scripts as describing a stereotypical situation [Wilensky82].
The amount of action detail that can be caught by scripts depends on the detail level described in the scripts, but typical scripts as the restaurant-script suggest that scripts give only a very coarse description of the real event. It is possible to integrate scripts in systems that go down to word level, for example the semantic nets of Norvig [Norvig87]. Norvig does not exploit the advantages that the availability of scripts offer, but theoretically this seems possible.

Dialog acts

Dialog acts, also called speech acts or performatives, assume that the speaking is performed in order to have some effect on the hearer. Thus, dialog acts define the motivation behind larger pieces of text, typically a sentence.

Typically, dialog acts represent some global knowledge about the utterances that seems very handy for making abstracts [Alexandersson00b] and for globally following the dialogue, but in them selves they are not very helpful to accurately represent what is going on.

As an example, Figure 7 and 8 show a hierarchical structure and an annotated dialog of dialog acts in VerbMobil [Alexandersson00]. Verbmobil [Wahlster00]. VerbMobil is a translation system, but even for translation there is a need to follow the discourse because the context often determines the proper translation (for example, 'platz' can translate to 'seat' and to 'hotel room').

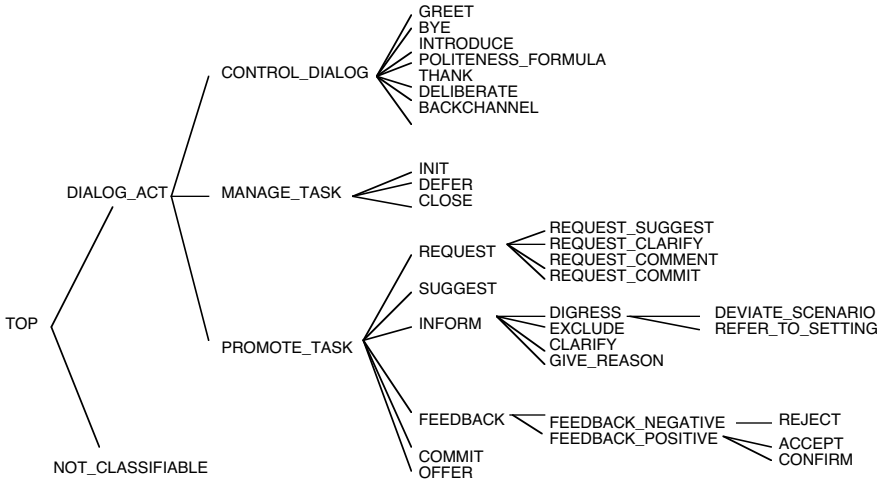

Figure 7. Hierarchical structure of dialog acts in VerbMobil. From [Alexandersson00].

```
Guten Tag, Herr Strassburg. (GREETING_BEGIN) grüse Sie (GREETING_BEGIN)
also mit den zweitägigen Terminenwürde das gut aussehen. die Woche vom
dreizehnten November bis siebzehnten. (SUGGEST) und im Prinzip könnten
wir sagen hätt ich die Zeit. Wochenende mal unberüchsichtigt. vom ersten
Dezember bis Fünfzehnter Dezember (SUGGEST) da sieht das bei mir im
Terminkalender also gut aus (INFORM) da könnte ich hab'ich noch einige
Freiräume (INFORM).
```

Figure 8. Example dialog-act annotated text. from [Alexandersson00]. (Prosody and other tags have been removed.)

Dialog acts are also used in the Knowledge Query and Manipulation Language (KQML) and the FIPA Agent Communication Language [FIPA01], both defining a standard for communication between software agents [Labrou97]. Those software-protocols can also be used for representing human dialog acts [Allen00]. However, as can be seen in Table 4, KQML performatives have much less detail as the VerbMobil dialog acts. Control dialog in natural language takes a very different form in software agents, but some mapping may be possible between the two. Constructions as digress, give_reason and request_comment are not available in KQML.

Table 4. KQML performatives. From [Labrou97]

Discourse	ask-if, ask-all, ask-one, stream-all, eos, tell, untell, deny, insert, uninsert, delete-one, delete-all, undelete, achieve, unachieve, advertise, unadvertise, subscribe
Intervention and mechanics	error, sorry, standby, ready, next, rest, discard
Facilitation and networking	register, unregister, forward, broadcast, transport-address, broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all

Conclusion

Higher-level constructs can help to recognise plans, make abstracts and predict future user actions. Most of the discussed methods don't capture the details of an utterance. For the i-DEA, explanations down to word-level are required because the secretary will need details at that level. So in order to be usable for the i-DEA, high-level semantics will have to be combined with word-level semantics. With software protocols as ACL and KQML, fields are available with each performative, to put in further details.

Extensions

There are several problems with the techniques from the previous section. Simple predicate logic is insufficient to catch several natural language details that may be relevant. The following sections look at attempts to limit the number of semantic primitives, the relation of the task with time and ordering representation, representation of quantifiers, discourse representation, problems with constructing a large common-sense knowledge base, and some other problems.

Limiting the number of primitives

General semantic nets do not restrict the number of concepts and relations. The underlying idea is often that there is no meaning of the individual terms but that the relations between the terms – the net itself – can be useful. But there are also other approaches that restrict the number of primitives and give them a tight meaning.

One example of tighter semantics are the performatives used to represent speech acts in recent systems and standards, as in VerbMobil [Wahlster00], KQML and FIPA ACL discussed in the previous sections. However, it was already clear that these need additional word-level logic to catch dialog details, so although their semantics are tight they are not powerful enough nor exhaustive.

A more rigorous approach, attempting to be both powerful enough and exhaustive, is the Conceptual Dependency notation (**CD**) [Schank72]. In CD notation there are about 11 to 14 primitives to represent all possible actions (depending on the variant), falling apart into general physical acts, animate physical acts and mental acts. Because of this low number of primitives and simple structures, this is an interesting alternative for an interlingua. This representation is also interesting for inference, as there are only a few primitives to reason about. Finally this representation seems to offer possibilities for a semantically driven parser instead of a syntactically driven one, which seems a good idea in the context of spoken language with all its problems [Wilensky02].

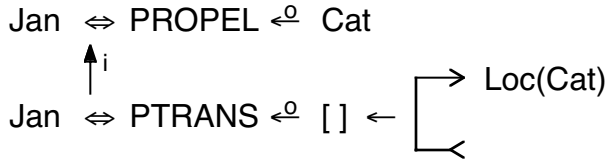

Figure 9. Conceptual dependency representation for "Jan hit the cat", roughly meaning "Jan moved the cat by applying propelling force to it".

Unfortunately there are some major problems with CD, as can already been seen in Figure 9. Concepts have to be 'forced' into the primitives, garbling and missing part of the meaning. There are no quantifiers in CD, making it impossible to represent for instance 'Jan hit all cats', but this problem can probably be solved. Finally, up to now semantically driven parsing did not work out, there are problems with for instance "Jan learned Pat left town", where such a parser would assume Pat to be the actor of learning.

Inheritance and overriding

A convenient way to describe properties of types (real-world objects, concepts, etc.) is to use inheritance. For instance a class birds can be defined, with properties as wings, feathers and can-fly. Seagulls and canaries can then be defined as being birds, without having to re-specify the wings etc. But sometimes one of the properties has to be overridden, for instance when defining a penguin the can-fly property has to be removed. Such override-possibility can pose serious problems to predicate logic. For instance, at some point during a discourse it may become known that tweety is a bird, and it is concluded that tweety can fly. Later, it is discovered that tweety is a penguin, and then the conclusion should be retracted. With normal logic there is no such action as retracting facts and inferences made with these facts, but if the system keeps track of why facts and inferences were made such a system can be made to work. Such a system is called a truth maintenance system.

Time and ordering

If a truth maintenance system is used, the knowledge base always represents the current state of affairs. However, this eliminates the possibility to find out about past situations and the possibility of explaining why things happened. To do that, some notion of time has to be incorporated into the predicates. A number of time-logics can be distinguished, we shortly discuss modal logic, temporal logic, event calculus and situation calculus (see [Garson01, Shapiro02] for a nice but partial introductions). Modal logic, also called tense logic, has no explicit time but expresses notions like 'it has always been that ...' and 'it will always be that ...'. As such, it distinguishes past, present and future. For example, the truth operator G may stand for "It will always be the case that" and F may stand for "It will at some time be the case that ...". A tense logic formula may then be $Ga \rightarrow Fa$ ("If always a is the case, then a is the case at some time"), or $F\exists x(Philosopher(x)\&King(x))$ ("There will exist someone who is at the same time both a philosopher and a king") [Galton99].

Temporal logic uses explicit time stamps or time intervals. For instance Drink(John,Water, 11:33) stands for something like "John (drinks/drank/will drink) water at 11:33". In temporal logic, explicit rules or mechanisms are required to 'forward' predicates in time, to avoid having the knowledge engineer to write many

rules like In(Table3,room5, T) & not(Move(Table3,T) –> In(Table3,room5,T+1) (this is called the frame problem, not to be mixed up with the frame representation for semantic knowledge).
Event-token reification and Event Calculus [Kowalski86] separate the time information from the other information by using the notion of an event. For instance John's drinking could be represented with Drink(John,e) & Time(e,11:33) & Contents(Water,e). This seems a good idea, as there are more attributes related to events, reduces the number of attributes for the Drink proposition and does not force a time stamp on all events.
It is not very clear what gives events such a special status as compared to say, an object. For instance, one could also write (Event(e1) & Male_Human(j1) & Drink(j1,e1) & Time(e1,11:33) & Name(j1,John)). Instead of 'events' that express some change, also 'situations' can be defined that instead express some stable state, as in Situation Calculus (e.g., [Clark00]). It seems that introducing a new variable to denote events, situations, or objects is useful when the information concerning events, situations, or objects is coming and inferred in several steps.

In natural language, there are at least five ways to express time [Alexandersson00]:
(1) simple, referring to an absolute time, for example "Ten past twelve on Monday"
(2) modified: simple but a bit fuzzy, such as "about four o'clock"
(3) span: a time interval defined by duration only ("let's meet for two hours") or by two simple points in time.
(4) referenced: relative to some point, for instance "a week from Friday"
(5) counted: uses counting to identify point or interval, as in "last Sunday in March" or "third week in May".

In VerbMobil, these notions can be used rather directly for shallow translation, but for deep processing further disambiguation is required because most temporal descriptions are heavily underspecified [Koch00]. For such cases, the interval is translated into a uniform format with one or two points in time with a year, month, day and time where for instance the day can be expressed as day of week or day of month in order to be able to represent both 'Monday' and 'the fourth'.
In story understanding, time is often implicit. In such a case it is usually assumed that the story is told in chronological order, and instead of a time the sentence number can be used. It is not clear whether such tactics are required for natural language processing, but a uniform format for time and temporal reasoning seems essential for the i-DEA.

Discourse issues

Discourse is more difficult than story understanding, as there are now multiple participants, there is turn-taking, barge-in (interrupting a user), and the system now interacts with the user giving rise to real-time requirements, issues about the appropriateness of taking a turn, etc. Unfortunately the term 'discourse' is usually used in the story understanding context (e.g., [Hess91], [Scott97], [Wiemer-Hastings02]) and does not deal with mentioned issues.
Schiffrin discusses a number of lexical items or 'discourse markers' that are typical for conversational speech, such as 'oh', 'well' and 'you know'. Although relevant for conversational speech, the relevance for discourse is limited.

Dialog act theory seems powerful enough to analyse the reasons of barge-in and turn-taking. However, this theory is aiming at categorization of utterances, and seems less suited for predicting user actions and for making a suggestions which steps are appropriate for the system. Maybe they can be combined with a more plan-based approach.
Cohen [Cohen97] distinguishes three discourse modeling mechanisms: dialogue grammars, plan-based models of dialogue, and joint action theories of dialogue. It is not clear how dialogue grammars (which 'explains' events by detecting typical pairs, such as a question/answer pair) could account for instance for turn-taking and barge-in. But plan-based models of dialogue, which explain events by explaining how they serve the underlying plan of the speaker, could be used to explain events like barge-in and turn-taking. For suggesting barge-in real-time aspects of plan-based models have to be developed further. Joint action theories of dialogue, which see dialog as a multi-player game (see also [McBurney01]), could also be used to explain discourse, but as far as we know concrete models still have to be developed.

Quantifiers

Quantifiers are required to represent sentences containing quantifiers like 'all', 'some', 'a few' etc.
Semantic nets require a modification even to introduce the familiar existential and universal quantifiers, for instance like the ANALOG system [Ali93]. Figure 10 shows the ANALOG representation of "Each farmer that owns a donkey beats it" (M1!). Variables are introduced with a 'some' or 'all' quantifier. V1 represents "Every farmer that beats a donkey he owns" and V2 represents "a beaten donkey that is owned by any farmer".

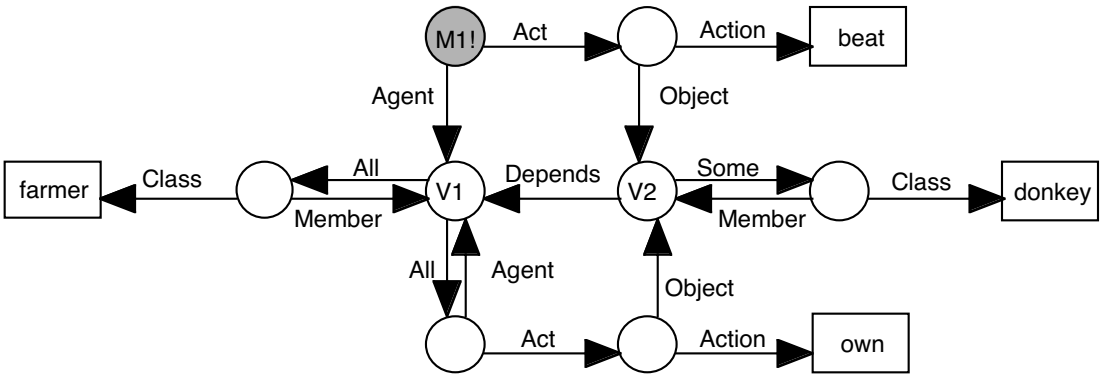

Figure 10. ANALOG representation of "Each farmer that owns a donkey beats it".

Linear predicate logic formulas can not catch certain natural language constructs, because of the scope restrictions in linear predicate formulas. A typical –although quite synthetic– example is "Some relative of each villager and some relative of each townsman hate each other", which needs a kind of "branching quantifier representation" as in Figure 11. Another famous example giving problems with

predicate logic is "Each farmer that owns a donkey beats it", where predicate logic can come only as close as "all farmers beat all donkeys they own".

$$\left.\begin{matrix}\forall x \exists y \\ \forall z \exists w\end{matrix}\right\rangle [(villager(x) \wedge townsman(z)) \Rightarrow (relative(x,y) \wedge relative(z,w) \wedge Hates(y,w))]$$

Figure 11. Branching quantifier representation for "Some relative of each villager and some relative of each townsman hate each other".

Representing ambiguities from language efficiently is especially tricky for quantifiers. For instance in the donkey sentence it is not clear whether each farmer beats only one of his donkeys, or all donkeys he owns. For instance the famous sentence "every man loves a woman" can be interpreted as $\exists w \forall m(Man(m) \rightarrow (Woman(w) \wedge Loves(m,w)))$ (there exists a single woman that every man loves) or as $\forall m \exists w(Man(m) \rightarrow (Woman(w) \wedge Loves(m,w)))$ (Every man loves at least one woman). When encountering such a sentence we have to represent both formulas, until it becomes clear which one is appropriate. In this example the notation could be $\{(\forall m \ \ Man(m))_{q1} \ (\exists w \ \ Woman(w))_{q2}\} Loves(m,w)$, and once the order is determined an additional assertion, for instance $q1 < q2$, can be added (e.g., [Wilensky02], lecture on quantification and scope).
Clearly quantification happens often in natural language, but from the examples considered here it appears that the ambiguous quantifier issues are not very important. Probably a clarification dialog can be set up in case such an ambiguity arises, or maybe even a blunt choice for one of the alternatives is sufficient.

Probabilistic rules

Probabilistic logic adds probabilities to each predicate and rule. Such a mechanism could be used to account for the non-certainty of most real-world knowledge. There are a few types of probabilistic logic: Bayesian logic, Dempster-Shafer theory and fuzzy logic. Bayesian logic relies on the formula (P(Y | X)=P(X | Y) P(Y)/P(X)) to determine the probability of combined propositions given individual probabilities of each proposition. Dempster-Shafer theory has a similar but more complex basic formula. An advantage of Dempster-Shafer theory over Bayesian logic is that it can distinguish NOT-X cases from no-info-on-X cases. A disadvantage is the sometimes counter-intuitive behaviour of Dempster-Shafer theory.
For large knowledge bases, using probabilistic rules becomes unmanageable [Lenat95]. A big problem with creating a probabilistic knowledge base is the excessive number of probabilities which have to be determined reliable and by hand [Zukerman01]. Furthermore, temporal dependency (see the section 'time and ordering') seems hard to combine with probabilistic logic. Reasoning with probabilities is more expensive than conventional reasoning, typically such reasoning has to deal with very large state spaces and require efficient numerical methods to handle those [Boutilier02].
Cyc [Lenat95] apparently does not offer probabilistic reasoning. ABLE [Bigus01] uses fuzzy rules, but only in combination with forward chaining. KODIAK [Wilensky02b] allows assignment of probabilities to assertions, for instance(add-fact '(val location monkey2 chair) :truth .7) indicates a 70% chance that monkey2 is in the chair. This is a little surprising, as for instance Norvig uses quite a large database of about 600

concepts [Norvig87] without performance problems. So there might be possibilities to cope with the complexity of this problem, but it more likely is the case that Norvig does not use probabilities and that KODIAK does not use probabilistic rules in his case.

Other problems

There can be the need for ambiguity about what a modifier modifies, for instance in "ugly yellow car" it is unclear whether the yellow or the car is ugly. Although it is possible to represent this ambiguity efficiently, as far as we know there are no semantic level formalisms for this except as just a disjunction of the alternatives.
Although we focus here on the representation of natural language sentences, it is unclear to what extend these notations are effective for reasoning and knowledge representation itself. It seems that the underlying idea of many of the theories in this area is that the knowledge system has to be at least as strong as the system representing sentences from the natural language. However, it is not necessary that a single mechanism keeps all the facts, instead several possible interpretations of sentences could be handled by separate mechanisms. Furthermore, those interpretations need not be made at the time of hearing (the approach taken by the majority of the methods we are aware of) but could be deferred until evaluation is necessary (as in [Norvig87]), and an evaluator looking for specific information could be used at that time, for instance disambiguating only where needed (as in [Beek91]). Only behind the planning level in the application, all ambiguities that could have effect on the appropriateness of an action have to be resolved before action can be initiated (assuming actions can not be undo-ed).

Reasoning over predicates

As we discussed, there are many possible extensions of predicate logic, each designed to catch a specific natural-language phenomena. However, the predicates will have to be used for reasoning, and unfortunately every new problem description language requires a new planning algorithm (a planning algorithm searches for a legal sequence of actions to reach a goal, such as the goal to find plans behind an utterance), and there are many such languages – there are at least forty [Mueller02] for story understanding alone! This problem highly restricts reuse of existing knowledge bases.
Communication standardization efforts like KIF [Genesereth98] and KQML [Finin97] define the interchange format (that is, the notational syntax and semantics), but not how to use or format the knowledge. Standardization attempts like Open Agent Architecture (OAA), that use light variants of KQML for communication, leave it to the individual agents to use the knowledge.

Conclusions

Many extensions are highly desirable, only probabilistic rules and exotic quantifier issues seem dispensable when dealing with real-world problems.

3. User modeling

User modeling can refer to simple estimation of user expertise level, up to detailed, dynamic recognition of user plans. The first can also be seen as facts about the user, and are discussed first. The last is about inferring and extrapolating current intentions of the user, and is often named 'plan recognition'.
Learning can come into play with user modeling as well, and is often seen as the key factor distinguishing agents from standard software. For user modeling, this amounts to some sort of preference profiling. But several other possibilities can be thought of, especially when it comes to plan recognition and explicit user instructions. However this is a complex issue, and in this report we will not discuss the topic further. The FIPA98 standard [FIPA98] gives a starting point for further information on this topic.

User 'facts'

Which static user data is relevant highly depends on the system's and user's task. Table 5 shows some applications and the parameters that were used for these applications. Two types of 'facts' can be distinguished: stereotypical data, which assumes that typical user characteristics can be coupled to typical user behaviour and needs, and personal facts. Usually the facts each have a discrete set of possible values, for instance novice, intermediate and expert.
In some cases, e.g. [Chin88] the knowledge itself is also categorized in a 'stereotypical' way – it is assuming that there are categories of users related to these classes of knowledge. This is especially useful in consultancy- and teaching systems, where the user will usually ask questions about this knowledge. In that case the system can quickly adopt its explanations about its knowledge to fit the user category.
Stereotypes assume that a single classification over all users and/or knowledge makes sense with respect to the task of the system. If the users and data are more complex, stereotypes may be not accurate enough. Stock [Stock93] has an interesting alternative approach, using a network of interest areas, associated with a set of concepts. The activation of a node, for instance because the user asks about a particular concept or area of interest, results in the activation of nodes in areas connected to this node by activation links, and the inhibition of nodes connected to this node by inhibitory links.

Table 5. A few examples of tasks and the relevant user model parameters.

Task	User model parameters
Advising a book for reading [Rich79]	sex, educational and intellectual level, preference for thrill, fast-moving plots or romance, tolerance for descriptions of sexuality, violence and suffering, comments on previously recommended books
Unix Consultant [Chin88]	User expertise level in Unix domain (novice, expert, intermediate, beginner)
Describing complex physical objects [Paris89]	Background domain knowledge (novice, intermediate)

Giving health advice [Hirst97]	basic information from a patient's medical record, information about a patient's attitude to health care, e.g. locus of control and desire to read technical detail,
News personalisation [Yahoo02]	Birthday, gender, country, industry sector, title, specialization, interest area (entertainment, health, shopping, travel, free stuff, business, etc.),
Office on-the-fly personalisation [Ward01]	RFID Tag to identify person and location.

Numerous of standards have been proposed for user modeling, for instance the BGP-MS system [Kobsa95], the user modeling component of the FIPA98 specification [FIPA98], the P3P user model [W3C02], Vcard [vCard96], PICS, NPS, etc. The P3P standard allows data sets (lists of attribute-value pairs) to be grouped and ordered hierarchically. In BGP-MS the user modeling fits in a larger context of user- and system- beliefs about goals, plans etc. The FIPA standard allows incorporation of many of these standards. Additionally it offers several learning techniques for more advanced user modeling.

Accuracy of the information

The required accuracy and reliability of the estimation of the user's knowledge and facts about the user depends on the system's task. For medical applications a questionnaire seems the way to go [Zukerman01], as medical systems need much and accurate information in order to get a good idea about the patient's situation and to give proper advice. For consultancy systems, such as the Unix Consultant [Chin88], where typical session times are short, a quick estimation based on the first user utterance may be the only option. A conservative approach has to be taken, but there are no big penalties for mistakes. Estimation of the user's progress is not really an issue in medical and consultancy systems. For systems with longer session times, such as in intelligent tutor systems, a detailed track record of the student's progress is required, and careful initial setup is also important. In tutor systems, detection of skills and misconceptions are of prime importance (e.g., [Errico96, Trella00]). For electronic secretary systems, skills and misconception detection is less relevant, if a misconception would be detected it might be more likely that the secretary is wrong. Not only the user progress but also the progress on the jobs at hand is important in this case.

A personal secretary system will have frequent user interaction, and therefore it seems that an extensive user knowledge model has to be built up, similar to tutor systems. If the secretary system also has to teach the user, for instance about the use of the secretary system itself, there will be overlap with existing tutor systems. However different information will have to be acquired as well, as the main purpose of the secretary system is probably to give service and not teaching.

Plan recognition

Although often the term 'Plan recognition' is used for this section, the term might suggest that it is only about representing his utterances in a nice framework. However, the central goal is to explain and extrapolate user input, of which plan recognition 'sec' is only the first step. The main problems in explaining the user's intentions and extrapolating them are robustness in the face of noisy input, effective

discrimination among competing hypotheses, and the scaling up recognition algorithms to large domains [Carberry01].
We attempt to keep issues within the i-DEA application itself out of this report, but clearly there will be interactions between the user's plan and the capabilities of the i-DEA, as we are mainly aiming at recognising user plans that the i-DEA can augment or support. Furthermore, plans available for recognition of user's plans could also be used to be executed, thus taking over part of the user's job. It is not clear whether the same knowledge structures can be used for this, or that different information is needed for recognising and executing plans and actions. The TRIPS architecture [Allen00] suggests that these issues can be separated, but the semantic power of the KQML performances that he uses seems not strong enough to handle multi-modal integration.
We discuss four different approaches: demons, semantic net inference, script recognition and explicit plan-inference rules.

Script recognition

One way to explain the user's actions is to search for the script best fitting the user's actions. For instance, if it can be determined that the user is running the 'restaurant script', his actions of getting into the car can be explained as the first step of that script: getting to the restaurant. Once the current script has been found, it can be used to support the user, for instance the system could phone the restaurant to book a table.
Recognising which script best fits the actual events can be difficult, especially if a light variant of an action is used, or when interruptions occur that force two scripts to be played at the same time or one to be aborted to be picked up later. Carberry [Carberry01] and Wilensky ([Wilensky02], lecture on Plans) discuss the issues in more detail.
Collagen [Rich00] accepts only well-defined actions and these fit only at certain scripts; once enough actions have been encountered to determine a unique script, that script has been 'recognised'. Once the script has been recognised, Collagen can propose the remaining steps, and can take over some of the steps. If the actions are less well-defined, this approach becomes difficult to apply.
Scripts can also be used to detect misconceptions and errors, for instance recipes and classifications have been developed to derive possible mis-actions from known plans [Calistri-Yeh91].
As already discussed, script recognition needs extension with word-level mechanisms in order to be useful for the i-DEA. For instance, that the user's plan is to have Indian food is a detail not available in the restaurant script, but still essential if the system is going to make restaurant suggestions or wants to phone the restaurant.

Semantic net inference

The FAUSTUS system of Norvig [Norvig87] infers the user's intentions by determining which concepts and relations between the concepts are activated by the user's utterance. For this he needs a large concept knowledge base, which is based on KODIAK [Wilensky02b] and contains a semantic net of all concepts and words used

in context of the user's task. As an example, Figure 12 shows a small part elaborating on the relation between having and giving.

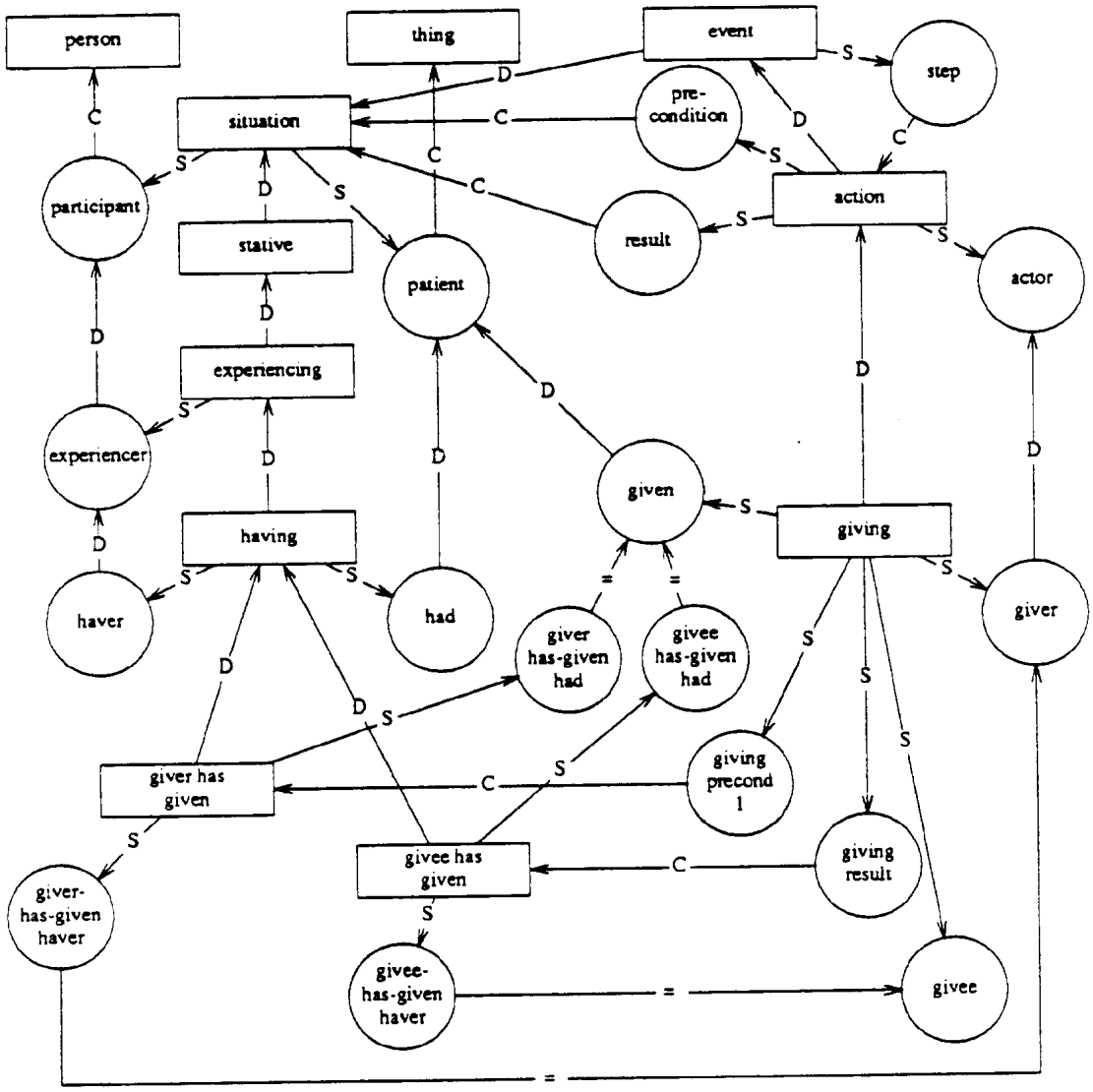

Figure 12. Part of Norvig's semantic network, describing the relation between having and giving. From [Norvig87].

FAUSTUS finds activated nodes from a given input sentence by using 'marker passing', as follows. Words in the text are marked in the semantic network. Then, links from these starting points in the network are followed, to find the connections between them. The nodes where such connections first show up are the 'collision points'. The path between the words in the text and the collision point uses various types of links (see the letters along the links in the figure 12), and exactly which link types are used determines the kind of relation between the two words. Only certain types of collisions are meaningful, most can be thrown away.

Meaningful collisions subsequently trigger inferences, and the type of inference depends on the paths to the collision. As an example, we look at the inferences made with "John was lost. He pulled over to a farmer standing by the side of the road. He asked him where he was.". Only after the third utterance, FAUSTUS makes the two

main inferences. It recognises that being near the farmer is related to asking him a question by a precondition relation (and resolves the pronominal references while making this connection). FAUSTUS could find this connection because both the asking and the being-near are explicit inputs. The other connection is a little trickier. The goal of knowing where one is was not an explicit input, but “where he was” is part of the last utterance, and there is a collision between paths starting from the representation of that phrase and another path starting from the asking that lead to the creation of the plan-for between John’s asking where he is and his hypothetical knowing where he is. (p138).

A big advantage of this system is much about the relations is stored in the graph, instead of a large collection of rules. This makes the system much easier to maintain, to keep consistent, and also seems to give better opportunities for automated learning of the graph.

A problem with this system is that some steps in natural language are too large to be recognised. Consider "Willa was hungry. She picked up the Michelin guide.". A well-informed listener knows that the Michelin guide is a guide for restaurants, and he probably will infer that Willa's plan is to go to a restaurant to get some food. Wilensky [Wilensky82] needs 7 steps to infer this, which is too long for FAUSTUS– the distance between concepts related to picking up a Michelin guide and those related to being hungry are probably too large. Norvig does not see this as a problem, as he thinks humans also have difficulty inferring so many steps.

Another problem is that we are not aware of mechanisms to extrapolate the user's actions based on semantic networks. Information to do this can be available in the net, for instance a restaurant script can be available in some form (Figure 13). This 'script'-node is just another node in the semantic network, and is inferred from the facts just as any other node. Therefore, it is not clear how to determine which aspects of the net are suited for extrapolation of the user's actions.

```
(A EAT-AT-RESTAURANT (^ EATING CONTRACTUAL-EVENT)
(diner SENTIENT-AGENT (^ eater))
(waiter-role WAITER (^ participant))
(food-role FOOD (^ patient))
(eat-at-restaurantssetting RESTAUPANT (^ setting))
(going-to-restaurant-step TRAVELING-TO-RESTAURANT (^ step) 1 l?)
(ordering-food-step ORDERING-R-FOOD (^ step) I 1)
(food-arrives-step TRANSFERRING-R-FOOD-TO-TABLE (^ step) 1 l?)
(main-restaurant-step EATING-R-FOOD (^ step) 1 l?)
(pay-for-food-step PAYING-FOR-R-FOOD (^ step) 1 l?)
(leaving-restaurant-step TPAVELING-FROM-RESTAURANT (^ step) 1 l?)
(= eat-at-restaurantssetting (destination going-to-restaurant-step))
(= eat-at-restaurantssetting (source leaving-restaurant-step))
(= food-role (patient ordering-food-step))
(= food-role (patient food-arrives-step))
(= food-role (patient main-restaurant-step))
(= food-role (merchandise pay-for-food-step))
(= diner (traveler going-to-restaurant-step))
(= diner (traveler leaving-restaurant-step))
(= diner (party1 eat-at-restaurant))
(= waiter- role (party2 eat-at-restaurant))
(= food-arrives-step (partyls-obligation eat-at-restaurant))
(= pay-for-food-step (party2s-obligation eat-at-restaurant) ))
```

Figure 13. Restaurant 'script' in semantic network. From [Norvig87], p129.

Finally, semantic net inference seems to miss the notion that the user has plans and goals, and that there are alternatives for these [Wilensky02]. Norvig claims that both script- and goal-based processing can be reproduced by his system that has no explicit processing mechanism aimed at one type of story or another, but just looks for connections in the input as they relate to what is known in memory. However, as we saw the mechanisms are quite different and his claim seems not proven sufficiently.

Logical plan and goal inference

With logic plan and goal inference, logic rules are used to detect user plans. Usually the user is assumed to have one of the few top-level goals or themes known by the system. The user's actions are explained once a logical derivation has been found towards one of those top-level goals.
The following example from Wilensky [Wilensky02] illustrates the difference nicely: "John went to a restaurant. When the waiter handed him a menu, he realized that he had left his glasses at home. He asked the waiter to tell him what was available." Wilensky argues that the restaurant script will fail on this case, and what is needed is (1) John wants to know what is served in the restaurant (2) that this is a sub-goal of deciding what to take (3) that reading the menu has preconditions, amongst others one's glasses if one needs them (4) that asking is an alternative plan for knowing.
We use the term 'logical plan inference' for such a mechanism. Such rules are more general than scripts, as they describe the basic ways in which actions and intentions are related, and if properly set up they should apply to every situation.

PAM

Wilensky's Plan Application Mechanism (PAM) [Wilensky83] is a good example of the traditional approach to logic plan inference. It assumes that the shortest such path was the proper explanation, ignoring the current focus of attention in the story. Furthermore, only the plans and goals in the story are estimated; the system does not attempt to recognise the plan of the agent asking the questions, but only looks at the format of the questions asked.
Figure 14 shows the program flow through Wilensky's Plan Applier Mechanism (PAM), which is basically an iteration through all input sentences, each sentence being accounted for by an incremental update of the story representation. PAM can understand (in the sense of make a logically coherent construction out of) quite complex stories such as "The Vice President wanted to become president. He got some arsenic". Paradoxically PAM has some troubles with for instance "John was tired. He ate a sandwich": because both being tired and eating a sandwich are basic needs that don't need any explanation, PAM does not try to find a correlation between the two sentences. Another problem is that PAM does not make any predictions, but probably PAM can be improved easily. Alternatively, there are comparable incremental story understanding systems that might be better suited for this, such as [Carberry90] which is capable of following longer dialogues for which the plan is not clear from the outset.

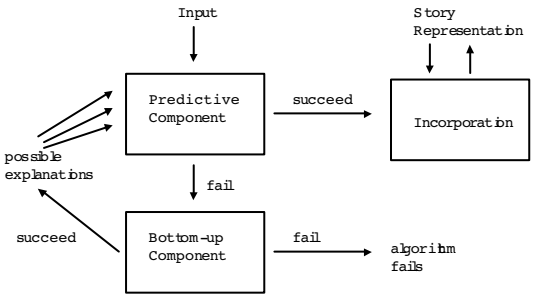

Figure 14. Discourse analysis flow of PAM. From [Wilensky02].

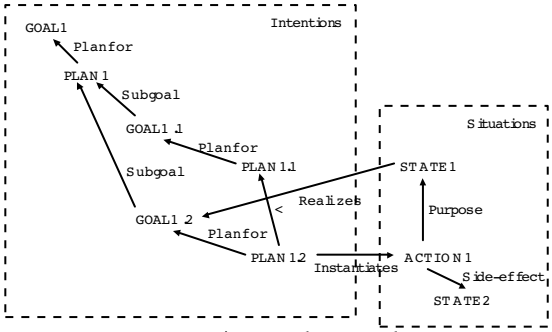

Figure 15. Goal/Plan hierarchy. From [Wilensky02].

Currently Wilensky proposes to use a goal/plan hierarchy (Figure 15), which besides plan recognition could be useful also for planning of system actions [Wilensky02]. He notices that a lot of facts are required allowing the intended system to work: about actions, such as that actions cause certain situations, have preconditions and that one action comprises another, about plan generation, such as that many goals have standard plans (e.g., eating as a plan to resolve hunger), usual preconditions for plans, and also preconditions, that facts have typical and non-typical explanations (e.g., the fact of eating is usually performed to alleviate hunger, not to make food disappear), etc. Finally, if the system becomes as complex as this, serious planners are needed to find the required inferences.
Wilensky noted that real plans may be more complex than the simple plans leading to a goal, for instance goal subsumption (planning for recurring goals, such as needing food or needing transport) may be planned for by obtaining ownership, getting a job, etc. Wilensky also notes that more realistic, longer stories have interwoven plans and goals. He suggests structures accounting for these plans in parallel.

Heuristic approach
Allen and Perrault [Allen80] took a more heuristic approach. Typical rules for plan estimation are "if system believes A has goal of executing ACT, and ACT has an effect E, then system may believe that A has a goal of achieving E", or "if S believes that A has goal to know whether P is true, then S may believe that A has a goal of achieving P". They split beliefs and acts in system- and agent-beliefs and acts, giving heuristic rules like SBAW(ACT) =i⇒SBAW(E) if E is effect of ACT (if system beliefs actor wants an action then system beliefs actor wants the effect of the action). They split the plans in partial plans, with associated tasks that can modify or further specify it. Given the facts, the partial plans can be rated to how probable they are to be the actual user's plan, using information on how well informed the plan is in the given context and how well the plan fits the expectations. They have heuristic rules to determine the likeliness of executing, awaiting and finished actions (Table 6).

Table 6. Heuristic rules determining probability of a rule. From [Allen80]

Description	Factor
Preconditions false	0.5
Effects already true	0.5
Referent (object or relation) identified	1.5
Referent impossible	0.2
Intersection (common goal or action) found between the rule and the expectations	1.5
Inference rule applied	1.25

Demons

Charniak et al. [Charniak86] proposed demons to detect relations between concepts (such as buying and needing money). His demons use rules like "if it is (going to) rain AND person P is outside THEN P will get wet". Once 'rain' is mentioned in the text then this rule becomes available for reasoning. The rule will stay active until a pre-defined number of utterances have gone by. More complex events are handled by 'base routines', for instance when the word 'trade' is encountered a large program handling exchange of goods is triggered. Charniak proposes that demons can trigger other demons of similar structure, for instance 'PiggyBank-Need-Money' can trigger 'Buy-Need-Money'. This is because Piggy Banks are known to contain money. There are some problems with this approach. First, the base routine code and demon code will probably become quite complex. Second, this code will probably get quite some if-then-else constructions, which makes the core knowledge implicit in the code, and very difficult to understand, reason about, extend, keep consistent, etc. Third, probably an unmanageable amount of demons will be triggered when hitting concepts like money or buy, and it is not clear what to do with that.

Chin [Chin88] also uses demons, which he names 'if-detected' demons, but in a different way. He has specific goal-detection rules, represented in a KODIAK network. His HAS-GOAL relation has a KNOW/BELIEVE/HAS-INTENTION aspect as well as the ACTIVE/INACTIVE/ DONE aspect. He distinguishes recurrent and background goals. Recurrent goals are system goals that run constantly, such as being polite or helping the user. Background goals are activated when a plan for a goal is detected. Chin has five situation classes giving rise to new system goals. The first three, themes giving rise to goals, plans giving rise to sub-goals and goal interactions giving rise to meta-goals , are general for agents. The last two, gaps in user knowledge and user misconceptions, give rise to the goal of correcting those problems, are typical for a consultant and may differ for other applications. All these situation classes are detected by demons.

If multiple plans have been found, a meta-plan may be launched to decide which plan has to be carried out. Similarly, a meta-plan may exist for the case when no plans could be found. For example, Figure 16 shows a meta rule for the case that the user wants to know something unethical.

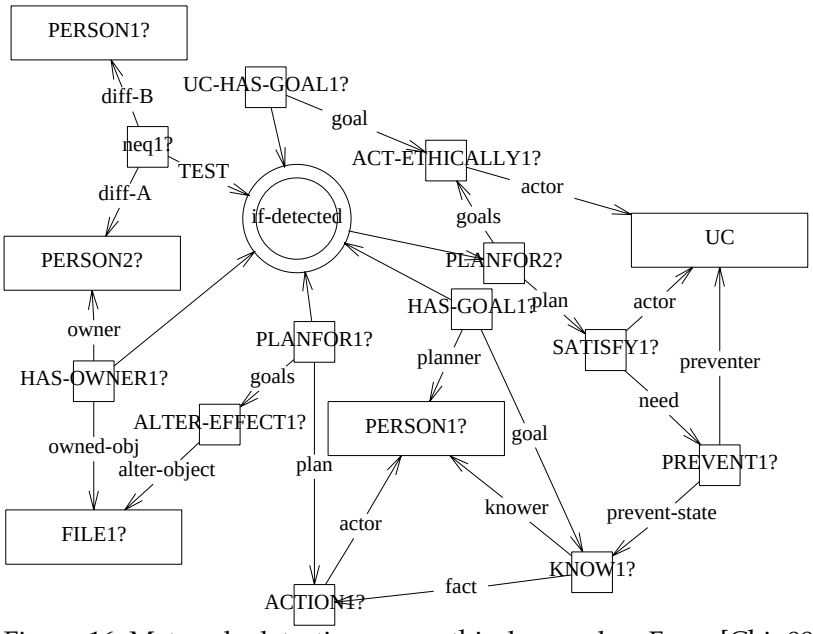

Figure 16. Meta rule detecting an unethical user plan. From [Chin88]. Arrows entering the if-detected demon are preconditions to trigger the demon, the outgoing are the action(s) done when triggered.

Chin's approach also shows minor shortcomings here. In the rule in Figure 16, the 'act-ethically' check is very specific: it detects whether (1) PERSON1 wants to alter someone else's file, and (2) whether UC wants to act ethically, and in that case the suggested plan is to prevent PERSON1 from knowing facts how to do this. This meta-rule is made specifically to counteract another specific rule suggesting the plan to tell PERSON1 how to alter (delete, move) files. It seems that a lot of such rules are needed in order to give substance to the notion of acting ethically. In this case the rule would already be much more general if 'FILE1' would be changed to 'SOMETHING1', but it is clear that the properties of the object hierarchy can have big consequences. Indeed he concludes that the main bottlenecks of this approach are the acquisition of the large amounts of knowledge needed and the question how to best represent it.

Ambiguity in recognised intentions

The plan recognition component may return multiple plausible user plans. Much research has been devoted to resolve ambiguity as soon as possible, generally using heuristics (e.g., [Allen80], [Carberry90]). It is tempting to use probabilistic rules, or weighing factors as in [Allen80], to represent a plausibility level. However, accurate reasoning without such rules as proposed by Carberry [Carberry90] seems preferable, not only to keep the planner happy but also because explicit rules allow explicit indication of which parts of the plan are not clear yet, making incremental tracking and updating of the user's plans possible.

Just waiting until ambiguities are resolved may be undesirable, especially for pro-active systems. On the other hand, asking the user continuously for clarification will diminish the user's trust in the system, so clarification requests have to be used carefully. Beek and Cohen [Beek91] demonstrate how the need to have a single plan can be relaxed. They show how a plan graph (similar to Figure 15, but with

additional preconditions, equality constraints, and temporal constraints) can be used to estimate whether ambiguity matters for the problem at hand. Their approach consists of three steps. First, they determine the possible plans. Next, they check the status of the plans, by annotating these plans with either 'failure of preconditions', 'temporally inconsistent', 'there is a better plan' and 'faultless'. Finally, it is determined whether clarification is needed, and if so a clarification dialog is started. The annotated plan graph can be used to create an efficient clarification dialog.

Building a large common-sense knowledge base

Various attempts are running to collect huge amounts of common-sense knowledge. We will discuss a few of them.

For many sub-domains specialised databases have been built containing more or less common-sense knowledge. For instance Wordnet comprises a huge database of English lexical knowledge, there are yellow pages, phone books, map services, travel planning, currency conversion, chemical substance databases etc. all available online. One can argue whether this is common sense knowledge, because for instance the yellow pages give only information on locations of shops, and no rules to reason about them.

On the other extreme, the world wide web can also be considered as a large common-sense knowledge base, with a lot of rules, hints and suggestions about how to actually achieve things. But unfortunately theses rules are based on natural language, mixed up with all other kinds of information, commercials, stories etc. Also, the credibility of suggestions may be uncertain, making it doubtful whether a rule is appropriate to use. Automatic extraction of information from such natural-language pages is possible in highly restricted areas. For instance Doorenbos uses a cocktail of heuristic search, pattern matching and inductive learning to extract price and product information from non-prepared web pages. It is possible to automatically generate semantic nets that have links between words that are often used together. Such semantic nets are already used, for instance to get an idea about the word context, for instance to support automatic translation, to enhance search results etc. However, searching and extracting rules and checking their reliability from such pages seems much more difficult.

Various attempts have been done to develop more semantically oriented description languages for web page contents, or a 'semantic web' [Gómez-Pérez02]. The usual approach is to describe a mapping from some kind of semantic network onto XML [Lassila99] or HTML [Heflin02]. However it seems doubtful that a significant fraction of web page builders will do the enormous amount of extra work to convert their pages into such a machine-readable form.

Automatically generated networks might be useful to enhance the scope of hand-coded knowledge, just as for instance Wordnet can be imported and used in Cyc. Such networks can probably also be used to support language understanding, for instance Norvig [Norvig87] uses a semantic network for inferring and referent resolution in English texts that probably can be built automatically, at least partially. Schank [Schank80] sketches how a network comparable to the one used by Norvig can be learned from events. He distinguishes 4 levels of memory: event memory holding single events (e.g., last visit to the dentist), generalized events (e.g., going to the dentist), situational memory (e.g., going to a health professional) and

intentional memory (e.g., getting any problem taken care of by a societal organization). Parts of events are abstracted further when overlap with earlier events is detected, and remaining parts that don't fit may drop out of memory.
With Cyc [Lenat95], an ambitious project to capture common sense in predicate logic is currently running. The building of the rulebase is all hand work, and building has been going on for nearly 15 years now. Recently, the project was made open-source [OpenCyc02], in an attempt to speed up the manual construction of the system. Although numerous application areas have been proposed for the Cyc knowledge base, we are not aware of currently available applications using it.
Wilensky's plan and goal inference ideas (see the section on logic plan and goal inference in the next chapter) require a large amount of common-sense knowledge, and again this knowledge has to be entered by hand. Wilensky's approach is sharper formulated than that of Cyc, as he has very clear questions that the knowledge base should be able to answer. This seems important to make clear which knowledge is needed and which is not, and such a restriction is barely needed to restrict the number of rules required. For the Unix Consultant system [Chin88], UCTeacher has been developed [Wilensky86], that can learn about UNIX from the user. This component is script-based, and can only learn new commands that are similar to ones already known. Again, adding knowledge is mainly handwork, although now done by the user.
Pohl et al. [Pohl95] distill application independent rules from literature on deriving prerequisites and presupposition patterns from observed user actions: (1) correct use of terminology implies that the user knows the concepts he mentions (2) incorrect use of terminology implies that the user does not know these concepts (3) requests for explanation of a concept imply that the user does not know that concept (4) request for details imply that the user is familiar with the concept he's asking about, and (5) user gives positive/negative feedback on system output implies that the system should consider increasing or decreasing the plausibility of the underlying system assumption. Unfortunately general, application independent dialog act types do not seem to be very helpful as user model acquisition heuristics since the assumptions that can be inferred from them using dialogue act analysis are likewise general.
In spite of impressive progress in the area of automatic data gathering, we are not aware of a system capable of learning and applying common-sense rules in this way. For the moment it seems that essential relations in the knowledge base have to be designed carefully in order to make the information useful, and the main bulk of the knowledge still has to be hand-coded.

Microtheories
It is very hard, if possible at all, to define universally true rules in a general common-sense knowledge domain. But it is possible to define rules for a specific domain.
Usually, systems have mechanisms to define rules and facts to be restricted to a certain domain. In KODIAK [Wilensky02] such restricted domains are named 'possible worlds'. KODIAK also has a notion of a 'current world' and a hierarchy of possible worlds so that assertions from (potentially multiple) parent worlds are also available. In Cyc [Lenat95], such a restricted domain with rules is called a microtheory, and similar structuring and methods are available as in KODIAK.

Charniak's demons [Charniak86] also allow restricted availability of knowledge, depending on the context mentioned in the text.

Conclusions

Semantic net inference seems a very clean way to represent knowledge and rules, and at the same time allows efficient inference and reference resolution. However the claim that it can do script- and plan-like inference as well seems not supported sufficiently. Logical plan inference seems more powerful, but the problem is that knowledge is more cluttered and more difficult to understand. In the end though, the two systems may be just have the same power, just as the representation itself of semantic nets and predicate rules have the same power.

Main problems are the acquisition of knowledge and the question how to represent the knowledge. Common-sense knowledge bases still have to be hand-coded and knowledge is highly task-specific. That most common-sense knowledge is coded for a specific language complicates reuse of existing knowledge. Coordinated work on this hand-coding, for instance by Cyc, seems the best way to go, but it remains to be seen how good the Cyc language is in practice. Although learning systems have been proposed, at this moment the best to expect from this is semi-automatic construction of knowledge bases with additional hand work.

A lot of story- and discourse understanding systems have been built, but often the evaluation part is neglected. Chin [Chin01] strongly recommends empirical evaluation with users. He suggests to use the number of correctly estimated user plans, properly proposed plans or expert suggestions as a reference. For help- and consultancy systems, subjective user satisfaction, task completion speed and error rates can also be a good test.

When multiple agents are involved, other issues may play, such as unwanted side-effects and the likelihood that the other agent will carry out the request. This topic is complex [Cohen97] and we will not discuss this further here.

4. Architecture

This section discusses two remaining points: multi-modal integration and architectural issues. We first discuss a general point about how to efficiently handle modality integration, and then we discuss some architectures with respect to multi-modality, discourse representation, etc.

Multimodal integration

Multimodal integration can be done at feature-level, which is especially effective with tightly synchronised modes such as voice and lip-reading, and theoretically can be better than semantic-level integration. But such integration is very difficult when the time scales of the modalities differ significantly. Semantic-level integration also allows much easier, uni-modal handling of the basic inputs, for which standard packages are often readily available.

Wu et al. [Wu99] showed that, assuming proper classification and weighing of the semantic primitives from the uni-modal sensors, the results can be close to the theoretical limits. His MTC architecture (Figure 17) has 3 layers, each using a kind of neural nets: (1) the recognisers (2) teams and (3) a judging committee layer. To set up the neural nets properly, he determines optimal weights for various modes. To do this efficiently he uses class-dependent weighing parameters instead of looking at individual events.

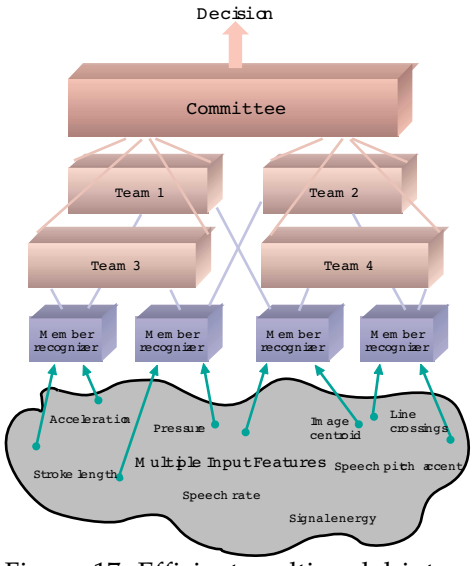

Figure 17. Efficient multimodal integration using proper probabilities. From bottom to top, the recognisers, the teams and the judging committee layers. From [Wu99].

Architectures

In this section we look at a few existing architectures, to see how multimodality is achieved and how knowledge is stored and used. We also discuss conflict resolution, which is a general architecture issue.

Smartkom

The SmartKom project [Wahlster02] is a state-of-the-art example of multimodal integration using semantic-level structures (Figure 18). In the middle we see the 'sprach-analyse' path, consisting of the 'audio-eingabe', 'sprach-erkennung' and 'prosodie-analyse' parts we discussed above. As a parallel path we see the 'gesten-analyse' path. These two paths are then integrated in the 'medien-fusion' component. A little surprising is that the 'mimik-interpretation' is not used in the media fusion component, apparently they use the mimik only for the user's feedback on the system's behaviour. After the media fusion component we see a large bus with 'aktions-planung' as central component, connecting interaction modeling, media fusion, dynamic help, lexicon management, context modeling, discourse modeling, and plan recognition. The idea here is that all these modules can give more or less independent hints about the user's plans, and blackboard techniques instead of a single pipeline are used to integrate those independently working modules. The function modeling is guided by the action planning component as expected. We are not aware of concrete ideas for this project on how to do the intentions erkennung and diskurs modellierung.

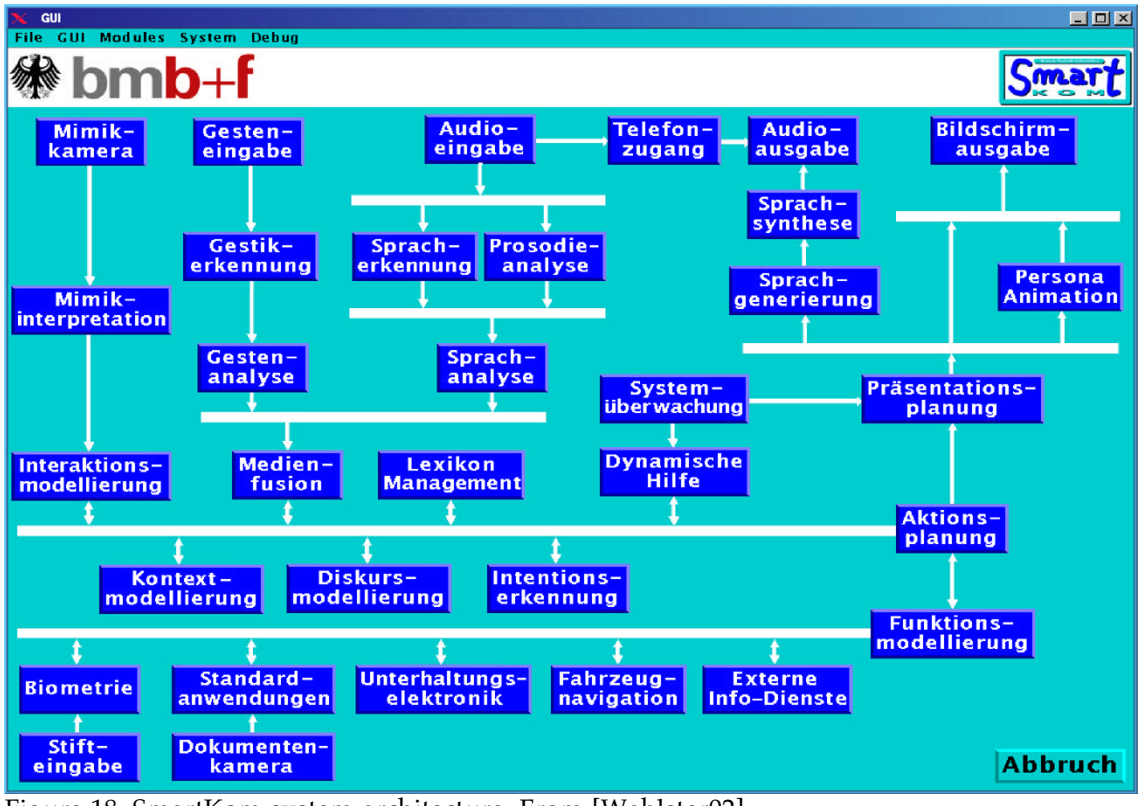

Figure 18. SmartKom system architecture. From [Wahlster02].

TRIPS

The TRIPS architecture (The Rochester Interactive Planning System) shown in Figure 19 [Allen00] proposes efficient, multimodal output generation from semantic structures, independent of the application. The generation keeps track of turn taking in real-time, even while part of the input is still being generated, and can pick suitable modalities to present parts of the input. This architecture did not handle

multimodal input, and the KQML and ACL performatives used in their discourse manager seem not very suited to do so either. An interesting aspect of this architecture is that the discourse manager (DM) is completely independent of the application at hand. The DM coordinates the processes to recognize the user's intentions and computes new obligations (e.g., if asked a question one should respond). It deals with abstract intentions such as introducing new goals, modifying existing plans and requesting background information, but the evaluation and details of these are handled by the behavioral agent and the plan manager.
TRIPS uses a standard speech parser, and therefore will be unable to handle prosody. Prosody can be an important input for plan recognition and dialog acts, and might be important to have in the i-DEA.

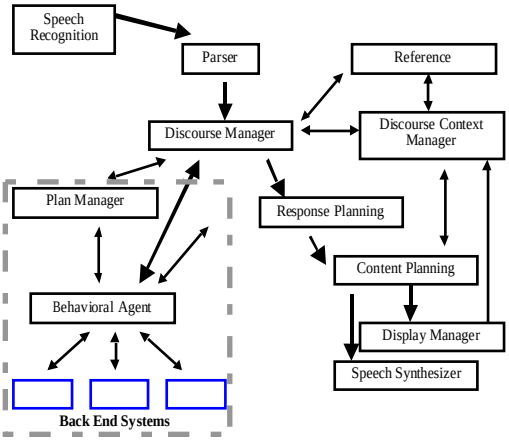

Figure 19. The abstract architecture of TRIPS [Allen00]. The dotted box indicates application-dependent components.

Agent Dialog Framework
McBurney and Parsons [McBurney01] built a three-level hierarchical formalism for agent dialogues called Agent Dialog Framework. At the lowest level are topics which are the subjects of dialogues. The lowest layer can be any language, for instance temporal modalities. At the next level are dialogue types – persuasions, inquiries, etc., and combinations of these, represented using games-theory. At the highest, control level are control dialogues, where agents decide which dialogues to enter. Here the five atomic dialogue types from Walton and Krabbe [Walton95] are used. Additionally, there are dialogue combinations: iteration, sequencing, parallelization (two discussions running simultaneously), embedding (a sub-dialog in a larger dialog), and testing (a dialog aiming at assessing truth-status of a proposition). These mechanisms are pretty detailed in modeling a multi-agent discourse, and the procedures they propose seem usable to steer agent behaviour. If the control dialog level is sufficiently realtime, barge-in initiative could be taken by the system as well. However, the dialogue level seems not powerful enough to handle multimodal integration, maybe this level can be improved for this.

Conflict resolution
In many cases conflicts can arise. These can arise at various levels, from phoneme recognition up to high semantic levels. There can be low-level conflicts, for instance

a connection between modules may break down (e.g., a wireless link fails), modules may fail or send illegal data. Alternatively, there may be time constraints, and modules may have to be 'timed out'. Second, the user's utterances may not be consistent, for instance his actions may give rise to conflicting goals. Third, the user's plans and goals may conflict with system goals and themes, such as the 'good ethics' theme [Chin88]. Finally, if the system has learning abilities, the system's knowledge base may become inconstent (breaking down the soundness of the logic systems).
It seems that many different mechanisms have to be used to resolve conflicts at various levels in a convenient way. The problem is similar to error handling in programs, which also suggests that good conflict resolution is essential for a robust, trustable system. For instance, we will need rules how to handle paradox or conflict, and to learn from mistakes; global processes that keep an eye on the progress and cuts ties and restart system parts where necessary; proper error handling, truth maintenance, etc. It may be good if the application (i-DEA) is notified when a conflict occurs, as keeping control of the progress is important [REF].

Conclusions

Concluding, multimodal integration seems not a big problem once uniform semantic representations have been developed fitting all sensor input.
We are not aware of an architecture that would perfectly fit the i-DEA. We discussed a few that seem most promising, but it is not clear which would be the best to start with, nor whether we can get a copy of the code. For the long-term interaction expected between the user and the i-DEA, a logic plan-based system seems to be the best choice. It is not clear what approach SmartKom will take here. KQML used in STRIPS could be used, but more details are needed to check this. The agent dialog framework seems not to deal with such issues, maybe it could be put just on top of for instance a KODIAK system.

5. Conclusions

There is a vast number of topics touching to the Cactus' research goals. So large in fact that this report is barely able to touch the surface. Many offer interesting potential for recognising and extrapolating of the user's plans and goals.
To implement all functionality from scratch within the Cactus project would take way too much time, while much of that work would not be of significant scientific value. On the other hand essential parts, such as prosody analysis and construction of a semantic net representing the user's input, are still state-of-the-art work not available commercially. If we want speech-based or even multimodal input and a semantic-based system, we will have to get large working parts from other projects, either by cooperating, by getting a licence or by buying parts. Especially the VerbMobil system and its successor SmartKom look interesting. The TRIPS system looks interesting as well, but seems to have limited multimodality possibilities. We may be able to tweak some or simulate some missing components. Concluding, we have to decide carefully where to spend our efforts on and where we can do some tweaks to simulate missing parts of the system.

Appendix A

Natural Language Processing

This appendix gives a sketch of the standard parts of a speech analysis pipeline. Figure 20 shows the first part of the natural language processing (NLP) pipeline. The individual steps are discussed in subsequent sections.

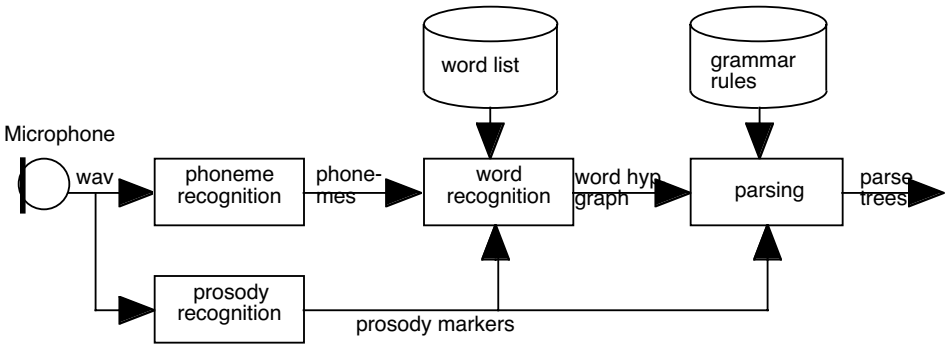

Figure 20. First part of NLP pipeline.

The parse trees (multiple trees, or a single structure somehow accommodating remaining ambiguity at this level) coming out of the pipeline are the input for more 'semantic' processing, which is discussed in the main text.

Phonemes
Phonemes are the smallest sound parts of spoken words, prosody refers to the pitch, emphasis, timing and pauses in the speech. Phonemes have to be glued together and word boundaries have to be detected in order to find the spoken words. Words and word boundaries can not be determined perfectly, giving a word hypothesis graph indicating certainty levels for the best n alternatives, where n usually is below 10 (Figure 21). Setting n higher than 1 enables modules further away in the pipeline to take the best alternatives based on other criteria than acoustic information.

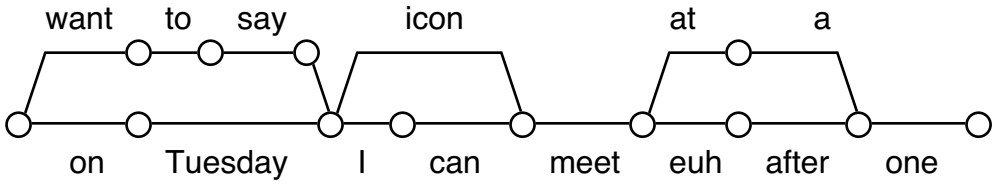

Figure 21. Word hypothesis graph for a phoneme string something like "ohntyousay-Icuhnmeetuh...ahftehwahn". Various paths indicate different possible interpretations of the phonemes in the speech. The words in the graph all get goodness-of-fit number.

Parsing
Usually a context free grammar is used to define how words build sentences (Figure 22). Natural language is not strictly context free, but this is solved by having the

grammar accept more than strictly grammatically correct. With spoken language sentences are often not formed grammatically correct anyway, and therefore too strict parsing has limited practical value in spoken language interfaces. Of course this means that non-grammatical sentences can be parsed as well, including nonsense sentences, and these sentences will have to be caught later in the system as required.

```
Sentence->NounPhrase VerbPhrase
NounPhrase -> Determiner Noun
VerbPhrase -> Verb
VerbPhrase -> Verb NounPhrase
Noun -> "car"
Noun -> "ape"
Determiner -> "the"
Verb -> "drives"
```

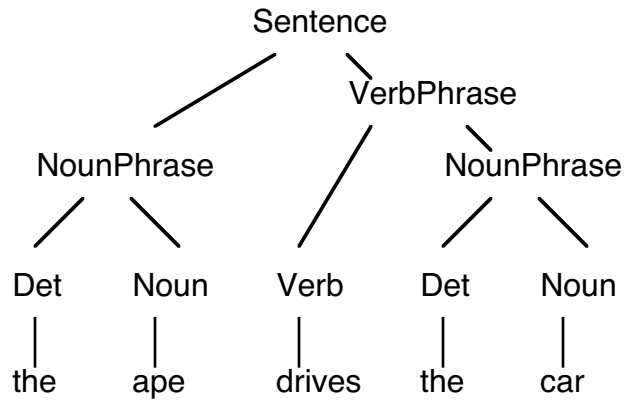

Figure 22. Simple context free grammar of English (left) and parse tree of "the ape drives the car" (right). In realistic systems, the terminals (ape, car, the, and drives in this example) are not in the rule system but in separate dictionaries.

For spoken language, handling of self-corrections (e.g., "on Tuesday I cannot • **no I can** meet after one") is important: for instance in the Verbmobil scenarios nearly 21% of all turns contain at least one self correction [Spilker00]. In Verbmobil such corrections are handled as early as possible, although it is noted that self corrections can occur at virtually every level in the system – from phoneme level up to the semantic level. Standard spoken language parsers often have serious problems with spontaneous speech, and prosody and stochastic models may be needed at this level already to handle self-corrections [Wahlster00].

Multimodal integration

It is possible to do a multimodal integration step directly after the parsing. This would be a feature-level integration. Certain keywords like 'here' and 'this' from the language can be integrated relatively straightforward with pen input, probably even at an earlier stage in the pipeline. But other, more complex drawings, might be interpretable only with domain- and discourse knowledge. For instance, an downward-pointing arrow could also be a sign for a tree or a transistor, depending on the context. Therefore we discuss multimodal integration further in the main text, where multimodal integration is put in a semantic-level perspective.

References

[Alexandersson00] Alexandersson, P, Poller, P., Kipp, M., & Engel, R. (2000). Multilingual summary generation in a speech-to-speech translation system for multilingual dialogues. Proceedings of INLG-2000. Available Internet: http://verbmobil.dfki.de/dialog.

[Alexandersson00b] Alexandersson, J., Engel, R., Kipp, M., Koch, S., K¸ssner, U., Reithinger, N. & Stede, M. (2000). Modeling Negotiation Dialogs. In W. Wahlster (Ed.), Verbmobil: Foundations of Speech-to-Speech Translation (Berlin: Springer), 441-451.

[Ali93] Ali, S. S., & Shapiro, S. C. (1993). Natural Language Processing using a Propositional Semantic Network with Structured Variables. Available Internet: http://www.cs.buffalo.edu/tech-reports/93-16.ps.Z.

[Allen00] Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L. and Stent, A. (2000). Towards a Generic Dialogue Shell, Natural Language Engineering, 6 (3), 1-16. Draft Available Internet: http://www.cs.rochester.edu/users/faculty/james.

[Allen80] Allen, J. F., & Perrault, C. R. (1980). Analyzing intention in utterances. Artificial Intelligence, 15, 143-178. Reprinted in B. J. Grosz, K. S. Jones, & B. L. Webber (Eds.), Readings in natural language processing (1986). Morgan Kaufmann Publishers, Inc., Los Altos, CA, 441-458.

[Barker99] Barker, K. (1999). Course notes for CSI5386. University of Ottawa, CA. Available Internet: http://www.site.uottawa.ca/~kbarker/teach/5386/notes.html.

[Beek91] Beek, P. van, & Cohen, R. (1991). Resolving plan ambiguity for cooperative response generation. In: Proceedings of the 12th International Joint Conference on Artificial Intellligence (IJCAI'91, Sydney, Australia), 938-944. Available Internet: http://ai.uwaterloo.ca/~vanbeek/publications/publications.html.

[Bigus01] Bigus, J. P., & Bigus, J. (2001). Constructing Intelligent Agents Using Java. New York: John Wiley & Sons. ISBN 0-471-39601-X.

[Boutilier02] Boutilier, C., Kwiatkowska, M., Haverkort, B., & Vardi, M. Y. (2002). Probabilistic Methods in Verification and Planning. Seminar topics description (11 - 16 May, 2003, Schloss Dagstuhl, Germany). Available Internet: http://www.cs.bham.ac.uk/~mzk/Dagstuhl0320.

[Calistri-Yeh91] Calistri-Yeh, R. J. (1991). Book reviews: Plan recognition in natural language. Available Internet: http://acl.ldc.upenn.edu/J/J91/J91-3006.pdf.

[Carberry01] Carberry, S. (2001). Techniques for Plan Recognition. User Modeling and User-Adapted Interaction, 11, 31-48. Available Internet: http://umuai.informatik.uni-essen.de/anniversary.html.

[Carberry90] Carberry, S. (1990). Plan recognition in natural language dialogue. ACL-MIT Press series in natural language processing. Bradford Books, MIT Press, Cambridge, MA. ISBN 0-262-03167-1.

[Charniak86] Charniak, E. (1986). Jack and Janet in search of a theory of knowledge. In B. J. Grosz, K. S. Jones, & B. L. Webber (Eds.), Readings in natural language processing. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1986.

[Chin01] Chin, D. N. (2001). Empirical evaluation of user models and user-adapted systems. User modeling and user-adapted interaction, 11, 181-194. Available Internet: http://umuai.informatik.uni-essen.de/anniversary.html.

[Chin88] Chin, D. (1988). Intelligent Agents as a Basis for Natural Language Interfaces. Ph.D. thesis, University of California, Berkeley. Also available as Technical Report UCB/CSD 88/396, Computer Science Division, University of California, Berkeley. Available Internet: ftp://ftp.ics.Hawaii.Edu/pub/tr/ics-tr-88-11.ps.Z.

[Cisco02] Cisco (2002). Cisco Personal Assistant. Available Internet: http://www.cisco.com/warp/public/cc/pd/unco/persasst and http://www.cisco.com/warp/public/cc/pd/unco/persasst/prodlit/cpasc_ds.pdf.

[Clark00] Clark, P., & Porter, B. (2000). KM - The Knowledge Machine 1.4.0: KM's Situation Mechanism. Internal Report, Department of Computer Science, University of Texas at Austin, USA. Available Internet: http://www.cs.utexas.edu/users/mfkb/RKF/km.html.

[Cohen97] Cohen, P. (1997). Dialogue Modeling. part of Chapter 6 in Cole, R. A. et al. (Eds.), Survey of the state of the art in human language theory. Giardini Editori e Stampatori, Pisa, Italy. ISBN 88-427-0018-5. Available Internet: http://www.lt-world.org/HLT_Survey/ltw-chapter6-all.pdf.

[Conita01] Conita (2001). Conita's Personal Virtual Assistant. Available Internet: http://www.conita.com/pdfs/conitafactsheet.pdf.[Errico96] Errico, B. (1996). Student modelling in the situation calculus. European Conf. on AI in Education (Lisbon, Portugal, September 30- October 2), ??. Available Internet: http://cblslca.leeds.ac.uk/~euroaied/papers/Errico.

[Executive02] Executive Services International (2002). Personal Assistant Services. Available Internet: http://virtual-offices-online.com/secretary.htm. [FIPA01] FIPA (2001). FIPA Communicative Act Library Specification. Foundation for Intelligent Physical Agents, Concord, CA. Available Internet: http://www.fipa.org/specs/fipa00037.

[FIPA98] FIPA (1998). Human-Agent Interaction (obsolete). Foundation for Intelligent Physical Agents, Concord, CA. Available Internet: http://www.fipa.org/specs/fipa00004.

[Finin97] Finin, T., Labrou, Y., & May, J. (1997). KQML as an agent communication language. In Bradshaw, J. (Ed.), Software Agents. MIT Press, Cambridge.

[Galton99] Galton, A. (1999). Temporal Logic. Available internet: http://plato.stanford.edu/entries/logic-temporal.

[Garson01] Garson, J. W. (2001). Modal Logic. Available Internet: http://plato.stanford.edu/entries/logic-modal.

[Genesereth98] Genesereth, M. R. (1998). Knowledge Interchange Format. Draft proposed American National Standard NCITS.T2/98-004. Available Internet: http://logic.stanford.edu/kif/dpans.html.

[Gómez-Pérez02] Gómez-Pérez, A., & Corcho, O. (2002). Ontology Languages for the Semantic Web. IEEE Intelligent Systems, January/February, 54-60.

[Heflin02] Heflin, J., Hendler, J., Luke, S., Gasarch, C., Zhendong, Q., Spector, L., & Rager, D. (2002). SHOE: Simple HTML Ontology Extensions. Available Internet: http://www.cs.umd.edu/projects/plus/SHOE.

[Hess91] Hess, M. (1991). Recent Developments in Discourse Representation Theory. Communication with Men and Machines, ??. Available Internet: http://www.ifi.unizh.ch/groups/CL/hess/drt.html.

[Hirst97] Hirst, G., DiMarco, C., Hovy, E., & Parsons, K. (1997). Authoring and Generating Health-Education Documents that are Tailored to the Needs of the Individual Patient. Proc. 6th Int. Conf. on User Modeling (UM97, Sardinia, Italy), 107-118.

[Irribarren96] Irribarren, R., & Vargo, M. (1996). System and Method for Integrating Voice, Facsimile and Electronic Mail Data through a Personal Computer. US Patent 5,530,740. Available Internet: http://www.uspto.gov/patft/index.html.

[Kardol99] Kardol. J. J. (1999). Bring Mabel Back. Undisclosed Technical Report RA-99-31287, KPN Research, 1998.

[Kobsa95] Kobsa, A. and W. Pohl (1995): The User Modeling Shell System BGP-MS. User Modeling and User-Adapted Interaction 4(2), 59-106. Available Internet: http://www.fit.fraunhofer.de/~kobsa/publi.html.

[Koch00] Koch, S., K¸ssner, U., & Stede, M. (2000). Contextual Disambiguation. In W. Wahlster (Ed.), Verbmobil: Foundations of Speech-to-Speech Translation (Berlin: Springer), 466-477.

[Kowalski86] Kowalski, R. A. & Sergot, M. J. (1986), A Logic-Based Calculus of Events. New Generation Computing, 4, 67-95.

[Labrou97] Labrou, Y., & Finin, T. (1997). A proposal for a new kqml specification. Technical Report TR-CS-97-03, University of Maryland, Baltimore County. Available Internet: http://www.csee.umbc.edu/~jklabrou/publications.html.

[Lagendijk02] Lagendijk, R. L., & Vliet, P. J. van (2002). Cactus Impulse Research Project (Context Aware Communication, Terminal, and User). Project proposal for the Freeband Impulse

Research Program in Telecommunications, May. Delft University of Technology. See also http://www.cactus.tudelft.nl/mainpage.htm.

[Lassila99] Lassila, O., & Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. W3C proposed recommendation PR-rdf-syntax-19990105. Available Internet: http://www.w3.org/TR/PR-rdf-syntax.

[Lenat95] Lenat, D. B. (1995). CYC: a large-scale investment in knowledge. Comm ACM, 38 (11), 33-38.

[MDCS00] MDCS (2000). Job Specification: Legal Secretary. Detroit, MI: Michigan Department of Civil Service. Available internet: http://www.state.mi.us/mdcs/employ/specifications2000/ legalsecretary.pdf.

[McBurney01] McBurney, P., & Parsons, S. (2001). Games that agents play: A formal framework for dialogues between autonomous agents. Journal of Logic, Language and Information, 11 (3): 315-334. Special Issue on Logic and

[Minsky75] Minsky, M. (1975). A Framework For Representing Knowledge. In P. H. Winston (Ed.), The Psychology of Computer Vision. New York: McGraw-Hill, 211-277. Also available as MIT-AI Laboratory Memo 306 (June 1974). Available Internet: http://www.ai.mit.edu/people/minsky/papers/Frames/frames.html.

[Mueller02] Mueller, E. T. (2002). Story understanding resources. Available Internet: http://xenia.media.mit.edu/~mueller/storyund/storyres.html.

[Norvig87] Norvig, P. (1987). A unified theory of inference for text understanding (Technical Report CSD-87-339). Berkeley, CA: Computer Science Division, University of California, Berkeley. Available FTP: ftp://sunsite.berkeley.edu/pub/techreps/CSD-87-339.html.

[OpenCyc02] OpenCyc (2002). OpenCyc: the project. Available internet: http://www.opencyc.org.

[Paris89] Paris, C. (1989). The Use of Explicit User Models in a Generation System for Tailoring Answers to the Userπs Level of Expertise. In A. Kobsa & W. Wahlster (Eds.), User Models in Dialog Systems, 133≠162. Springer, Berlin, Heidelberg.

[Pohl95] Pohl, W., Kobsa, A., & Kutter, O. (1995). User model acquisition heuristics based on dialogue acts. International Workshop on the Design of Cooperative Systems, (Antibes-Juan-les-Pins, France), 471-486. Available Internet: fit.gmd.de/~kobsa/papers/1995-COOP95-kobsa.ps.

[Rich00] Rich, C., Sidner, C. L., & Lesh, N. (2000). COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction. AI Magazine, Special issue on Intelligent User Interfaces. Also available as Technical Report TR2000-38, Mitsubishi Electric Research Laboratories. Available Internet: http://www.merl.com/projects/collagen.

[Rich79] Rich, E. (1979). User Modeling via Stereotypes. Cognitive Science, 3, 329≠354.

[Schank72] Schank, R. C. (1972). Conceptual dependency: a theory of natural language understanding. Cognitive Psychology, 3, 552-631.

[Schank80] Schank, R. C. (1980). Language and Memory. Cognitive Science, 4 (3), 243-284. Reprinted in B. J. Grosz, K. S. Jones, & B. L. Webber (Eds.), Readings in natural language processing (1986). Morgan Kaufmann Publishers, Inc., Los Altos, CA, 171-192.

[Scott97] Scott, D., & Kamp, H. (1997). Discourse Modeling. part of Chapter 6 in Cole, R. A. et al. (Eds.), Survey of the state of the art in human language theory. Giardini Editori e Stampatori, Pisa, Italy. ISBN 88-427-0018-5. Available Internet: http://www.lt-world.org/HLT_Survey/ltw-chapter6-all.pdf.

[Shapiro02] Shapiro, S. C. (2002). Knowledge Representation. To appear in Eneyclopedia of Cognitive Science, Macmillan Publishers Ltd. Available Internet: http://www.cse.Buffalo.EDU/pub/WWW/faculty/shapiro/Papers.

[Spilker00] Spilker, J., Klarner, M., & Gˆrz, G. (2000). Processing Self-Corrections in a Speech-to-Speech system. In W. Wahlster (Ed.), Verbmobil: Foundations of Speech-to-Speech Translation (Berlin: Springer), 131-140.

[Stock93] Stock, O. et al. (1993). Alfresco enjoying the combination of natural language processing and hypermedia for information exploration. In Mark T. Maybury, editor, Intelligent Multimedia Interfaces, pages 197--224. 1993.

[Trella00] Trella, M., Conejo, R., & Guzm·n, E. (2000).

[Ueda98] Ueda, T., & Yasuko, M. (1998). Electronic Secretary System with Animated Secretary Character. US Patent 5,761,644 (Jun 2, 1998). Available Internet: http://www.uspto.gov/patft/index.html.

[W3C02] W3C (2002). Voice Extensible Markup Language (VoiceXML): W3C working draft, 24 april. World Wide Web Consortium. Available Internet: http://www.w3.org/TR/voicexml20.

[Wahlster00] Wahlster, W. (2000), Verbmobil: Foundations of Speech-to-Speech translation. Berlin: Springer. ISBN 3-540-67783-6.

[Wahlster02] Wahlster, W. (2002). Multimodale Interaktion und Interface Agenten: Trends f¸r Morgen und ‹bermorgen. Keynote speech, UseWare'02 (Darmstadt, June 12). Available Internet: http://smartkom.dfki.de/eng/start_en.html.

[Walton95] Walton, D. N., & Krabbe, E. C. W. (1995). Commitment in dialogue: Basic concepts of interpersonal reasoning. Albany, NY, USA: SUNY press.

[Ward01] Ward, A., & Addlesee, M. (2001). RFID Tagging People. Research project at AT&T Cambridge. Available Internet: http://www.uk.research.att.com/location/rfidtagging.html.

[Webley02] Webley Systems (2002). Unified Communications. Available Internet: http://www.webley.com/unified_communications.html.

[Wiemer-Hastings02] Wiemer-Hastings, P. (2002). A brief introduction to Discourse Representation Theory. Part of the course 'Survey of Cognitive Science'. Available Internet: http://reed.cs.depaul.edu/peterwh/cogsci/#object.

[Wilensky02] Wilensky, R. (2002). An AI Approach to NLP. Lecture notes for CS288, Computer Science Sivision, University of California, Berkeley. Available Internet: http://www.cs.berkeley.edu/~wilensky.

[Wilensky02b] Wilensky, R. (2002). Documentation for the Kodiak Representation Language.Lecture notes for CS288, Computer Science Sivision, University of California, Berkeley. http://www.cs.berkeley.edu/~wilensky/c288/kodiak.htm.

[Wilensky82] Wilensky, R. (1982). Points: A theory of the structure of stories in memory. In W. Lehnert & M. Rengle (Eds), Strategies for Natural-Language Processing, 345-374. Hillsdale, ??: Lawrence Erlbaum Associates. Reprinted in B. J. Grosz, K. S. Jones, & B. L. Webber (Eds.), Readings in natural language processing, 459-473. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1986.

[Wilensky83] Wilensky, R. (1983). Planning and Understanding. Addison-Wesley.

[Wilensky86] Wilensky, R., Mayfield, J., Albert, A., Chin, D. N., Cox, C., Luria, M., Martin, J., & Wu, D. (1986). UC ã A Progress Report. Computer Science Division, University of California, Berkeley, Report No. UCB/CSD 87/303.

[Wu99] Wu, L., Oviatt, S., & Cohen, P. (1999). Multimodal integration: A statistical view. IEEE Transactions on Multimedia 1, 4, 334≠342.

[YPCA02] YPCA (2002). Eileen: Your Personal Call Assistant. YPCA, Leidschendam, the Netherlands. Available Internet: http://www.ypca.nl.

[Yahoo02] Yahoo (2002). My Yahoo!. Available Internet: my.yahoo.com.

[Zukerman01] Zukerman, I., & Litman, D. (2001). Natural Language Processing and User Modeling: Synergies and Limitations. User modeling and user-adapted interaction, 11, 129-158. Available Internet: http://umuai.informatik.uni-essen.de/anniversary.html.