TR-I-189

A Design for a Disambiguation-Based Dialog Understanding System

曖昧さのない対話理解システムの設計

John K. Myers

真龍主·星音

November 14, 1990

Abstract

This report discussed the current progress in the author's research in dialog understanding, and plans for the future. A design for a comprehensive disambiguation-based understanding system is presented. First, a report of the status of the NP plan inference system is given, along with what it can and cannot do. The current version of the FS-LF system, which translates feature-structures into logical forms, is presented. Current research on the automatic extraction of illocutionary force from semantic utterances is detailed. Next, the report presents preliminary results from new research on scheduling parallel processes. After this, an analysis of the understanding task shows that *disambiguation* problems, followed by prediction and prosody problems, are the most common and significant problems in dialog interpretation. Accordingly, it is most important for ATR to build an understanding system based on disambiguation and prediction. ..The design of a next-generation understanding system that meets these requirements is presented. The system will perform disambiguation and prediction by being based partially on trajectories, causal evidential reasoning, simulation of intentional agents, and reasoning with uncertain non-deterministic actions. The report next details the research that has been performed for this design, in trajectories, evidential reasoning, intention theory, and simulating intentional agents. Finally, a brief summary of the research on Uncertain Non-Deterministic Action (UNDA) theory is presented, along with some applications for ATR. Appendicies contain examples of results.

A Design for a Disambiguation-Based Dialog Understanding System

John K. Myers

ATR Interpreting Telephony Research Laboratories
Sanpeidani, Inuidani, Seika-cho, Soraku-gun, Kyoto 619-02, Japan
myers@atr-la.atr.co.jp

Abstract

This report discusses the current progress in the author's research in dialog understanding, and plans for the future. A design for a comprehensive disambiguation-based understanding system is presented. When completed, the system should solve about 75% of ATR's outstanding translation problems. The many components of the design require research in many different areas. Some of this research, such as the ATMS system, has already been completed. Preliminary research in other areas is presented here for the first time.

The paper starts with reports on the current status of previous lines of research. A report on the status of the NP plan inference system is given, along with an analysis of its capabilities and weaknesses. The current version of the FS-LF system, which translates feature-structures into logical forms, is presented. Current research on the automatic extraction of illocutionary force from semantic utterances is detailed. Next, the report presents preliminary results from new research on scheduling parallel processes, which will be needed in the future. After this, an analysis of the understanding task is presented. The analysis forms a rationale for the work to follow. This analysis reveals that *disambiguation* problems, followed by prediction and prosody problems, are the most common and significant problems in dialog interpretation. Accordingly, it is most important for ATR to build an understanding system based on disambiguation and prediction.

The design of a next-generation understanding system that meets these requirements is presented. The system will perform disambiguation and prediction by being based partially on trajectories, causal evidential reasoning, simulation of intentional agents, and reasoning with uncertain non-deterministic actions. The concept of trajectories in conversation feature space, and how they can be used for prediction, is discussed. Next, the report details research in evidential reasoning: how language understanding can be seen to be an evidential reasoning problem, and how Pearl's Causal Evidential Reasoning algorithms solve many important probabilistic-belief problems. After this, research in intentions and simulating intentional agents is discussed. This will also be required for good disambiguation. Finally, a summary of the research on Uncertain Non-Deterministic Action (UNDA) theory, along with the associated MU uncertainty calculus, is presented. This theory allows planning and decision-making with uncertain actions and uncertain quantities. Some immediately obvious applications of UNDA in ATR are discussed, including scheduling processes when the process durations are uncertain. It is expected that UNDA will have many significant applications.

All of these technologies, when assembled, will contribute towards forming a capable understanding system that is based on solving the concrete needs of ATR.

The paper concludes with some appendicies containing results, including most notably a sketch of how a causal evidential reasoning system can handle disambiguation of "unagi-da-bun" type utterances.

Contents

List of Tables

List of Figures

1 Introduction

It is the responsibility of the dialog understanding section (1) to understand and explain, for concrete reasons, all details that the rest of the system is incapable of understanding or translating properly, and (2) to understand as much of the conversation to be translated as possible, for abstract reasons.

This means that dialog understanding research should proceed broadly along two fronts. First, it is important to understand exactly where the concrete current problems are, and to place emphasis on the special-case solutions to these particular problems. Second, it is important to build a general-purpose dialog understanding system that is as powerful as possible. The system must be well-grounded in strong theory. It must be capable of understanding any new conversation–not just ones that it has been trained on. Only by using an extremely strong, non-fragile, theoretically-proven understanding system will ATR be able to realize its goal of interpreting general conversations automatically by telephone.

The research discussed in this report is an attempt to contribute in these directions. The first parts of the report are concerned with concrete, near-term problems in building systems that help with understanding. The second half of the report is concerned with research in basic theory, and a design for a next-generation understanding system. The design, based on disambiguation, is intended to solve most of ATR's known outstanding near-term problems in dialog understanding, if possible.

2 The NP 3.3 Plan Inference System

2.1 Progress in NP

Version 1.0 used logical forms. It was reported on in [Mye89a].

Version 2.0 was a complete rewrite that converted the NP system from using logical forms to the use of feature structures. Instead of using an inference engine, the system used the rewriting facilities of RWS along with a plan-schema pre-interpreter, a set of *instruction rules*, and an instruction post-interpreter. The pre-interpreter built up a set of instructions and put them in the single consequent of an RWS rule. When the rule executed, the instructions were instantiated and returned. The post-interpreter interpreted these instructions and built a corresponding ATMS network. Although this method worked, it was clumsy. Instantiation was performed bottom-up, which had the advantage that information could be discarded when going from the low levels of the hierarchy up to the high levels. However, instantiation was performed with a set of instructions–recognizing the action from the decompositions was performed separately from implying the effects from the action. This meant that the action description had to contain enough information to instantiate the effects, a disadvantage. In addition, because the decompositions were necessarily segmented into separate rewriting rules, two different decompositions to the same action might unnecessarily reinstantiate the identical implications network, resulting in duplicate implications.

Version 3.0 was another rewrite that introduced the NFL feature-structure-based inference engine, and phased out most of the use of the RWS rewriting system. Instead of the system using all of the RWS used for recognizing patterns, rewriting the consequents, and returning the results as an instruction rule to be interpreted, only part of the RWS was used for just recognizing patterns for NFL. The NFL inference engine instantiated actions bottom-up. Since all of the decompositions were present in the rule, the information required for the effects could be derived from all of the decompositions together, and it was no longer necessary to include deriving information in the action description. The system was invoked in a single pass using a complex series of six arguments. Multiple alternatives were not explicitly supported. Three levels of verbosity were added to the graphics display: small, medium, and large.

Version 3.1 cleaned up the user interface by introducing the "-utt" user commands. The system was invoked incrementally, in an interactive fashion, reflecting a more realistic method of use. In addition, the "alternative-utt" command was introduced to explicitly support possible alternative inputs. The graphics was converted to bold-outline actual assertions, instead of reversing them as white-on-black.

Version 3.2 cleaned up some other minor items. Assumptions can now accept probabilities, although they are not used by the system.

Version 3.3 saw NP installed as a Lisp System in the Symbolics network. The NP system is now available for anyone to use. It can be invoked with the Load System NP command. The user interface has been cleaned up some more, so that it is easier to use and understand.

2.2 Capabilities of the NP system

This section gives an overview of the main capabilities of the NP inference system. A full explanation can be found in [Mye89c].

2.2.1 Works Directly With Feature Structures.

The NP system represents and reasons with feature structures directly. There is no need to convert the feature structures into an internal representation, and then convert them back again. NP is the first inference system known to work with feature structures.[1]

2.2.2 Works With Plan Schemata.

NP works with action plan schemata having preconditions, decompositions, and effects. The plan schemata are more powerful than antecedent-consequent rewriting rules, but can be used as rewriting rules if required.

2.2.3 Performs Plan Inference.

The system performs bottom-up plan recognition, top-down plan prediction, and plan inference.

2.2.4 Fast ATMS-based Hypothetical Reasoning.

Reasoning, especially reasoning with long, complex feature structures, takes time due to pattern matching and variable instantiation costs. In order to speed up computations, the system can perform hypothetical reasoning ahead of time, "off-line", as long as the system can guess that the concepts used for reasoning *might* be used in an actual conversation. Then, when the actual conversation is processed, hypothetical concepts are activated into

[1]This capability was proposed by Mr. Hitoshi Iida.

accomplished, the system responds quickly. Results indicate time savings are around 10:1 when hypothetical reasoning can be done ahead of time.

2.2.5 Works With Contextual Information.

The NP system is built on top of the NFL "expert-system"-style inference engine. An inference engine accepts a number of assertions and uses rules to instantiate more assertions. All of the assertions are put together into one big "soup", and used for random access. There is no need to have the assertions be ordered, they are stored in an unordered set. Some feature-structure systems store assertions in a list; this requires the system to scan the list every time a rule is fired. The time required to fire a rule thus increases at $O(\frac{1}{2}n)$, where n is the number of assertions. However, in a random-access system such as NFL, this cost does not occur; the time is constant ($O(1)$) with the number of assertions.

This also makes sense for an understanding system that makes inferences and comes to conclusions based on known facts. A factual concept, such as `(name-of person Suzuki)` or `(has person form)`, has no inherent order relative to other concepts. The system should be able to use any IF-THEN rule without worrying about how the facts are stored.

Because the system can use rules with *multiple* antecedents (IF X AND Y AND Z THEN U AND V), it is able to write rules that depend on context. This is more powerful than a rewriting system that can only take a *single* feature structure and rewrite it into another feature structure.

2.2.6 Works With Possible and Hypothetical Inputs.

The NP system uses an original 5-valued logic, which enables it to work with possible and hypothetical inputs and outputs. This is more powerful than systems that can only accept concrete input.

2.2.7 Multiple Conflicting Inputs.

The NP system is able to accept and represent the results from an input that has multiple mutually-inconsistent possible values. For instance, say that the Speech Recognition and Parsing modules yield five different parses for the same utterance. It is impossible to represent these as five different inputs to the system, because they could interact; the system must represent this as one input that takes on one of five possible conflicting values. The NP system is able to accept such inputs and reason with them, using the ATMS to split the input values into different mutually-exclusive possible worlds. NP is the first plan-inference system to offer this capability.

3 Main Problems of the NP 3.3 Plan Inference System

3.1 Deterministic Plans Without Conditionals

The current plan inference system represents and works with plans with deterministic outcomes. In this formalism, when an action occurs, its effects must necessarily occur.

Action effects are used as the preconditions to other actions, to represent coherent behavior. This is the only way that the system can attach actions together, to understand what is going on. For example, to represent requests, the system forms the following sequence:

```
ACTION #1:                    The questioner requests action X
EFFECT of ACTION #1:          The office wants to perform action X
PRECONDITION of ACTION #2:    The office wants to perform action X  (the same thing)
ACTION #2:                    The office performs action X
```

Although this sequence is actually much too short,[2] it is acceptable for our purposes. For instance, it can be used to represent the following sequence:

```
The questioner asks the office to send a form
The office wants to send a form
The office sends a form
```

and, also,

```
The questioner asks the office to provide the questioner with some information
The office wants to provide the questioner with some information
The office provides the questioner with some information
```

It is obvious that the middle step is needed. The office will not spontaneously send the questioner a form; it has to *want* to send the form before it will act. Similarly, the office does not spontaneously *want* to send the form–the office is *caused* to want to send the form, as a result of the request of the questioner.

However, this representation does not allow for conditionals, nor nondeterministic outcomes. Using the same sequence, the system confidently predicts:

```
The questioner requests the office to refund the fee
The office wants to refund the fee
The office refunds the fee
```

There are no other alternatives. The plan *has* to happen, according to the representation of the system. However, of course, it doesn't–the office refuses to refund the fee. The system is incapable of representing this sequence as it actually occurs.

The reason for this is that the current system can only represent actions with one, deterministic, set of outcome effects. It is impossible to represent actions that have more than one, nondeterministic, set of outcome effects. But, this is exactly what is required.

[2] A more realistic intentional action sequence would be the following: `The office hears the request; the office understands the request; the office likes to cooperate with the questioner; the office likes to perform anything the questioner requests; the office likes to perform the specific request of the questioner; the office wants to perform the questioner's request; the office considers whether or not to perform the questioner's request; the office does not have any conflicting wants, morals, or intentions; the office decides to perform the questioner's request; the office intends to perform the questioner's request; the office is ready to perform the questioner's request; the office can perform the questioner's request; the office is not doing anything else; the office chooses to perform the questioner's request; the office performs the requested action.` It is still an open research question whether all of this is useful and is needed, or whether it is too much and is inconvenient.

The solution to this problem has been the extensive researching and development of a system that can represent, and reason with, nondeterministic actions. Not only can simple nondeterminism be represented, but probabilities can be attached to each of the outcome sets to indicate which outcome is more likely. The results of this research are discussed in slightly more detail in Section 12, and in a forthcoming paper [Mye90e].

3.2 Deductive Reasoning Cannot Cope with Unobservable Preconditions

The current system uses a strict formalized logic, based on Goldman's "generation" concept[Gol70][WE90] to do plan recognition, which says that the preconditions of an action, plus the decompositions of an action, *necessarily entail* the action's performance. This model is useful for recognizing "by definition" plan actions: if the action "by definition" consists of its decompositions, then the action must be taking place. For example, if one person asks a question, and another person provides information that answers that question, then "by definition" the other person is answering that question. Those two states together necessarily entail the action; it *must* happen. This logic implements a form of *deductive reasoning.*

This type of logic is very useful for representing many types of problems. However, it is too strict to be useful in some cases. It breaks down when one of the preconditions to an action is *unobservable.* "Unobservable" means that, in principle, a state cannot be observed by an outside agent (in this case, the computer). This would not be a problem if the computer did not have to deal with unobservable states very often, or if it could solve for those states. However, when understanding a conversation, the *mental attitudes* of the conversation participants are extremely important. For instance, if the office makes a statement, how can the computer tell whether it is a simple informing act, or whether it is an advice-giving act? The difference between these is that in the second case, the office *cares* about the questioner and *wants* to give advice. But these are mental attitutes "inside the speaker's head"—the computer cannot observe these, and has no way of knowing, ahead of time, whether these states are true or not. Therefore, the current system cannot recognize advice-giving.

Another example of this problem is seen in understanding spontaneous, unsolicited information. For instance, when in conversation number three the office says "Gengo-gaku ya shinri-gaku o senkou-suru kata nimo sanka-shite-itadaku yotei desu." ("We are also expecting linguistics and psychologists as participants."), the questioner did not ask to be told any information about the participants. But, as was demonstrated in the previous section 3.1, it is necessary to *want* to inform someone about something before an informing act can occur. Since no request was made, the system could not prove that the state of wanting was present, and the informing act could not be recognized.

In situations such as this, it is tempting to reverse the action definition, and make the state of `wanting-to-inform` be an effect of the action `informing-act`, not a precondition. If an informing act is observed, then obviously the agent must have wanted to perform an informing act. As a matter of fact, many expert systems make use of exactly this kind of reasoning–using evidence to reason forward to understand the situation.

But, this is obviously absurd. The state of `wanting-to-inform` cannot be both a precondition of the informing action and also an effect! Something is quite wrong with this logic.

The basic problem is that deductive reasoning is incapable of filling-in any information that is not explicitly present. The kind of reasoning that is needed for this is called "abductive reasoning"–something that can make educated guesses at unobservable states.

One of the best types of abductive reasoning is Pearl's Causal Evidential Reasoning theory [Pea88] (discussed in more detail in Section 10.2). This theory says that actions must be defined carefully, so that reason flows in the direction of causality. Normal deductive-style reasoning must always reason from causes to effects. However, it is permissable to gather probabilistic *evidence*, and to use this to reason backwards from effects to possible causes. The evidence of a particular effect happening tends to make the computer believe more in its possible causes–but the math works out correctly because it is very different from deductive reasoning.

Causal evidential reasoning thus allows a system to reason backwards from observed evidence (such as `a-statement-being-made`) to unobserved causes (such as an `informing-act being performed`, and `wanting-to-inform` being true). The values that a state takes is no longer true/false, but a probabilistic number that indicates degree of belief. Causal evidential reasoning will be an important part of the next understanding system, as is discussed in the design in Section 8 (see Figure 1).

Causal evidential reasoning also teaches us that a cause cannot be an effect of an action, but must always be a precondition. So, in the case of the previous example, it is wrong to build an action that has the state want-to-inform as an effect of the informing act. Building such actions is a mistake. The state want-to-inform is a cause of the informing action, and must be a precondition.

3.3 A Logic-Based System Cannot Numerically Rank Alternatives

As is discussed in Section 7, perhaps the main task of an understanding system should be the disambiguation of possible alternatives. However, this is impossible in a system that is based on binary "true/false" logic. There is no way to represent whether one alternative is better than another alternative.

Fortunately, the NP system uses a five-valued uncertainty logic, consisting of the values "actual/possible/hypothetical/inconsistent/null". Thus, the system in theory is able to represent disambiguations. If any alternative becomes inconsistent, it is automatically discarded. The system does not examine alternatives that are hypothetical or null. If one alternative becomes actual, then all of its competitor alternatives automatically become inconsistent. So, the NP plan-inference system does have a type of ranking system, and can theoretically perform disambiguation of a sorts.

However, in practice, what happens is that everything turns out to be "possible". As long as an alternative is not completely impossible, it must still be "possible" somehow. And, if *every* alternative is ranked "possible", then there is no way to disambiguate between them.

The problem is that the system is using a logic-based ranking system, and not a numeric-based one. Even though the logic has five values, it still cannot represent degrees of belief well. It is necessary to have some kind of system that can represent degrees of partial, uncertain belief, and can reason with them.

There are two viable approaches towards solving this problem. One is to use the uncertainties and the MU calculus developed in [Mye90e] (see Section 12). The second approach is to use the evidential belief factors contained in Pearl's Causal Evidential Reasoning theory (see Section 10.2). It should even be possible to combine the two. In this way, numeric beliefs will be represented, and a disambiguation system will be able to rank its beliefs properly.

3.4 No Time in NP

3.4.1 Problems with Time

A major shortcoming of the current system is that it cannot represent past, present, and future actions. All of the possible and actual facts are thrown together into one big "soup". This is a direct consequence of the desired ability to perform inferences based on a randomly-accessed knowledge base (see Section 2.2.5).

Representation of non-instantaneous or complex (non-primitive) actions would require the definition of a temporal calculus. A major philosophical problem arises when a complex action can be composed out of two or more decomposition actions: when does the complex action take place? For instance, suppose there is a complex action called `question-answer-pair`, that decomposes into `question (Q)` and `answer (A)`. Now, suppose the following dialog fragment is observed (conversation 6), with each utterance labeled by time:

```
T1: Q:  Would you like to participate?          Ask-question
T2:     How much does it cost?                  Ask-question
T3:     8,000 yen.                              Inform
T4:       That includes dinner.                 Inform/advise
T5:     Are the speakers also participating?    Ask-question
T6:     Some of them are supposed to.           Inform
T7:     I see.                                  Acknowledge
T8: A:    Then I would also like to participate. Inform
```

At what time does the question-answer-pair action take place? The question is at T1, while the answer is at T8! Obviously defining the time of the action to be any one single instant will not work–neither T1 nor T8, nor their average (T4?) give satisfactory answers. It is not even enough to define a duration [T1–T8], because many extra things (including an advising act and an acknowledgement act) occur in the middle. It appears that a fragmented duration representation is required: the question-answer-pair action took place during [T1–T1] *and also* during [T8–T8]. Presumably, any higher-level complex actions that used this `question-answer-pair` as a decomposition action, would incorporate the fragmented durations in an analogous manner.

The reason it is necessary to define time is to determine relationships such as *before* and *after*. However, with a fragmented duration representation, these become *strictly before*, *strictly after*, *overlapping*, *interleaved*, *encloses*, *enclosed-by*, etc. [All87]. It is unclear whether such strict mathematical definitions would prove useful or accurate in describing the fuzzy temporal relationships actually used by people in defining actions. At present, the author's research efforts have largely ignored the problem of time, and have concentrated on the other topics presented in this report.

A strict definition of time would also cause problems with the level of actualization of recognized plans. Currently, the difference between "can do", "will do in the future", and "has done already" is ignored. Otherwise, actualization paradoxes would result. For instance, when the office says, "I will send you the form", the system represents that the office sends the form, the questioner receives the form, and that the questioner has the form. If someone does not have something that he wants, then he will work to get it. If the questioner does not have the form, the questioner is not satisfied, and the questioner will continue to bother the office to cause the office to send the form, until the form is received. However, this does not happen–the questioner stops asking after the office says that the office will send the form. The immediate solution has been to ignore time, and treat something that *will* happen as something that *has* happened. The long-term solution will require a theory of possible and probable future events, in which the questioner is satisfied because he believes that he *probably will* receive the form, in which case he *probably will* be able to attend the conference. But there are still major philosophical problems in uniting this with intentional action theory, and representing the difference between events that one believes "probably will" happen as a result of one's own actions, and events that one believes "probably will" happen *not* as a result of one's own actions. If I believe that I really will get a million dollars after I work for ten years, and *it's going to happen*, then there's no reason for me to work for ten years. It's going to happen! The solution to this paradox lies in the direction of multiple possible future worlds, where what could happen in one world affects the actions that an agent takes in a different world. It is expected that the UNDA theory should be useful in helping to solve these problems, but the theory for probable expectations affecting intentions has not been developed yet.

3.4.2 Problems with Nonmonotonicity

Another aspect of the time problem is the inability of the current NP plan inference system to represent the deletion of a state in a possible plan. Currently, effects can only add states, they cannot delete them. This is a direct result of a design decision to build NP on top of a plain ATMS.

A plain ATMS can represent different multiple possible-worlds that are, in effect, timelessly stacked in parallel. States can be added; states can be deleted; states can be possibly-true in some worlds and not-true in other worlds. However, it is impossible to represent both the past and a *nonmonotonic* future in *multiple* worlds at the same time, because this would mean the ability to represent a state that is true but then could become false for part of the system, and there is no way to discriminate between when it should stay true and when it should become not-true. This requires, in effect, a two-dimensional array—a sequential list of parallel worlds.

There are two approaches to this problem. The first approach is to only allow possible states to be added monotonically. This is actually not such a large restriction; the states can still be assumptions that are both true and not-true at the same time. It is also possible to work with negative states, i.e. states with a negated content; when the state becomes true, the content becomes false. Also, if a state retraction is desired, the state can either be made inconsistant, or it can be completely deleted. The only thing that is not possible is to represent a state that both could be true and could have been deleted at the same time. This is the approach taken by the current NP system.

The monotonic approach has the advantage that planning (plan recognition, prediction, and inference) is thus bounded and fast. Planning can be done easily in an implicit manner. The mathematics automatically determines the order of actions, and no explicit search needs to be performed. Each new action or state is simply put into "the soup", and then the desired answers are read off of the corresponding nodes. This results in a plan inference system that gives all possible answers, and finishes in a short, finite amount of time.

The second approach to the problem is to augment the ATMS by building a system that allows states to be added in a nonmonotonic fashion. The (one-dimensional) "parallel" possible-worlds representation becomes a (two-dimensional) tree, where branches of the tree represent different possible world-lines or world-histories, and the depth of each branch of the tree represents the number of actions that have been performed in that possible world—the length of the history [MN86]. A system called "Action Worlds" that demonstrates this capability has been implemented, using the ATR ATMS. However, it has not yet been incorporated into version 3.3 of the NP plan inference system.

The nonmonotonic approach has the disadvantage that plan inference is slow and unbounded. Planning must be done explicitly. The order of the actions needs to be specified explicitly by part of the system, and searching needs to be performed explicitly. Each new action is attached onto a particular branch of the tree. Since action order is significant, a single action might have to be explicitly attached at many places. Since a nonmonotonic action set will usually contain one action that reverses the effects of another action, normally the set of possible answers will be infinite, and the system may also take infinite time to finish.

There are a number of possible approaches to get around the difficulties shown by the slow, unbounded nonmonotonic system. The first approach is to somehow combine the monotonic system with the nonmonotonic system, into a *hybrid system*. In theory, the fast monotonic system would come up with a partial plan that works well or that almost works. Next, some kind of "plan repair module" would convert the monotonic plan over into the nonmonotonic system, analyze the places that do not work, and add the necessary nonmonotonic steps to the plan. Such a plan inference system would probably be faster than a simply nonmonotonic system.

The second approach is to build a case-based, thesaurus-based *habitual system* that recognizes successful plans directly, by understanding the type of the required plan and instantiating it with respect to the current situation. The system would probably use a neural-net module for its recognition process. Further design specifications for such a plan-inference system are discussed in Section 8. It is anticipated that such a habitual plan-inference system would be much faster than a simply nonmonotonic system.

4 Research on Translating Feature Structures to Logical Forms (FS-LF)

4.1 Results

As an example of how the current FS-LF system can be used, Appendix C shows the system translating conversation 1 from a Nadine-style feature-structure representation into a FOPC-style logical-form representation. The presented logical-form representation is *not* unique, and should be taken as an example only. It is based on the types of logical forms presented by Hobbs and Kameyama in [HK90]. For instance, it constantly encodes the speaker and hearer case information as `(person x)` and `(person y)`. If other styles of logical forms are desired, the translation rules specified in Appendix D can be rewritten in a straightforward manner. Note, however, that it is not useful to create very elaborate specifications or examples until the logical-form program that will use the FS-LF system's output has been specified, as is discussed in Section 4.3.

Translation from feature structures to Iida/Arita/Yamaoka-style logical forms has been demonstrated in the appendix of a previous report [Mye90a].

4.2 Improvements to the FS-LF system

A new version (1.3) of the FS-LF "feature-structure to logical-form" translation system was implemented and tested. It was not necessary to add any new commands. The new version has the following features, which should be added to the FS-LF manual [Mye90a]:

4.2.1 Logic Characters are Allowed in Logical Forms:

Version 1.3 permits the usage of logic characters, e.g. “∃∀ ∧ ∨¬ ←→≡⊂⊃ ∩∪” and others, as atomic constants or as parts of other symbols, inside logical form specifications. The symbols are accessed using the SYMBOL key on the Symbolics. For a summary of allowable symbols, type SYMBOL-HELP. This capability is demonstrated in the examples (see Appendix C).

4.2.2 Deep Feature Structures are Allowed:

The previous version 1.2 of FS-LF did not correctly translate feature structures that were unexpectedly deep, but instead returned the {FS} value. This problem was discussed in the first point of Section 6 on page 7 of the FS-LF manual. The current version 1.3 will now translate arbitrarily deep feature structures. `FS-to-LF-spec` variables are now allowed to represent feature structures, as well as constants. Such feature structures are translated recursively. In the current version, the translated deep feature-structures cannot appear in arbitrary places in the resulting logical form, but must be nested at the corresponding variable location.

4.2.3 Operator Scope Not Explored:

The current system does no computations with operator scope. Each operator is translated directly from the input feature structure, and the operator's arguments are translated directly as well, in the manner specified by the `fs-to-lf-spec` specification. The operator's arguments are left in their nested order. In some logical formalisms, it is possible to unnest nested modal operators, and obtain multiple correct answers, based on scope interpretation. This issue was not explored in the current system 1.3. If desired, this feature may be modified in the future.

4.2.4 Unexpected Adverbial And Prepositional Phrases Not Translated:

This is a direct result of the “significant contents of a FS must be known ahead of time” feature discussed in the fourth point on page 7 of the FS-LF manual. Again, the rationale is that logical forms encode information in a position-specific manner; therefore, the informational code for each position of the logical form must be known ahead of time. Optional arguments are not supported. (If a required argument is not found, the system will return NIL for its value and continue processing, but it will also complain with a warning message.) This feature may be modified in a future version.

4.2.5 New Variable: *FS-LF-translate-loops*:

The current system 1.3 has problems with co-reference variables in feature structures that form *cyclic* loops. (If the co-reference variables form *non-cyclic* loops, i.e. directed-graph networks, then there is no problem.) To combat this problem, a new variable, `*FS-LF-translate-loops*`, is introduced. If this variable is `T`, all loops are translated. However, if there is a cycle, the machine will stack-overflow. If this variable is `NIL`, the program will not translate a branch that it has seen before, but instead will substitute the constant `**ALREADY-TRANSLATED-LOOP**`. However, this will happen for all loops, including harmless co-reference networks. Since cyclic loops are rarely encountered in feature structures, this variable should be kept set to T. This feature will be improved further if there is demand.

4.2.6 Coreferences are Expanded:

In the current version 1.3, all coreferences are in effect expanded and dereferenced. Thus, there is no difference between feature structure branches that are identical and ones that are equal but not identical. This feature could be changed if there is demand.

4.2.7 Lisp-Reader Characters are Not Allowed in Logical Forms:

Since the logical forms given to FS-to-LF-spec are read by the standard lisp-reader, they are not allowed to contain unusual format-control characters that are not used for their normal purposes. It is currently impossible to use the following characters as *symbols* in a logical form: “ "#‘ , : | ⊗ ”, without having them take on their current LISP meaning (e.g., “;” stands for “ignore everything else to the end of the line, as a comment”). In particular, it is currently impossible to use the colon (“:”) or the comma (“,”) as an atomic character by itself. These characters may be quoted by using a backslash (e.g., “\:”) before one of them, but in this case the atom is printed as surrounded by bars (“|:|”) by the LISP printer. If there is sufficient demand, this feature may be modified in the future.

4.3 Comments on Translating Feature Structures to Logical Forms

The current version of FS-LF fulfils its literal task—that of translating feature-structures into logical forms. However, there are some problems with the present system that must be mentioned. These problems deal not so much with the FS-LF system itself, as with the concept of a feature-structure to logical-form translation system.

The current system will take the surface form of a feature-structure utterance, and translate it into a surface-form logical form. However, it cannot fill in missing information. That is the job of an anaphora-resolution and ellipsis-resolution system, not a feature-structure-to-logical-form system. An anaphora-resolution system can be built to use feature structures, or it can be built to use logical forms, in either manner. But a form-translation system should not incorporate the large knowledge bases and inference mechanisms required for understanding–it is better to build a special-purpose understanding system that is specifically designed for anaphora-resolution before or after form-translation, than to try to fit this function into a form-translation system.

Thus, there is insufficient information in the surface form of the utterance itself, whether it is expressed in feature structures or logical forms, to be able to create a deep semantic representation. There are at least three major sources of knowledge required in order to understand the deep semantic meaning of an utterance in the general case:

1. The utterance itself
2. The situation of the agents
    - (a) The history of the dialog (dialog context)
    - (b) The environment of the agents (situational context)
    - (c) The desires and intentions of the agents (attitudinal context)
3. The knowledge of the agents
    - (a) Shared knowledge
        - Cultural knowledge
        - Personal histories
        - Scripts
    - (b) Common-sense knowledge
    - (c) Plans and actions for attaining goals

Since the utterance's form contains only the first source of information, the amount of information in the utterance itself is insufficient for general translation into a deep logical form.

However, the beliefs, intentions, desires, and knowledge of an agent are directly *unobservable.* This information can only be guessed at, based on evidence provided by the observable conversation.

From these statements it is possible to conclude that any deep-form translation system that abstracts meaningful logical forms using only deductive reasoning, which reprocesses existing information, is doomed to fail. It is necessary to build a shallow-form translation system, and couple this at least with an abductive reasoning system, in order to derive meaningful deep-semantic-representation forms.

Information theory says that the amount of information contained in a message is independent of its representation. The surface representation of a concept or an amount of information is important only when looking at issues of efficiency. The amount of information contained in a data-structure remains the same whether the data-structure is represented as a feature-structure, a set of nested frames, a group of objects, or as a logical form. (This happens in the same way that 3/4, $(9 \div 16)$, and 0.75 all represent the same information). The only considerations are how long it takes to extract the desired information, and whether a program that processes the results is accustomed to working with such data structures.

Obviously, if a program is accustomed to working with feature structures, then it cannot work with logical forms. Conversely, if a program is efficient at working with logical forms, then it cannot work with feature structures. However, the power of a program lies in the types of algorithms that it uses, not in what representation method it uses to manipulate information. In theory, a program written using one representation method could be translated into a program using a different method.

Thus, a feature-structure-to-logical-form translation system is useful for allowing a program that works with logical forms to work on the data. Unfortunately, in order to take full advantage of such logical forms, ATR would have to have a first-order predicate-calculus theorem-prover program, similar to the one used by Jerry Hobbs and Mark Stickel at SRI [HSME88,Sti90]. At the present we do not have such a program. It is also known that current-technology general theorem-proving programs (typically based on breadth-first search) can take a long time to find an answer, so it is unclear whether ATR would want to use such a program or not.

Thus, it seems that translating a feature-structure surface representation into a logical-form surface representation uses processing time unnecessarily. After the hypothetical theorem-proving system understands the logical form, it is still necessary to translate it back into feature structures for input to the transfer and language-generation systems, which would use even more time. It seems better to build an understanding engine that works directly with feature structures, as the current NP system does.

At the same time, it seems that the real problems involved in creating a useful utterance representation have to do with anaphora, ellipsis, and zero-pronoun resolution, and the other disambiguation problems discussed in Section 7. Whether this disambiguation is performed by a logical-form computer program, or whether it is performed using feature structures for computation and then possibly translating the results into logical forms, it seems like these are the hard problems that should be worked on. Therefore, Section 8 presents a design for a disambiguation system. The results of this system can then be translated into logical forms, if desired.

In addition to these practical problems, there are also theoretical problems with the logical form representation. The First-Order Predicate Calculus (FOPC) used most often for logical-form representation is based on a binary, true/false logic describing clear, concrete situations in the actual world. This logic is not powerful enough to represent at least the following phenomena:

- Fuzziness
- Probability and chance

- Possibility, ability, and impossibility
- Causality
- Hypothetical, and possible future, situations
- Local but not global inconsistency
- Degrees of uncertainty

In addition, the calculus is designed around a very small set of operators, including *all/every*, *at-least-one-exists*, and *not*. This is useful for representing sentences involving indefinite quantifiers, including the words *some, any, all, every*, etc. Such sentences occur in only about 4% of the utterances; the problems of representing and interpreting such utterances only comprise about 0.7% of all the practical interpretation problems encountered [MT90]. But there is a very large body of more important operators that this calculus ignores completely. Such modal operators as WANTS(), BELIEVES(), THINKS-ABOUT(), INTENDS(), CAN-DO(), REMEMBERS(), OWNS(), POSESSES(), RESPECTS(), MIGHT-DO(), EXPECTS(), BENEFIT-OF(), etc., etc., are not included in this calculus at all. And, simply representing these operators is not enough; they have scope and calculation rules, just as $\forall$ and $\exists$ do. Thus, it seems that the current logical-form calculus is inadequate for representing and working with most significant common concepts. Research efforts should be devoted to creating new calculi that meet these practical needs, rather than attempting to define everything as a "there-exists" problem.

There is also a growing body of evidence, briefly discussed in Section 3.3, that a non-numerical logical calculus will be insufficient. At the least, the system will need to be able to represent degrees of belief, degrees of desire, and degrees of commitment. This argues for a numerically-based calculus, and again provides a further argument for making a special-purpose understanding system, not an unspecific logical-form translation system.

In addition, the logical-form operators currently used by researchers seem ad-hoc and easily open to ambiguities. Whenever a new concept is needed, it is apparently simply defined from vocabulary by the current researcher (e.g., EVERY-PROFESSOR-OF() OFFICE(), TOKYO(), or MACBETH-MURDERS()). There is no apparent attempt to provide basic definitions. Even if there were, the operators are still open to ambiguities from cultural-specific biases. For instance, SISTER(), besides having at least five meanings in English, is ambiguous between at least IMOTO() and ANE() in Japanese.

Finally, most efforts in logical-form representation of sentences have focussed on representing pathological sentences, or other artificially constructed examples, while ignoring real sentences from actual dialog. Nobody uses "It was the rutabagas that Henrietta ate." in normal, everyday dialog [Sid83]. It is important to focus on grammatical constructs and representation problems that are frequent and are encountered in the data, instead of esoteric problems that are theoretically interesting but practically rare.

In conclusion, although ATR now has the capability to translate Nadine feature structures into logical forms, it seems that the real research problems center around disambiguation, and definition of models of action and intention. Research is correspondingly centering around applying these to dialog understanding.

Table 1: Table of Speech-Act Primitives

Illocutionary Force Categories	Speech-Act Primitives
Assertives	Inform
Commissives	Promise Offer Accept/Consent Reject/Refuse
Directives	Ask-question ("Askq") Request Suggest/Advise/Recommend Direct Order Permit Prohibit
Declaratives	Confirm Name Self-Identify
Expressives	Thank Greet Say-Goodbye Apologize Complain/Protest Compliment Welcome Acknowledge

5 Illocutionary Force

As a first step in building a program that can automatically recognize and understand illocutionary force, research in the representation and use of speech acts was performed.

The two major authorities on speech acts are Searle and Wierzbicka. Searle lists eight types of illocutionary forces in [Sea85], and later, with Vanderveken, 107 English illocutionary-act verbs divided into five illocutionary force categories [SV69]. Wierzbicka presents an excellent dictionary that explains 270 English speech act verbs divided into 37 general-meaning groups [Wie87]. Twenty of the verbs Searle and Vanderveken present are not included in Wierzbicka's dictionary. 183 of the verbs in Wierzbicka's dictionary are not listed by Searle and Vanderveken.

A first step in working with illocutionary force is to designate a complete set of representational primitives. These are presented in Table 5. Searle's illocutionary force categories are used to classify Searle's and Wierzbicka's speech-act verbs (see Section 5.1 for an explanation). Extra speech-acts have been added as needed. Only the set of speech-acts actually used in ATR's corpus are used, and the rest ignored. (For instance, "threaten" has been left off of the list.) However, if other speech-act verbs are needed in the future, they will be added.

It was found necessary to add two original speech-act types:

Self-Identify This is the action of the speaker informing the other person of the speaker's name. It is not a *naming* action, as the speaker already has a name. However, it is a declarative, and not an assertive—once the speaker says, "My name is X", it changes the world, and the hearer must now call the speaker by that name.

Acknowledge This is the action of the speaker showing that he has heard what the other person just said to him. It is an expressive, and not an assertive, because it communicates an attitude or feeling of the speaker. A typical acknowledgement is "hai", or "O.K.".

5.1 Explanation of Illocutionary Force Categories

This section briefly explains the illocutionary force categories developed by Searle that will be used in understanding ATR's dialogs.

5.1.1 Assertives

An Assertive is the action of providing a fact or making a statement. An assertive states something about the *world*. Assertives can be glossed in English by using the word *that*: <A> <inform>s <B> **that** X.

5.1.2 Commissives

A Commissive is an action that is performed by saying something. It commits the *speaker* to some kind of future action. It in effect forms at least half of a contract or a deal between the speaker and the hearer.

5.1.3 Directives

A Directive is the action of directly attempting to cause an action in the hearer by saying something. It attempts to make the *hearer* do something. Directives can be glossed in English by using *to*: <A> <request>s <B> **to** X. Directives are marked by having at least the following features: `refusal-allowed` +/-, `power-or-authority` +/0/-.

5.1.4 Declaratives

A Declarative is a de-facto action that is accomplished through the act of saying something. A declarative changes the *world* because it was said. Declaratives are sometimes called "performatives". They can usually be glossed in English by using the word *hereby*: <A> **hereby** <name>s <B> X.

5.1.5 Expressives

An Expressive communicates something about the state of the speaker. It expresses the feelings or the attitude of the *speaker*. Expressives convey emotion.

5.2 Classification

Conversations 1-5 were classified by hand using these speech-act primitives. Each utterance was labeled with its apparent illocutionary force(s). If a direct force and an indirect force were detected, they were labeled as such. The results are shown in Appendix B. These results may be useful in automatically abstracting illocutionary force from surface utterance forms.

In order to clarify the relationship between surface feature structures and illocutionary force, the utterance illocutionary forces for conversations one through five were reclassified under the feature-structure relationship of the surface utterance. These results are also shown in Appendix B.

6 Scheduling Parallel Processes

6.1 Exercises

Preliminary research has been started on the problem of dynamically coordinating and scheduling parallel processes on the Sequent. An extremely simple Sequent CLIP Lisp program has demonstrated the ability to run groups of parallel processes sequentially by group, and to combine and coordinate the results of multiple parallel processes. It is expected that the resulting experience will be useful in programming dynamic scheduling algorithms. In addition, it should be extremely straightforward to implement the causal evidential reasoning engine discussed in Section 10 on the Sequent.

Example runs are shown in Appendix G. The examples demonstrate the current capability to perform static scheduling using two models: (1) the "parallel fork-join" model, where each stage has to wait for all of the parallel results of the previous stage to be finished; (2) the "parallel pipeline" model, where messages can flow through the system in sequential paths at their own speed.[3] These are combined into example (3), which demonstrates static scheduling of a random number of processes created at run-time using both parallel fork-joins and parallel pipelines, in a manner that is realistically consistent with the ATR interpreting telephone task. A single speech-recognition process results in multiple utterance candidates; each candidate results in multiple parses; each parse has to be understood, and assigned a simulated score. The understanding results are combined using a join, and the highest scoring parse is chosen for transfer, language generation, and speech generation.

6.2 Theory

6.2.1 The Issues

A complex interpreting telephone system is composed of many subsystems, each of which might require a number of processes. The problem in scheduling is to determine the *order* of processing, *which* processes get allocated processing resources (time and computers), *whether* to continue to allocate resources to a subsystem or not, and *when* to stop processing and force an answer to be spoken.

[3]It is also possible to implement a "serial pipeline" model, where messages flow through the system in a single pipeline, the speed of a message being determined by the speed of the message in front. However, it seems that serial pipeline processing will probably not be needed for ATR's problems.

A scheduling problem results because the processes are slow, the information to be processed is large, the system is limited in both computational and time resources, the system can produce various qualities of interpretations, the system generates many different results from which one has to be chosen, and the system does not have to come up with a result.

Current technology indicates a high-quality speech-to-speech interpretation time of many minutes. Even with improvements in the algorithms and in the computational hardware of a few orders of magnitude, the process would still take many tenths of seconds. Scheduling will remain a serious problem for at least the next twenty years.

The number of inferences that can be made from a single utterance are large, perhaps unbounded. The amount of resources that can be spent on polishing and evaluating a generated utterance is also extremely large. It seems as if it is always possible to spend more time evaluating the implications of the meaning of an utterance, or polishing how it should be generated. So, the amount of information-processing to be done appears to be able to fill as much time as can be allowed; again, scheduling is required.

A computational resource limit is imposed by the number of computers and processors, of various types, available to the interpreting system. In general, for a single-conversation system, this number will be fixed. A time resource limit is fixed by the patience of the listener and the reputation of the system. If the system consistently takes a long time to interpret conversational utterances, the listener will be dissatisfied, and will not want to use the interpretation service in the future.

The problem is made more complex by the fact that there is no hard limit on the length of a conversation. It may be acceptable to take a long time to interpret a few utterances, as long as the average interpretation rate is fast. In addition, it is important for the scheduler not to stop a process if it is about to get a significant answer. So, the scheduler, which wants to produce an answer quickly, must continually negotiate with the various interpretation processes, which need more time to produce clean and correct interpretations.

The problem is also made more complex by the fact that it is possible to output various *qualities* of interpretations, and still perform capably. Speech recognition can report only the top one or two candidates, which could be incorrect. Parsing can misunderstand the meaning of one or two words. Understanding can pick a bad parse or an O.K. parse instead of a good parse. Transfer can translate an expression literally, instead of detecting idioms and canned expressions. Language generation can create a literal sentence, instead of polishing it with anaphoras and ellipses. Speech generation can create a flat utterance, instead of polishing it with prosodies and coarticulation smoothings. In all of these cases, it is possible to sacrifice quality for speed, or obtain exceptional quality at the cost of taking much processing time. It will be necessary for the scheduling program to understand these tradeoffs, and make decisions as to which processes are important and should be given more time.

The system generates many different results. One of these results must be chosen. The system must determine when to stop processing and pick the best result.

Unlike conventional computer programs, it is legal for the computer to say, "I can't interpret your last utterance. Can you please rephrase it?". The system can give *no result.* Although this is not desirable, it is a permissable outcome that must be considered when trading off processing time versus quality.

6.3 Requirements

6.3.1 Estimate of Quality of Existing Results

An effective scheduling and control algorithm demands that subsections provide estimates of the quality of their results, after the results have been obtained. This is needed in order to specify which results should receive priority in further processing.

All estimates obtained by the subsystems (1) may be incorrect, and (2) are allowed to be expressed in ranges or distributions rather than exact scores. An example of the first quality is the score estimates currently yielded by the Speech Recognition subsystem. Sometimes the correct utterance is ranked third in the scoring. However, the scheduling algorithm should be robust enough to deal with such problems.

6.3.2 Estimate of Time Needed for Next Results

It is useful for a subsystem to yield estimates of the amount of time needed to obtain the next level of results. The scheduler can use this information to allocate processing priority.

6.3.3 Estimate of Quality of Next Results

If possible, it is also useful for a subsystem to yield estimates of the quality of its next level of results. This is useful in trading off processing time versus result quality. As before, such results can be incorrect, and they can be interval ranges rather than scalars.

6.3.4 "Anytime Algorithm" Partial Results

When possible, a subsystem should (1) obtain and store *partial results* or *results of low quality* while it is working on obtaining the actual results of high quality. The subsystem should also (2) be able to report these results at any time, and (3) be able to stop working at any time.[4] This concept is discussed slightly more in the Section 6.4.1.

6.3.5 Timers and Statistic Gathering Routines

The scheduler itself must keep track of the amount of time that the subsystems actually use. It must incorporate routines for gathering statistics of operational characteristics over a long term.

6.3.6 A Theory of Exasperation

It will be necessary to develop a theory describing how impatient a customer is, as a function of the amount of time he or she has to wait. Observations, such as the customer's tone of voice, and active inputs, such as a "hurry-up" button that the customer can press, will also be important. This theory will be used to generate upper-time-bound soft limits on the interpretation process.

[4]Also, ideally, the subsystem should (4) be able to resume working on a stopped job if so commanded. However, this is an advanced capability that may not prove useful, so it will not be stressed at the present.

6.4 Methods of Attack

6.4.1 Anytime Algorithms

Recently, the problem of acting under limited resources has become a popular research topic. One important new concept is that of the *anytime algorithm*, first proposed by Dean and Boddy in [DB88,BD89]. A so-called "anytime algorithm" is an algorithm that gives results of generally increasing quality over time.

Typical present-day computer algorithms accept input, process the input for a long time until they are finished, and then yield results. A graph of the quality of the results versus time looks like a step function: the quality of the results is zero until the algorithm is finished, at which point it jumps to full quality and remains at that level afterwards.

However, an "anytime algorithm" can give results of increasing quality over time. An "anytime algorithm" accepts input, and perhaps immediately computes a preliminary answer. It then attempts to refine the quality of its answer by further computation. The graph of the quality of the results versus time looks more like a ramp, or a staircase function, than a step function.

This theory is not so important if the computer is used in a static sense, for single answers, when it is not significant how long the customer has to wait for results. However, anytime algorithms become important when the computer is performing an adaptive, real-time service, such as interpreting between two people.

6.4.2 Incremental Time Allocation

The scheduler will begin by estimating the total amount of time allotted for interpreting the utterance. This depends on the amount of time that the user has had to wait for previous utterances, and the estimated level of impatience of the customer. The total time allotment will generate an estimated soft *ceiling* on the amount of time spent interpreting.

Next, the scheduler will estimate the amount of time required to interpret the utterance, and the quality of the interpretation. This will be a sliding scale, rather than scalar estimates. Since it is necessary to maintain a minimal level of quality, this will establish an estimated soft *floor* for the amount of time spent interpreting.

At this point, the scheduler will perform a trade-off on the utility of time versus quality, and come up with a target duration for the interpretation process that attempts to obtain good quality results in a small amount of time.

The scheduler will then allocate processing resources to the various subsystem processes. As the processes obtain results, the scheduler will keep track of the quality of the results and determine whether further processing is required. As time progresses, the scheduler may also decide to halt a process at any time and demand its current results for processing by the next stage. This demand can be a soft demand, subject to negotiation between the scheduler and the subsystem process, if the process feels that it can obtain good results if it is allocated just a small amount of additional time.

The scheduler will retain the results and keep track of the best one. Sometimes, if the scheduler spends more time processing results, it will obtain a higher quality result than the current best one (e.g., if the Speech Recognition score estimates were incorrect). But finally, the scheduler must decide to stop the processing, and report the current best result as speech output to the hearer.

Note that it is possible for the system to get in a few more half-seconds of processing time as the Speech Generation system is speaking. It is thus possible for the system to "change its mind" and take back an erroneous interpretation in the middle of its speech. The system can also make hesitation noises ("Ehhhhh touuuuuu...", "Ummmmmmm") if it estimates that the interpretation will take an unusually long period of time to process.

6.4.3 Estimate Reliability Estimation

An important task of the scheduling algorithm will be keeping track of the quality of the eventual results, and the corresponding submodule-estimate reliability. When the reliability of the submodules' estimates is known accurately, the scheduling algorithm can also be more accurate. Accurate submodule estimates will be used actively, while inaccurate submodule estimates will be discounted.

6.4.4 Limited Resource Allocation Problem

One method is to view the scheduling problem as a classic *resource-allocation problem.* A particular branch of mathematics has been developed to deal with static resource-allocation problems. The mathematics starts by assuming a limited known quantity N of a particular limited resource, a known number n of activities j that each get allocated x_j amount of the resource (with $x_j >= 0$ and $N = \Sigma_1^n x_j$), and a known evaluation function $f(x_1, ...x_n)$. Mathematical methods vary depending upon the size of the problem, whether the resource is discrete or continuous, whether an exact or an approximate solution is required, and whether it is important to apportion the resource to the various activities in a "fair" manner or not. Methods also vary widely based on what is known about the function $f()$, e.g. whether or not it is separable, convex, or bounded [IK88].

7 Disambiguation is the Main Task in Understanding

The current design for an interpreting telephone (including speech recognition, syntactic and semantic parsing, transfer, generation, and speech generation), will be incapable of correctly translating certain kinds of language problems. It is expected that an understanding system must fill in these gaps, and provide answers to these problems. In order to build a good language-understanding system, it is necessary to understand and characterize these problems.

A previous study [MT90] investigated the known tasks that an understanding system would have to perform because the current system design is not powerful enough. These results have been analyzed, and are presented here grouped by type.

The largest type is "disambiguation". Disambiguation is defined as the problem of choosing between multiple alternatives, based on which alternative "makes the most sense". A disambiguation system requires components to generate the different alternatives, predict what is expected, reason with the alternatives and determine how likely they are by how much they match expectations, rank the alternatives, and decide on the best one to choose. Such a system will also require knowledge sources, in the form of rules that are used for reasoning.

The results of the study show that disambiguation is extremely important, and accounts for about 60% of the number of problems encountered in translating conversations. If a

Table 2: **Disambiguation Problems**

PROBLEM	% UTTERANCES	% PROBLEMS
Disambiguation of ambiguous speech & parsing results	100	17.4
Article Generation (The/A/null etc.)	89	15.5
Difficult Prepositions, Postpositions and Particles	28	4.9
Subject Generation	25	4.3
Verbal Honorifics and Humble Forms	21	3.6
Nominal Honorifics	14	2.4
"Will" Generation Required	12	2.1
Incomplete Specification	12	2.1
Zero Indirect Referrent	12	2.1
Closing Signals	7	1.2
Plural Generation Required	6	1.0
Ambiguous or Vague Vocabulary	5	0.9
Usage of Indefinites	4	0.7
Aspect Generation	2	0.3
Zero Verb and Case Markers	1	0.2
Gender Determination for Titles	1	0.2
Total Disambiguation Problems	339	58.9

Table 3: **Prediction Problems**

PROBLEM	% UTTERANCES	% PROBLEMS
Prediction of Next Utterance Content for Speech Recog.	100	17.4
Total Prediction Problems	100	17.4

Table 4: **Prosody Problems**

PROBLEM	% UTTERANCES	% PROBLEMS
Representing, Recognizing, Understanding, Transferring, and Generating Prosodic Information	100	17.4
Total Prosody Problems	100	17.4

Table 5: **Other Problems**

PROBLEM	% UTTERANCES	% PROBLEMS
Softening Aspect (-ga, -no de, -n, etc.)	16	2.8
Ill-Formed Input Utterances (Spontaneous Speech)	15 (est.)	2.6
Short Answers	2	0.3
Causative Transfer Problem	2	0.3
Difficult or Impossible Interpretations	2	0.3
Total Other Problems	37	6.4

Table 6: Summary of All Types of Problems

PROBLEM TYPE	% PROBLEMS
Total Disambiguation Problems	58.9
Total Prediction Problems	17.4
Total Prosody Problems	17.4
Total Other Problems	6.4

prediction-based disambiguation system is built, so that the system can do both prediction and disambiguation, then this would account for about 75% of the problems. Prosody is also extremely important, accounting for a remaining 17% of the problems.

The conclusion is that dialog-understanding researchers should concentrate on designing and building a disambiguation system that can perform prediction as well. Also, prosody should be investigated.

8 Description of a Design for a Disambiguation-based Translation System

A conversation-understanding system based on prediction of utterances and disambiguation has been designed. It is estimated that such a system should, in principle, be able to solve approximately 75% of the problems known to be encountered in automatic translation, leaving prosody problems and other problems totalling 25% as topics for future research (see Section 7). The design is extremely ambitious, and requires the incorporation of several new software technologies. However, it is necessary for ATR to implement such a design in order to attain the goal of having a general-purpose understanding system that can deal with conversations it has not previously been trained on.

A diagram of the proposed design is presented in Figure 1. Each component in the design will now be briefly explained.

The core of the understanding system is a disambiguation system. The disambiguation system will accept multiple alternatives that describe possibilities concerning the current utterance. It must then choose the most probable consistent set of possibilities, based on the known evidence, as its recommended disambiguated belief. This belief will then be reported to the rest of the automatic interpretation system, and used for transfer or generation. In addition, if desired it could be passed to a logical-form understanding system, for further processing. An advanced disambiguation system capability would include the ability to detect when there is not enough information to decide on a particular set of beliefs, or when the chosen set is too implausible to be accepted.

The disambiguation system will be based on the use of three other systems: a causal evidential reasoning system, a thesaurus-based illocutionary-act distance metric, and an utterance-and-speech-act prediction system. In addition, the disambiguation system, and all of the other systems mentioned here, will have to make use of a comprehensive plan library.

The causal evidential reasoning system will be an engine that represents beliefs in concepts, and the stochastic causal relationships between these concepts. If one concept normally causes another concept to be true, then this probability will be represented. The system will accept uncertain evidence, in the form of observations of utterance possibil-

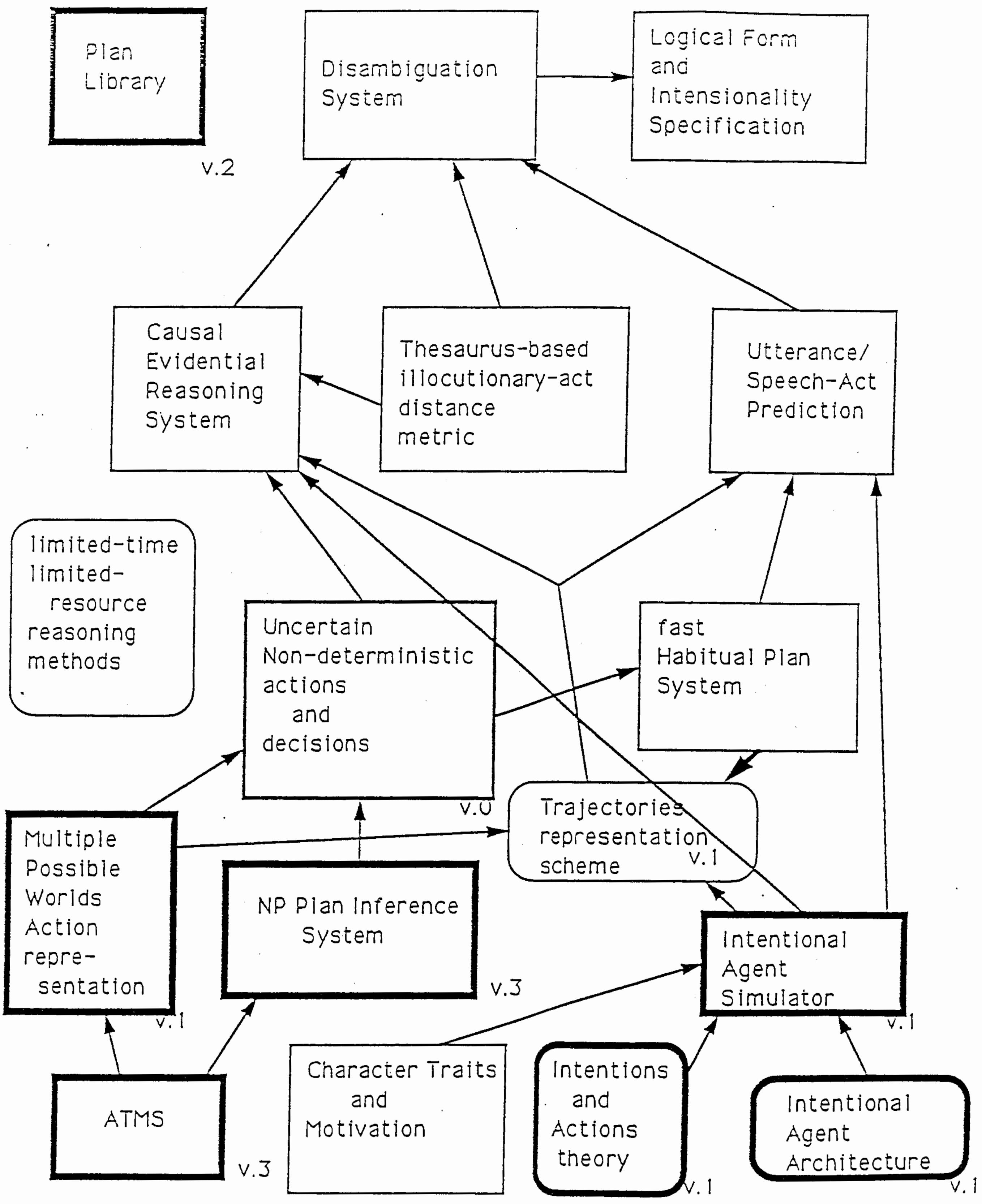

Plan Library
v.2
Disambiguation System
Logical Form and Intensionality Specification
Causal Evidential Reasoning System
Thesaurus-based illocutionary-act distance metric
Utterance/ Speech-Act Prediction
limited-time limited-resource reasoning methods
Uncertain Non-deterministic actions and decisions
v.0
fast Habitual Plan System
Trajectories representation scheme
v.1
Multiple Possible Worlds Action repre-sentation
v.1
NP Plan Inference System
v.3
Intentional Agent Simulator
v.1
ATMS
v.3
Character Traits and Motivation
Intentions and Actions theory
v.1
Intentional Agent Architecture
v.1

Figure 1: A Design for a Disambiguation-Based Understanding System

ities, from semantic parses of the speech recognition system's output. It will maintain the degree of belief in concepts, based on prior probabilities of the concepts and their causal relationships. The causal evidential reasoning system will use the thesaurus-based illocutionary-act distance metric and the Uncertain Non-Deterministic Action (UNDA) plan inference/representation module (see Section 12) to help form its networks.

A crucial question is where the probabilities and causal relations, which the causal evidential reasoning engine needs in order to form beliefs, will come from. These will be taken from predictions offered by the trajectories module, and the intentional agent simulator. It is necessary to have good predictions of the probabilities, in order to get valid beliefs, which will eventually be used for disambiguation.

The thesaurus-based illocutionary-act distance metric must compare the difference, and measure the distance, between two illocutionary acts. This is necessary in order to measure the distance between observed and predicted speech acts. Since an illocutionary act consists of an illocutionary force plus a content, this metric must measure both the distance between two illocutionary forces, and the distance between two semantic utterance contents. The illocutionary force distance metric algorithm should be specified by hand; a good starting place is the hierarchical classification given in Section 5. The system will eventually have to deal with the problem of an utterance that validly has more than one illocutionary force. The semantic utterance distance metric will be harder to specify. A good starting place will be the thesaurus distance between the two verbs of the sentences, combined with the distance between the two subjects and also between the two objects. The topic promises to be a difficult one, requiring some serious research.

The utterance and speech-act prediction module must form predictions of the next speech act (and corresponding utterance) in the conversation. The module will predict not just one speech act, but a number of possibilities, each one being labeled with an associated probability. These predictions will be fed to the speech recognition system, which will use them to help recognize the next utterance.

The utterance and speech-act prediction module will use the results of the trajectories module, along with those of the fast habitual plan system, and the intentional agent simulator. Each of these systems will contribute suggestions as to which speech acts the conversation participant will perform next. The utterance and speech-act prediction module must integrate these suggestions, make sure they are labeled with probabilities in a consistent manner, and package them for processing by the disambiguation system and the speech recognition system.

The plan library consists of a dictionary of all kinds of actions that are possible to perform, along with all kinds of semantic meanings. It is unfortunately necessary (although not sufficient) to have such a large dictionary of actions, if the system is to be able to deal with all different kinds of input–even if restricted to the conference domain—in a robust manner.

The fast habitual plan system feeds results to the utterance and speech-act prediction module, and the trajectories module. A habitual plan system is necessary for two purposes: it should predict the habitual plans of the conversation participants, and it should help the understanding system plan inference system, and the understanding system itself, with plans that the understanding system habitually performs.

Real people in the real world do *not* customarily go through something resembling the current search-based methods used for planning (and plan inference) in all but the most unusual of circumstances. In daily life, people rarely encounter completely new situations–

everything is simply a permutation of things that have been seen before. As a result, people form habits to deal with specific types of problems. The type of situation presenting a problem, along with the types of desired goals, triggers a recognition process that yields the type of solution-plan as a result. The solution-plan is then instantiated in the current situation, in a case-based manner, using specific role variables. As a result, people plan in a fast, habitual manner, that does not require costly searching. Also, their habitual plans are more easily predictable than plans created by searching.

An analogous method could be used to implement planning and plan inference of habitual plans for the computer. It might even be possible to use neural nets to implement this, for speed. In any case, a fast habitual plan system would speed up the disambiguation process for a conversation understanding system.

The Uncertain Non-Deterministic Action (UNDA) system has the job of representing plans with non-deterministic actions, of making decisions with such actions, and of planning with such actions. In addition, the UNDA system must perform plan inference on conversation participant plans that contain non-deterministic actions (known as *contingency plans*). Each plan has a set of possible outcome situations, with an associated probability of occurrence. The system must plan which actions to take, from the starting situation to a set of possible goal situations, so that the probability of obtaining a goal is maximized (using expected value).

The UNDA system is necessary because people make plans with contingencies in them, and people work with actions with uncertain outcomes. The current planning technology is insufficient to be able to represent and cope with such plans (as is discussed in Section 3.1). Thus, the new UNDA technology is required for competant conversation understanding. UNDA and the MU calculus are discussed in Section 12, and in a forthcoming journal paper.

The trajectories system provides a non-plan-based, statistics-based approach towards action, speech act, and utterance prediction. It is quite possible that a prediction method based on statistics will be stronger than one based on logic. The trajectories system will be responsible for predicting possible candidates (with associated probabilities) for what comes next, based on statistical histories. It will contribute results to the causal evidential reasoning system, and also to the utterace and speech-act prediction system. Trajectories are discussed in Section 9.

Both the trajectories system and the UNDA system will have to make use of the multiple possible Action Worlds representation built on top of the ATMS (Assumption-Based Truth Maintenance system). The ATMS by itself can only represent a parallel universe of timeless multiple possible worlds. The Action Worlds representation is needed to be able to represent, in different multiple worlds, both the states before an action takes place, and the nonmonotonic states after an action. Using Action Worlds, it is possible to represent a universe of sequential and parallel multiple possible worlds, with time information. Unfortunately, generality is sacrificed when a nonmonotonic representation is used: it becomes necessary to state *when* an action is performed, in addition to *whether* an action is performed. This requires explicit searching and slows down any planning that can be done with the representation. Version 1 of Action Worlds has been implemented. Future versions will require a more explicit time representation, and the ability to represent uncertain time durations and commencements.

The UNDA system will also need to use the NP Plan Inference System to help read in alternative feature structures, form inference rules, and compile actions. In addition, the

NP system will help the UNDA system perform plan inference. The NP system is described in [Mye90f], [Mye90d] and [Mye89c], and uses the same ATMS that was discussed in the previous Action Worlds section. The ATMS is described in [Mye89b]. The NP system has been implemented to Version 3.3. The ATMS has been implemented to Version 3.0.

The Intentional Agent Simulator attempts to provide an intention-based simulation and prediction of the internal states of the conversation participants. It models the wants, beliefs, intentions, endeavors, and currently executing actions of the conversing agents. The Intentional Agent Simulator must predict what the speaker is going to say next, based on an understanding of his intentions and his opinion of the current situation. This information has the important role of providing the causal evidential reasoning engine with causal links and accurate probabilities based on the current situation, which are needed to perform accurate causal evidential reasoning.

The Intentional Agent Simulator is implemented using a body of theory about intentions and actions that includes the concept of the *ability* to execute a plan [Mye90b], the problems of executing actions that can fail [Mye88b,Mye88a,Mye88c]), and the requirement to build a *contingency-plan* tree and make decisions about the value of actions to be taken when actions have nondeterministic outcomes [Mye90e]. The Simulator is also implemented using the Architecture for General ENTities (AGENT.001), an architecture for building intentional agents that describes what modules and functions are required for implementation. Finally, the Simulator will require the implementation of a Character Traits and Motivation module, that will describe why people want to do something, how much they are motivated to perform something, and how an observer can predict what a person will probably do based on previously known character traits.

Finally, the understanding system modules in general will require implementations of limited-time and limited-resource reasoning and scheduling methods. It will probably be too costly to explore every possible implication of any one particular new fact. Thus, the system will have to make decisions about what kind of processing should be performed while understanding an utterance, which knowledge sources should be given more power, and which processes should be held back because there is not enough time and memory resources. It is expected that this scheduling should be applicable to other automatic interpretation systems besides the understanding system, and that it will probably make use of the UNDA decision-making technology. Further discussion is found in Section 6.

9 Trajectories

Section 7 points out that the main problems in language understanding can be classed as disambiguation problems and prediction problems. But, disambiguation needs a good method of prediction, in order to compare the disambiguation candidates against predicted occurrences. Previous methods of prediction have focussed on logical approaches that predict what "should" happen, but have ignored descriptive representations of what *does* actually happen. A new approach to prediction is required, entailing a new representation methodology. Such an approach is described here. This approach will not replace the old methods used to perform prediction, but will rather augment them.

A new theoretical entity, called a "trajectory", is proposed. A trajectory is the path, labeled with statistical probabilities, that a conversation takes through a particular feature space. Examples of feature spaces include vocabulary, syntactic, semantic, prosodic, illocutionary force, and domain action spaces. The emphasis in a trajectory is on what possibilities follow from the current point, given the previous history and entire context of the current trajectory. Just as when a ball is thrown on a windy day, it is possible to predict approximately where the ball will land, given its flight path; so, given the "flight path" or trajectory of a conversation, it should be possible to roughly predict the course of the next utterances. It is argued that people do this all the time, and that trajectories play an important role in the understanding of conversations.

A trajectory is similar to a syntactic or conversational grammar, in that a trajectory helps determine what is allowed to follow in an utterance or conversation after a given point. However, grammars are composed of binary yes/no rules, that have no probabilities attached. A trajectory has probabilities, and can predict *which* following item is the most likely, out of a set of possibilities. In addition, a conversational grammar rules out all types of following utterances other than those that the grammar describes. However, in a real conversation, it is possible (although unlikely) to follow one type of utterance with any other type. For instance, it is possible to follow a question with another question, or with an order, instead of with an answering statement. Since trajectories work with probabilities, trajectories are able to accept and represent all kinds of transitions, by not ruling any out, but making them highly unlikely.

A trajectory is similar to a script, in that a trajectory determines a series of actions that occur. However, there are several important differences. A script is a specified series of actions designed to accomplish some goal of the performing agent. Scripts are thus *prescriptive*, they specify what the agent *should* do next. Scripts are typically strictly linear, having only one path. If a script is allowed to deviate, it can only do so at special places called *turning points* [SR81, p.76]. Since scripts normally only allow a single specific action to follow the current action, there is no concept of probability, nor which actions are most likely to follow while which actions are unlikely. A trajectory, on the other hand, is *descriptive*–it describes what *actually* happens, not what *should* happen, on the basis of gathered statistics. The goal that the agent has in mind does not have to be important. Trajectories branch at every point–it is always possible to perform more than one next action, or to have more than one next outcome. And, of course, trajectories are designed to predict *which* next actions are most probable, and which are unlikely but doable.

Scripts are extremely similar to Markov chains. However, a Markov chain bases its predictions on only the previous one, two, or N states in a history, and requires representation by a huge $N+1$-dimensional array, whereas a trajectory will be represented by a network, and is based on the entire context of the actions. For instance, trajectories can include information based on such context states as the beliefs and desires of the conversation participants. Since the transition probabilities will vary based on different situational contexts, more information is required than just a single matrix to determine the probabilities. At the same time, since only the transitions that are used will be represented, less actual information is required to be stored in the computer.

Something similar to trajectories is believed to be used by people who are engaged in the learning process. When people learn about the world, it seems that they code their knowledge into trajectories. So, trajectories seem to be a natural representation for knowledge and prediction. It also appears possible that trajectories might be able to be implemented using neural net technology. However, at first, standard software methods will be used.

Trajectories only partially represent the difficult theoretical concept of "coherence". More research is required to determine what coherence is, how coherence is abstracted from trajectories, whether it is needed, and how some trajectories are seen to be coherent while others are noncoherent. One powerful feature of trajectories is that they can represent real-world occurence sequences that are not yet coherent—this is an advantage. However, it appears that trajectories are necessary but not sufficient. They will have to be augmented, or complement exising plan-recognition systems, in order to be powerful enough to be useful in a general understanding system. More research is required.

Trajectories will be used in conversation understanding by contributing to disambiguation in two different ways. First, the prior probabilities provided by the trajectories, when combined with the posterior probabilities abstracted from the evidence of the observations, can be used to compute the most likely interpretation of the observed utterance candidates. Second, the information provided in the trajectories can be used to "fill in" missing information (such as zero anaphoras) in the current utterance. The trajectories should provide reasonable expectations as to what will follow. In these ways, trajectories will make conversation understanding more powerful.

10 Research in Evidential Reasoning

10.1 Language Understanding is an Evidential Reasoning Problem

The language understanding task consists of the following: The system must accept a list of uncertainly observed inputs, which represent possible observations of the speech recognition system. The system must have a representation of world knowledge and contextual knowledge about the previous progress of the conversation. The system must use the possible observations and the knowledge to determine the "correct" meaning of the utterance, that "makes the most sense" for the utterer to have said. This meaning will be represented in beliefs, which will be available to the transfer and generation systems.

An examination of this problem shows that it is actually a form of an evidential reasoning problem. Evidential reasoning is a branch of AI that deals with forming conclusions from uncertain, perhaps conflicting, observed evidence. Uncertain observations are gathered. Each observation is weighted as to how reliable it is, i.e. how certain the observer is that the observation corresponds to reality. (For instance, in the automatic interpretation problem, the observations are the possibly observed utterance candidates, and the evidence is derived from the recognition score associated with each one.) The system also has preconceived opinions, called *a priori probabilities*, about how likely each concept is. (For instance, these would be derived from the world knowledge and the contextual knowledge of the conversation.) Using the observed evidence plus the prior opinions, the evidential reasoning algorithm computes a *degree of belief* in each concept as to how much that concept is believed to be true. The system uses numerical scores, not logical categories, to specify the degree of belief. Since the belief is based on likelihood, it in fact represents the concepts that are the "most likely", or "make the most sense".

Thus, it is seen that language understanding can be classified as an evidential reasoning problem, and that established evidential reasoning mathematical methods should be applied to attack the problem.

10.2 Pearl's Causal Evidential Reasoning System Solves Many Problems

10.2.1 Judea Pearl and "Probabilistic Reasoning"

Many people are doing research in reasoning with probabilities in AI. One of these is Dr. Judea Pearl of UCLA, an excellent mathematician/computer-scientist who has been researching Bayesian inferencing methods and searching for at least ten years. Dr. Pearl is an editor of the AI Journal, and the director of the Cognitive Systems Laboratory at UCLA. He has recently (1988) published a book, "Probabilistic Reasoning in Intelligent Systems", that describes a new mathematical method, called "Causal Evidential Reasoning", for working with probabilistic belief. This method solves many problems that were previously barriers towards useful application of probabilities.

Causal evidential reasoning is a method of representing the degree of belief in concepts that are connected by probabilistic causal relationships. If something normally causes something else, this knowledge can be represented in the system. So, the system consists of nodes representing concepts, which are labeled with a priori probabilities and degrees of belief, along with "causes" links, which are labeled with probabilities (in the form of causal matricies). The probabilities propagate. At any one point in time, the system can report the degree of belief in any particular concept.

The system is able to accept two kinds of input, in the form of observations. The first kind is *certain observations*, where the value of a particular concept is definitely known. The second kind is *uncertain observations*, where the observation provides evidence towards a particular concept, but the knowledge is not definite. It is possible to receive multiple conflicting pieces of evidence.[5] These two cases are handled slightly differently by the method, but the result is the same: the effects of the observation are mathematically incorporated into the belief network in a proper manner. The beliefs of all of the nodes are updated to reflect the evidence obtained from the observation, through the probabilistic causal links. Hence the name, *causal evidential reasoning.*

The methods presented in the book appear to be quite useful. It looks like his methods can be applied towards solving some of the outstanding problems in dialog understanding. Therefore, implementation of a causal evidential reasoning engine has begun. A short discussion is presented in Section 10.5, and preliminary results from a single-parent first version of the engine are presented in Appendix F.

Unfortunately, the system in Pearl's book is difficult to implement. The mathematics and the concepts are quite complicated. Fortunately, Pearl splits his book up into sections that add more and more features to his system, so that understanding and implementation can proceed in stages.

Important relevant sections include philosophy, and a method for building and updating belief networks where each node has only a single causal parent. Next, multiple-parent networks without loops are introduced. After this, new methods for dealing with multiple-parent networks with loops are discussed. Next, Pearl discusses "belief revision", the formation of coherent *sets* of beliefs that together offer the best explanation for a set of evidences. After this, the networks are extended so that other relationships, such as "is-a" hierarchies, can be used in addition to "causes" alone. It will be necessary to build a reasoning engine that can perform all of these capabilities in order to have a system that is

[5]For instance, the top five outputs from the ATR speech recognition system are conflicting evidence from an uncertain observation.

powerful enough to handle the problems encountered by ATR in the dialog understanding task.

Pearl's system solves some specified types of problems with consistency in a belief system that were previously unsolved in other belief and reasoning systems. These are discussed in the following sections.

10.2.2 Predictive Evidence is Different from Diagnostic Evidence

Until now, most evidential reasoning systems have worked with only one kind of evidence. They have used it to form rules both about *prediction* (causes tend to imply effects) and *diagnosis* (effects tend to imply causes). For example, if we know that "the office wants to give advice", then we can predict that "the next utterance is advice" with a stronger belief than usual. At the same time, if the office says "You should send in the form soon, because the rates will go up", this is most probably advice, and it provides diagnostic evidence that "the office wants to give advice".

In most systems, each of these inferences would have to be implemented with a separate rule. These would look like the following:

```
IF 'the office wants to give advice' becomes stronger,
 THEN increase the belief in 'the utterance is advice'.

IF 'the utterance is advice' becomes stronger,
 THEN increase the belief in 'the office wants to give advice'.
```

It is easy to see that this system causes problems. If both rules are implemented, the system will get stuck in an infinite feedback loop, causing it to believe these concepts more and more from only a little bit of evidence. If only one or the other of these rules is implemented, then the system can only reason in one direction, and the logic of the system will be incomplete.

The answer is to propagate predictive evidence completely separately from diagnostic evidence, in an anti-parallel fashion. Only when the belief of a particular concept is desired, should the predictive evidence be combined with the diagnostic evidence to form belief. The system cannot use this combined belief internally to propagate results; it can only report the combined belief as an answer to an external reasoning engine. The system uses one inference rule forwards and backwards to compute both predictive and diagnostic evidence, so there is no reason to enter any rules twice. In this manner, the system correctly solves the problem of reasoning with both predictive and diagnostic evidence, while avoiding problems of infinite feedback.

Observed evidence is treated as a third kind of evidence. Uncertain observations are mathematically similar to diagnostic evidence for a concept.

10.2.3 "Explaining Away" Competing Causes

A particular phenomenon that, until now, has not been accounted for, is the concept of "explaining away" a competing cause to an effect that already has a good explanation. For instance, take the case of the "unagi-da-bun" problem given in Section A. Suppose we have one concept that the person is hungry, and another concept that the person is crazy. Without any other evidence, these concepts take on their priors, i.e. the background probabilities that they are true– for instance, (hungry = 0.1), and (crazy = 0.001). Now suppose that the person says, "Watashi wa unagi da" (lit., I am an eel), which provides increased evidence that the person is hungry (meaning, "I want to eat an eel"), but it also provides evidence that the person is crazy[6]. Perhaps these are increased to (hungry = 0.5), (crazy = 0.01). Next, suppose that we find out that the person really is crazy (crazy = 0.95). We no longer believe that the person is most probably hungry, because we have already found a logical explanation for the person saying "I am an eel". The evidence no longer applies to that node, and the hunger probability drops back to its default level of 0.1. It has been "explained away".

Although this represents evidential reasoning as we would like it to happen, previous systems have not been able to take this into account. Who would believe that learning that a person is crazy implies that he is less hungry than we thought before? There appears to be no connection between these two concepts. In most systems, *adding* evidence can never nonmonotonically *lower* the belief in something else. However, this is exactly what is desired, and this is what Pearl's advanced "belief revision" system can provide.

This capability will be important for understanding ambiguous inputs such as "Sochira wa kaigi jimu-kyoku desu *ka*" and "Sochira wa kaigi jimu-kyoku desu *ga*", where the observed evidence is equally distributed between two or more coherent inputs with different meanings. The observed diagnostic evidence, combined with the causal predictive evidence, will determine which meaning is the most likely.

10.2.4 Sets of Beliefs and Belief Revision

Unfortunately, the highest-probable state value of a state, when taken by itself, is *not* guaranteed to correspond to the highest-probable possible world represented in the system. It is regrettably extremely important to perform so-called "belief revision" and find the highest-probable coherent *set* of beliefs, i.e. a *situation* or a *possible world*, rather than just querying the probabilities of single states. The most probable coherent world will then yield the correct answers for understanding the current conversational scenario. This introduces additional complexities into the mathematics.

10.3 Causal Evidential Reasoning is a Form of Abductive Reasoning

Deductive reasoning starts from concrete observations and known facts, and proceeds to form conclusions based on these observations and facts. Deductive reasoning reasons from explicitly present causes, given a cause-and-effect rule, to effects. Deductive reasoning cannot consistently reason backwards from effects to causes, nor can deductive reasoning reason from unobserved causes to effects. However, deductive reasoning is sound, and should always give the right answers. The current NP system uses deductive reasoning.

Abductive reasoning also starts from concrete observations and known facts, and forms conclusions. However, abductive reasoning reasons from explicitly present effects, given a cause-and-effect rule, to possible or probable causes. Abductive reasoning is used to reason backwards from effects to causes, even when the cause has not been observed. Abductive reasoning is not sound, and does not always necessarily give the right answers. However, it is needed for language understanding.

[6]Or perhaps he is acting in a play, or is a cartoon character.

It can be seen from these definitions that causal evidential reasoning is a form of both (probabilistic) deductive and abductive reasoning. Causal evidential reasoning reasons forward from causes to effects, and it can reason backwards from observed effects to unobserved causes. The belief in a cause is determined by the amount of evidence associated with an effect, along with prior probabilities on both the cause and the cause-and-effect relationship.

Thus, any discussions that apply to language understanding using abductive reasoning, also apply to language understanding using causal evidential reasoning.

It is important to note that both deductive and abductive reasoning require the specification of existing cause-and-effect rules. A third form of reasoning, "inductive reasoning", abstracts a cause-and-effect rule from a given cause and effect. It may be that an eventual effective understanding system might have to discover new nodes and new cause-and-effect relationships on the fly, as it is processing conversations. If only the effect is known (the possibly observed utterance candidates), and both the causes and the cause-and-effect rules have to be solved for by the system, the problem seems extremely difficult. Thus, current efforts will focus on implementing an abductive/deductive reasoning system, while the problem of incorporating inductive reasoning will be reserved to be attacked later.

10.4 Problems with Abductive Reasoning

This section is an abstraction of a paper by Norvig and Wilensky presented at COLING '90 [NW90a] and at an abduction workshop [NW90b]. The paper criticizes language-understanding abstractive reasoning systems. It is important to acknowledge these criticisms, so that any future abductive reasoning system that ATR builds can try to circumvent these problems.

The authors maintain that, when performing abductive reasoning, two different scores are needed for a single probable-cause node. The first score should measure how much it costs to assume the cause, i.e. how probable it is. The second score should measure how well the cause explains the observed effect, i.e. the quality of the explanation. The level of quality of explanation that is required is dependent on the use that the hearer will make of the concepts.

However, in some cases, it is important to maintain ambiguity, i.e. not assign *any* cause to a particular effect. There are at least three cases: (1) There are strong and almost equal votes for two or more causes, so that the system is *unsure* of the cause; (2) The effect concept is so ordinary that the system does not *need* to disambiguate by determining a cause (e.g., is "anata" or "you" singular or plural? There is no need to disambiguate.); and, (3) the cause is so *unbelievable* that the system must reject it. It is important to distinguish between these cases.

The system should be able to detect false and self-contradictory sentences. In addition, the system should have a model of the people who are in the conversation, and their goals; this will help in understanding and disambiguating sentences that are not literally true. If the system only deals with the truth of the utterance itself, and does not think about why the person might be saying such an utterance, then the system will fail to understand such sentences.[7]

If the system is based on probabilities about events and objects in the real world,

[7]A user model is also needed to understand such phenomena as politeness, and why people repeat information that they already have said. The authors do not mention this.

then it can never talk about modals, future actions, hypothetical actions, or imaginary objects. Conversely, if the system is not based on statistics, then its probabilities are not well-grounded.[8]

Deliberate vagueness or ambiguity is also a difficult problem. (For instance, "onegaishimasu" ("if you please") deliberately leaves it vague as to exactly what favor the speaker is requesting.)

The system should not simply choose the concept with the highest probability. In some cases, it is important to look at *why* the system is being asked to decide something–that is, the outcome of the decision matters. In these cases, it is more important to make a *safe* decision than a *correct* decision. Also, as has been discussed, sometimes two explanations have similar, high probabilities, or sometimes the highest explanation has a very low probability.

The system should be able to understand conjunctive causes, and cases where one action actually performs two different functions for the sake of efficiency. (For example, in some cases "hai" means *both* "yes" and "I heard you".) This is difficult for single-explanation systems to do.

It is important to model the process involved in creating an utterance. According to the authors, this starts from a representation, and goes on through intending, believing, directly implying, predicting, and acting to say an utterance. Importantly, a successful model of understanding should be able to work with all of these steps, but should not need to work with them most of the time–some form of default reasoning should negate the need to step through the chain each time. This is still a research issue.

There is a large problem with *coherence* in stories and dialogs that are understood using abduction based only on prior probabilities. Dialogs naturally "cohere" (hang together, e.g. the same topic is mentioned in several different sentences) because people are speaking for a purpose: to communicate a complex conceptual network. However, when abduction is based only on prior probabilities, there is no incentive for the dialog to cohere. It may be more probable that the speaker is speaking about different things. As a result, again because the speaker's model is not taken into account, the system does not understand the dialog well.

All of these points are important criticisms, that must be taken into account when building an abductive understanding system. It is interesting that the authors stress speaker models so much. The Intentional Agent Simulator is being built in an effort to address problems such as these.

10.5 Partial Implementation of a Causal Evidential Reasoning System

As was previously discussed, the schedule for implementing a full causal evidential reasoning system includes the following: (1) Single-parent node tree-network belief updating; (2) Multiple-parent node network belief updating, without loops; (3) Methods for dealing with multiple-parent node networks with loops, but without cycles; (4) "Belief revision", computing most probable coherent *sets* of nodes rather than simply single-node probabilities; (5) Extending causal networks to other relationships, such as `is-a` hierarchies;

[8]This is another version of the "Bayesians vs. subjectivists" discussion on the "true" meaning of probabilities. It can be at least partially solved by an appropriate definition of probabilities as a subjective phenomenon based on objective data.

(6) Integrating causal evidential reasoning with the ATMS's multiple possible worlds; (7) Integrating causal evidential reasoning with the UNDA theory's nondeterministic action representation; (8) Extending the evidential reasoning mathematics with the more powerful MU uncertainty mathematics.

A preliminary version of the first item in this schedule, an engine that performs probabilistic belief updating (both *deductive* and *abductive* reasoning) on tree-networks with single-parent nodes, has been implemented. The system is named CAER, for CAusal Evidential Reasoning. A preliminary demonstration of the capabilities of the CAER reasoning engine is presented in Appendix F.

10.6 Remaining Problems with Pearl's Causal Evidential Reasoning System

Even though causal evidential reasoning offers a powerful tool for deductive and abductive inference that will be useful for dialog understanding, there are still some problems with it. The most obvious problems are discussed in the following subsections.

10.6.1 Time, and Instances vs. Prototypes

There are two related problems that have always caused difficulty with belief systems. These are the problems of time, and instances vs. prototypes. Pearl's system also seems to have these problems.

Pearl's system, like many belief systems, has no representation for time. In effect, everything either happens "at the same time", or at an unspecified time. If it is assumed that simple causality only operates forward in time or simultaneously, then at least this can form a partial temporal order on the beliefs. However, this assumption breaks down when agents are taken into consideration–something that *might* happen in the future can cause an agent to perform an action in the present. The result of these two problems is that the system is unable to represent causal cycles and feedback loops. If a first thing causes a second thing, and the second thing causes a different instance (or the same instance) of the first thing to occur, and so forth in a cycle, the system is unable to represent this. Further research will be required to determine how frequent causal loops are, and how the system can be modified to be able to represent them properly.

10.6.2 Noncausal Relationships

Causal evidential reasoning is designed specifically to deal with implications and causes. However, many important relationships, such as `is-a` or `has-part`, are non-causal in nature. Therefore, the philosophical theory behind causal evidential reasoning does not allow the mathematics to be transferred over to these relationships and still get correct results. This is a serious problem for integrating causal action knowledge with ordinary declarative knowledge.

10.6.3 Action and Nondeterministic Outcomes

Pearl's system has no way to represent state change nor dynamic action. Everything has happened already, and the problem is to solve for what is believed about a *static* world. It is obvious that this formalism is not powerful enough to reason about plans and actions as they are happening, nor to perform plan inference by reasoning about different possibilities as to what might happen in a *dynamic* world. It will be necessary to augment Pearl's representation system. Approaches to this problem are discussed in Section 10.8.

10.6.4 Where Do the Nodes Come From?

Causal evidential reasoning is designed to perform evidential-propagation calculations on an *existing* network of concept nodes and causal relationships. It is thus necessary to specify ahead of time all concepts that hypothetically could be required by the system. Using present technology, these will have to be prespecified by hand, by the system designer. However, at some point it will be necessary to build an automatic learning system that can dynamically create original concepts as required, at run-time. The current system design cannot create new concepts; this must be done by an outside system.

10.6.5 Where Do the Causal Arcs Come From?

The causal arcs that connect concepts together must come from a model of causality. Unfortunately, the causal network and the models will have to be constructed *dynamically*, to reflect an evolving conversational situation that will be unknown ahead of time. Fortunately, there is a facility for specifying such models in a dynamic fashion. The IAM multi-agent situation simulator (Section 11.6), coupled with the AGENT.001 intentional agent mind simulator (Section 11.5), will be able to predice the possible actions of agents and specify the causal relationships between concepts.

10.6.6 Where Do the Probabilities Come From?

Obtaining realistic conditional probabilities is an extremely important problem. These will be obtained from:

1. Statistics gathered on actual conversations.
2. Trajectory information.
3. Case-based (general-specific) reasoning applied to outputs from the IAM and AGENT.001 simulators.
4. Other sources, to be researched.

10.7 Other Objections to Abductive Reasoning Systems

Current abductive reasoning technology can only offer existing concepts, or perhaps new combinations of existing concepts, as antecedent hypothises for consequent explanation. However, people use "abductive reasoning" in a much more powerful way–dynamically creating new concepts to explain results that previously had no antecedents. People are very good at "finding possible explanations" for observed phenomena, explanations that are based on actions, desires, and intentions, and that are completely new. Perhaps they use case-based reasoning to instantiate possible general explanations for the specific case under consideration.

People can also perform "inductive reasoning"–generating causal rules from single instances of cause-effect observations. But this only happens when previous causal knowledge is insufficient to describe the current observation. If a previous explanatory rule can be invoked, there is no need to learn. However, apparently the explanation rule must be based on some mechanism, as an instance/class relationship. When a mechanism can be guessed at (perhaps using abduction?), the inducted rule becomes significantly stronger. Adults also have a "gullibility factor" or "reality check" that stops them from learning things that are too unbelievable. Induction appears to be similar to learning actions: there is a need to learn the capability conditions, and the possible outcome effects. Classes of explanation rules implies the existence of classes of actions or mechanisms. The resulting picture of the requirements for a good inductive reasoning system is complex.

In any case, the type of reasoning required for actually reasoning about real-life problems, such as realistic conversation understanding, appears to be more powerful than the type of reasoning that current abductive-reasoning technology offers. Even after a new abductive reasoning system has been implemented, more research will be required.

10.8 Extensions to Pearl's System

There are a number of deficiencies in the Causal Evidential Reasoning System as it has been defined by Pearl. Given the current state of expertise at ATR, it appears possible to extend his methods, and make some original contributions to the state-of-the-art.

10.8.1 ATMS Implementation

Apparently, Pearl's system uses a single-level (flat) network, and does not take advantage of "possible worlds". This causes inconsistency problems, especially when applied to conjunctions of exclusive disjunctions.

For instance, suppose that there is half a chance (50%) that the questioner's company will send the questioner to the conference, while there is half a chance (50%) that they will send someone else (an exclusive disjunction). Assume that there is another rule that says that if two people both come from the same company, but there is only one conference ticket, then there is a problem (a conjunction). Apparently, Pearl's system would give the answer that 25% of the time (50% x 50%) there will be a problem with two people coming, while 25% of the time no one will come at all! Pearl apparently solves this problem by not allowing conjunctive causes–that is, his system can't represent such a situation in the first place. However, this problem can be solved correctly by using an ATMS to represent the two mutually exclusive possible worlds where either one person comes or the other person comes. Since the two possible worlds never (incorrectly) overlap, there are no problems with inconsistencies. Augmenting a causal evidential reasoning system by reimplementing it using an ATMS appears to be a relatively straightforward task.

10.8.2 MU Calculus Uncertainties

Pearl's system apparently only uses first-order certain probabilities for computation. It appears to be a relatively simple matter to extend his methods by using the MU calculus, so that evidential reasoning can be performed with uncertain second-order probabilities and/or fuzzy logic. This would permit a belief network to work with such useful uncertainty representations.

10.8.3 UNDA Non-deterministic Actions

As was previously discussed, Pearl's system cannot represent actions, especially actions with nondeterministic outcomes. It appears to be a relatively simple matter to combine the UNDA representation method with Pearl's Causal Evidential Reasoning method, to come up with a system that can reason with actions. This does require conjunctive causes, however. The preconditions plus the choice to act will cause the action to take place; the action occuring will cause one of the outcome situations to occur. The resulting system should be able to reason about beliefs about actions, and evidences of actions. However, it will still be necessary to augment this with a temporal representation system.

11 Research in Intentions and Intentional Agent Simulation

This section presents research progress that has been made in the refinement of existing: theory of intentions; model of an intentional agent architecture designed to implement the theory (known as AGENT.001); and multiple intentional agent simulator (known as IAM). Research on intentions is significant for building an understanding system that can disambiguate utterances well.

The theory fixes several problems in existing theories of intention. In order to explain some of the theory of intentions, it is first useful to summarize the other leading research in the field. This has been conducted by the philosopher Michael Bratman [Bra87,Bra90], and the computer scientists Philip Cohen and Hector Levesque [CL86,CL87,CL90].

This section starts with a summary of Bratman's theory. Next, a summary of Cohen and Levesque's theory is presented. After this comes a summary of the problems and errors in the previous two theories. Next, a summary of the main points of an original theory of intentions is presented, that answers these problems. All of these sections are summaries only, as the full theories are too long to discuss in detail. After this, the AGENT.001 agent simulator is briefly discussed. Finally, the IAM situation simulator is briefly discussed.

11.1 Summary of Bratman's Theory of Intentions

There are at least three kinds of intentions (1a, 1b, and 2a):

1. Intention as a Description of Action
    (a) Intentional Action (Acting Intentionally.) The agent performs the action "on purpose" (as opposed to "by accident").
    (b) Acting With An Intention. The agent performs the action in order to accomplish a specific goal.
2. Intention as a Description of the Mind
    (a) Intending To Act. The internal state of having performance of the action as an immediate goal.

Intending to Act can be further subdivided into two categories:

1. Present Intentions

2. Future-Directed Intentions

"Having a plan" can be divided into two categories:

1. Knowing *how* to do something.

2. *Intending* to do something.

Future-directed intentions are new, and need to be justified. There are three apparent problems with future-directed intentions (the "*trilemma*"):

1. Action at a Distance. How can presently-held intentions affect future actions, which are separated in time?

2. Irrevocable. Future intentions appear to be irrevocable: they persist until they turn into present intentions. But, an agent should be able to call back his intentions.

3. Waste of Time. An ideal agent might be able to decide everything in the present. So deciding things for the future is a waste of time.

These apparent problems are answered with the following arguments:

1. No Action at a Distance. Future-directed intentions become present-directed intentions. In this way, future-directed intentions can indirectly cause action.

2. Not Irrevocable.

3. Coordination and Limited Reasoning. Intentions serve to coordinate the actions of the agent, both with other actions of the agent and with actions of other agents. Also, intentions help time- and resource-limited agents pre-plan their behavior. Intentions are useful and thus not a waste of time.

Intentions have the following *Three Functional Roles*:

1. Pose Problems for Means-End Reasoning. We often first form an intention as to *what* we want to do, then we consider *how* to do it. A prior intention thus acts as a *standard of relevance* for measuring proposed optional actions against.

2. Filter of Admissibility: Constrain Other Intentions

3. Control Behavior: Induce Endeavoring

They also have the following aspects:

1. Intentional plans are *partial.* Intended ends can be fixed, while planning the means can be deferred until later.

2. Intentional plans are *hierarchical.* Intended ends can be fixed, while intended means get debated.

3. Intentional plans must be *consistent.* Intentions cannot be inconsistent with the agent's beliefs, nor can they be internally inconsistent.

4. Intentional plans must get filled in with means as soon as possible, or suffer from *means-end incoherence.*

5. Intentions are *stable*. They "resist" being reconsidered.

6. Intentions are "*conduct-controlling pro-attitudes*", not merely "*potential conduct-influencer pro-attitudes*". Intentions directly cause actions.

This flexibility allows the world to change while high-level intentions stay the same.

Intentions are based on binary "*flat-out belief*" in concepts.

Utility-based and decision-theoretic intention theories ignore future-directed intentions and are therefore incomplete.

What one *expects* to happen must be different from what one *intends* to happen. One cannot intend to do something one expects not to happen. Conversely, if one intends to do something, then normally one expects it to happen.

The Problem of the *Package Deal*: I may expect to bring about something that I do not intend to do. If I decide to do an action that results in an undesirable side-effect action, then I must intend to get a scenario in which both the action and the side-effect take place. Therefore I must intend to do both the action and the side-effect. Therefore I must intend to do the action, and I must also intend to do the side-effect. But, how can I intend to do the side-effect when I don't want it in the first place? I don't *really* intend to do the side-effect, because it does not pose problems, constrain other intentions, nor induce specific endeavoring. (Example: Bombing an enemy factory also undesirably kills children in a nearby school.)

Principles of the Package Deal:

1. Holistic Conclusion. If I know that an action results in a side-effect, and I take this side-effect into account when I decide whether to do that action, and if I am rational, then I will have "concluded in favor of" a scenario in which both the action and the side-effect happen.

2. Holistic Choice. "Concluding in favor of" a scenario means *choosing* that scenario.

3. Choice-Intention. Choosing to do a series of actions means *intending* to do that series of actions.

4. Intention Division. If I intend to do actions A and B, and I know that A and B are each within my control, and if I am rational, then I will intend to do action A and also intend to do action B.

If the Package Deal is incorrect, then one of these principles must be incorrect.

Discussion of the Package Deal principles:

1. The Holistic Conclusion principle is built on being "clear-headed" and "intellectually honest". It cannot be rejected.

2. It seems possible to argue that merely *evaluating* a situation is not the same as *choosing* a situation. However, this is wrong. Holistic Choice follows from the Holistic Conclusion arguments. The agent must choose something; this choice comes from clearheadedly evaluating and concluding in favor of a scenario. Rejecting Holistic Choice is wrong.

3. Choice is not the same thing as intention. The Choice-Intention principle should somehow be rejected. It is necessary to choose holistic scenarios, but it is not necessary to intend everything in a scenario. Conversely, intentions require further reasoning and action, whereas a choice does not require this.

4. Rejecting Intentional Division goes against common sense. Obviously, if I intend to do A and B, then I intend to do A and I intend to do B. Intentional Division cannot be rejected.

11.2 Cohen and Levesque's Theory of Intentions

Cohen and Levesque follow Bratman's Three Functional Roles. In addition, the following basic statements are proposed. If an agent intends to achieve state S, then:

1. The agent believes S is possible.

2. The agent does not believe that it will *not* achieve S. (The agent does not expect to fail.)

3. Sometimes the agent believes that it will achieve S. (Sometimes the agent expects to succeed.)

4. Agents do not have to intend all the expected side-effects of their intentions.

5. Intention is choice plus commitment.

6. Chosen desires (intentions) must be consistent.

7. Agents will replan if an endeavored intention fails.

"Persistent goals" are kept as long as the conjunction of certain conditions hold. If any condition fails to hold, the goal must be dropped. These include:

1. The goal is achievable.

2. The goal has not yet been achieved.

Cohen and Levesque build a first-order approximation to a theory of intentions. They do not try to build a completely correct theory, but rather one full of idealizations that attempts to cover frequent cases. They explicitly acknowledge this approach.

Cohen and Levesque take a logical, objective viewpoint. Everything, including beliefs, goals, and intentions, is boolean. There is some outside objective observer who can definitely tell the truth or falsity of any proposition. They base their theory of intentions on a possible-worlds model. The present is assumed to be a single, known world, that has a single, fixed, predetermined future. However, agents might not know about the actual present or future, and so they (in some sense) believe in multiple possible future worlds. The theory is based on the BEL() operator for belief. Agents also somehow "desire" and "choose" possible future worlds, although the mechanism for this is not specified; any proposition that is true in *all* chosen worlds is a GOAL(), another primitive operator in the theory.[9]

1. The operator HAPPENS(a) is defined as something that is about to happen in a world. The operator DONE(a) is defined as something that already has happened in a world. The operator $\diamond$ (Eventually) is defined as something that will happen in a world. The operator $\Box$ (Always) is defined as something that always has and will happen in a world.

2. Agents are immortal.

3. Agents have unlimited, infinitely fast computational powers. If it is possible for an agent to come to a conclusion, it has done so.

4. Beliefs are closed under implication. Since agents are infinitely fast, an agent believes all the implications of its beliefs.

5. Goals are closed under implication. If an agent has something as a goal, and that something implies a consequence, then the agent has the consequence as a goal as well.

6. Self-knowledge of beliefs. Agents always know when they believe something.

7. Knowledge of Ignorance. Agents always know when they don't believe something.

8. Consistent Beliefs. Agents cannot consider inconsistent or conflicting possible worlds under the BEL operator.

9. Omniscience of Theorems. Agents know all true theorems.

10. Knowledge is True Belief. Since there can exist an objective outside observer that can determine whether something is true or not, it makes sense to define an operator KNOW that corresponds to an agent BELieving something that is actually true.

11. Competence is Belief Implying Reality. Again, because of the objective observer, it makes sense to define an operator COMPETENT that corresponds to when an agent BELieves something, it automatically KNOWs it, and that something is automatically true.

12. Agents have an unlimited memory. They never forget anything.

13. Knowledge of Actions. Agents know (and are competent in) every primitive action that they have ever done. An agent never does anything primitive that it doesn't know about.

[9]A strict interpretation of their logic foundations, including the B() and G() accessibility operators, is more complex than this. For instance, B(p) operates over possible worlds and involves non-belief of facts incompatable with p, i.e. $B(p) = \sim Believe(\sim p)$, an unusual interpretation that allows belief in imaginary concepts. G() suffers from similar problems. However, since Cohen and Levesque use only BEL() and GOAL(), and ignore B() and G) in their paper, this summary will do the same.

14. Self-belief of next actions. Whenever an agent believes that is about to do *any* thing next, that agent always believes that it is about to perform some specific thing. In other words, an agent about to act is never at a loss for actions. Agents always know what they are going to do next, unless they are not doing anything (in their opinion).

15. Achievement Goals (A-GOALs). If an agent definitely believes that something is currently false, and the agent has a GOAL that that thing should eventually be true, then the agent has an Achievement Goal (A-GOAL) to have that thing happen. "Maintenance goals" correspond to cases when the agent definitely believes that something is currently true. [The cases in which the agent has no opinion on whether the desired thing is currently true or not, or in which the agent finds out that its belief in the desired thing's current truth, are not discussed.]

16. No Permanent Achievement Goals. Agents must eventually drop all achievement goals. Either the goals are achieved, in which case they should somehow be dropped, or they are not achieved, in which case they should somehow be dropped. Agents "should" somehow start working to achieve their goals, or somehow stop trying (perhaps in the case that the goal seems too hard). An agent cannot have an achievement goal that it keeps throughout its life. [Although this is stated as a requirement in the mathematics, there is no discussion of the mechanism used to enforce it.]

17. Anything that an agent definitely believes is true, must be a GOAL. Agents have as GOALs everything that is true.

18. Agents think that everything that they are going to do next must be a GOAL. They never think that their next action might not be a GOAL. If they are about to do something bad, the set of GOALs is adjusted to include that bad thing.

19. Persistent Goals (P-GOALs). A Persistent Goal is an Achievement Goal that the agent does not give up until it thinks the goal has been satisfied, or until it thinks the goal will never be true.[10]

20. As long as the side-effect is not currently true, P-GOALs are closed under theorem implication. That is, if an agent has a P-GOAL to perform an action, and if the action has a side-effect that is not currently true, and the action always implies the side-effect, then the agent has a P-GOAL to obtain the side-effect.

21. Persistent Goals imply Success. If an agent has a P-GOAL to perform an action that it is COMPETENT in, and the agent does not ever believe that the action will *not* occur, then the action must eventually occur. Although this is proved by their logic, how it happens is not discussed. The proof basically depends on assuming that it must happen, so therefore it will happen.

22. Persistent Goals imply Belief in Success. Given the same preconditions, if an agent has a persistent goal, the agent believes that the agent will eventually perform the action. Perpetual failure is not considered.

23. Intending Actions (INTEND$_1$). Intending an action is defined as an agent having a persistent goal towards first having believed that the action was just about to happen,

[10]There is a typographical error in the authors' Definition 8 in [CL90]; the third $\wedge$ should be a $\vee$.

and then next having done the action. Note that the definition of intending an action is based on PGOAL, and inherits all the problems from this definition.[11]

24. Intending States (INTEND$_2$). Intending a state (actually, a situation) is defined as having a persistent goal towards the agent having believed that: there exists some (sequence of) events that the agent has performed, after which the state happened; this persistent goal also includes the agent at the same point in the past *not* having a goal such that it does not happen that there is a (possibly different) series of actions, after which the state is true; also included in the persistent goal is the fact that the different series of actions happens immediately after this belief and this non-goal, after which the state occurs.

    Since a theory of causality is never defined, there is no notion of the agent *causing* the state to happen. As long as the agent performs some series of actions, after which the state is true, the intention will be valid. The actions may be completely unrelated. In addition, how the agent comes to believe what is defined (i.e., where the planned sequence of events comes from) is not specified.

25. The Screen of Admissibility. If an agent intends$_1$ to perform an action, and the agent always believes that after having done a different action, it is impossible to finish doing the intended action, then the agent cannot intend$_1$ to perform the different action followed by the intended action.

26. Tracking Success. If an agent has at some point intended to perform an action and at the same time believed that it was about to perform the action, after which some event happened; and the agent believes that the agent has not done the action; and the agent does not now believe that the agent has never and will never have done the action; then the agent now intends$_1$ to do the action.

27. Persistent Relativized Goals (P-R-GOALs). A persistent relativized goal is defined as an achievement goal AGOAL that the agent gives up after it either believes the goal state to be true, or it believes the goal state to always be false, or it believes that the relative precondition is currently false.

28. Relativized Intentions. INTEND$_1$ and INTEND$_2$ can be defined with a relative condition, such that the intention's goal is dropped after the condition is currently believed to be false.

11.3 Problems and Errors in the Previous Theories

11.3.1 Problems with Bratman's Theory

1. Revocable vs. Irrevocable. Irrevocable or "stable" intentions can pose problems for reasoning, act as a filter for other intentions, and control behavior. If intentions are revokable, they cannot do these things, e.g. they cannot act as a filter to constrain later intentions, all the time. Bratman does not reconcile this problem, and gives no mechanism for specifying when an intention can be reconsidered or nominated for revocation.

[11]There is a typographical error in the authors' Definition 9. HAPPENS(a) should be HAPPENS(x a).

2. Choice is not Intention. When Bratman rejects the fact that choosing to perform an action directly forms an intention to perform the action, he does not put anything in its place. He merely says that a choice will somehow generate "some intentions or others", but he admits he does not understand how this would work.

3. Intentional plans must be consistent. There is no logical constraint that says that intentions cannot be inconsistent. If this statement is about a single possible world, it is a desired guideline only, not a requirement. When an intention is formed from one mind in a society of minds, this statement may be false. If this statement is about contingency plans for uncertain non-deterministic actions, which span multiple possible worlds, it is obviously false, as actions based on different outcomes will in general be mutually inconsistent.

4. The known-means guarantee. Bratman's theory offers no guarantee that the means to obtain a goal have been filled in by the time it is time to attempt to obtain the goal.

11.3.2 Problems with Cohen and Levesque's Theory

1. The mechanism for desire and for choosing future worlds is never specified. This is a serious shortcoming, especially since their basic GOAL() operator depends on these concepts. They specifically do not define a WANT operator. There is no discussion of where desires come from, nor when they change; thus, their theory does not specify where GOALs come from, nor when a GOAL gets withdrawn due to changing desire. Since A-GOALs, P-GOALs, P-R-GOALs, and INTENDs are based on GOALs, this is a very serious problem.

2. The mechanism for an agent to learn that its belief about the doability of an action is wrong, is never mentioned. Agents somehow magically change their opinions and drop their goals if they try to do something and it doesn't work after some period. Related to this is the problem of how to stop procrastination: agents somehow magically decide that they have "put things off" for enough time, and start performing their intended actions. However, how or why this happens is never mentioned. These are more major holes in the theory.

3. Nondeterministic outcomes are not represented. When an action occurs, its effects *must* occur. The universe is entirely deterministic, and does not allow agents to have real choices. This is obviously too strong.

4. The concepts of agents as immortal, unlimited-computation, non-forgetting entities, with beliefs that are consistent, that contain all theorems, and that are closed under implication, are obviously grossly inaccurate. Self-knowledge of belief, knowledge of ignorance, knowledge of actions, self-belief of next actions, and no permanent achievement goals are all debatable concepts.

5. The KNOW operator, which depends on an objective observer, is useless in a subjective world. Since an agent can never represent "p", but only, "I believe p", an agent can never derive the fact that it KNOWs something and use that fact in its calculations. Correspondingly, the COMPETENT operator can never be derived by an agent. Both COMPETENT and KNOW are dangerous concepts for an agent to

use, as they represent causality backwards. There is a tendency for an agent to try and bring about a concept "p" in the world by proving in its mind that it BELieves "p" and KNOWs "p", or that it BELieves "p" and is COMPETENT in "p", instead of producing "p" in the world and then BELieving "p" from observation.

6. There is no consideration of the significant problem of how to resolve inconsistent intentions. Intentions are defined as being consistent (as are goals and beliefs). The problem of how to make conflicting intentions become consistent is completely ignored, as is the possibility of maintaining and working with inconsistent intentions.

7. The concept of GOAL() is quite strange. For instance, in all cases `GOAL(The sun rises)` or `GOAL(The moon exists)` is true. However, if an agent definitely wants to go to Tahiti or Hawaii, but can't do both, both `GOAL(Go to Tahiti)` and `GOAL(Go to Hawaii)` are *false*. Under certain conditions, it is easily possible to have a GOAL() set that is completely empty. Since GOAL() is tied to worlds, and not states, there is no true concept of wanting a particular state or a situation.

8. Having GOAL be closed under implication implies that agents have the goal of obtaining undesirable side-effects. For instance, `GOAL(Have-Cavity-Filled)` and `BEL(Have-Cavity-Filled`$\Rightarrow$`Pain)` implies `GOAL(Pain)`.

9. The concept of "eventually" corresponds to the concept of "sometime future intentions" discussed below, but does not capture the immediacy of "current intentions". Since agents are immortal, there is no *drive* to get something done *soon*. Intentions can hang around indefinitely without being serviced, as long as the agent believes that they eventually will happen.

10. Allowing GOAL() to only operate over worlds that are subsets of BEL() worlds is much too strong, especially when the strong consistency of their BEL() operator is taken into account. People can obviously have goals of events that they don't believe will happen; they would not buy lottery tickets otherwise. The theory even excludes GOAL()s of things that *might* not happen.

11.4 A Different Theory of Intentions

1. Three axes of intentions:

   - Action vs. Achieved State: Intending to perform an action, and intending to achieve a resultant state, are different in kind. One can only action-intend to perform an action oneself; one can never action-intend to have another perform an action. (E.g., "I intend to have him call you," is a state-intention.)
   - Self vs. Other: Self-intentions are used to originate one's own actions. Self-intentions are descriptions of one's own mind, or operators in one's own mind. Perceived intentions of another are used to *predict* the actions of the other. In the case of cooperative agents, other-intentions can be used for coordination. Other-intentions are descriptions of another agent's actions and apparent mind. The intentions are a description of the mind and the actions together, not separately.
   - Sometime-future vs. Current vs. Presently Active Intentions: Sometime-future intentions encode actions that have been decided on, but have not been strongly

committed to, as they do not cause scheduling. Current intentions cause scheduling, and generate presently-active intentions. Presently-active intentions generate actions.

2. All mathematics represents *subjective* beliefs or states held by an agent. All beliefs are approximate *models* of reality made by an agent. Apart from trivial cases of computer universes, there is no outside objective observer who can definitely tell the truth or falsity of any proposition.

3. The present (reality) is in general unknowable in detail, and must often be modeled with multiple possible worlds. The future is in general nondeterministic, and must be modeled with multiple possible worlds.

4. The principle of Holistic Conclusion is correct. Relevant noticed effects of a situation must be taken into account by a rational agent when evaluating the situation. Note that some effects may be nondeterministic, and may not be expected to occur.

5. It is possible to evaluate a situation's outcomes without commitment to action. This occurs while exploring hypothetical actions.

6. It is possible to evaluate a situation's outcomes while being seriously committed to action, and yet still not choose the highest outcome. It is *not* necessary to choose some action. This is connected to the theory of *alternative meta-choices*, and occurs in situations where the highest outcome does not exceed some desireability threshhold. Some of the alternatives involve seeking more information, re-evaluating the situations, or doing nothing. This weakens the principle of holistic choice.

7. Deciding and then choosing to perform an action *by definition* forms an intention to perform that action. It is a volitional choice to *intend to perform an action.* (However, current intentions are different from sometime-future intentions. It is possible to choose to intend to do an action, and still not actively plan to perform that action.)

8. Needs are externally describable basic requirements of an organism.

9. Drives are hard-wired mechanisms that, in a successful organism, are based on sensed signals indicating needs. Drives cause desires or wants.

10. Desires are abstract cravings based on drives, customary wanting habits, the current situation, likes, dislikes, and other factors.

11. Wants are concrete cravings based on desires, requests/orders, morals, and other factors.

12. Intentions are quantitative, not binary.

13. Expectations are quantitative, not binary.

14. Intentions and expectations form conditional trees, not linear sequences.

15. Intentions and expectations are interrelated. They are different parts of the same tree. However, it is *not* necessary that if one intends to do something, then one expects it to happen. One expects *something* to happen. The expected outcome may even be unspecified.

16. Outcome expectation strengths can have measureable uncertainties (probabilities).

17. Goals are states or situations, not worlds. Goals can be inconsistent.

18. Goal states have values (which may be uncertain).

19. Goal situations are composed of goal states and non-desired states.

20. The expected value of a goal situation is a function of its expectation and its value.

21. The expected utility of a goal situation is a function of its expected value.

22. Decisions, including decisions to intend to do something, can be based on comparison of expected utility. Since these decisions take into account currently-scheduled future intentions, they are *not* incomplete.

23. Action decisions must be compared against other possible actions. It is not possible to decide whether or not to do something out of context–one must always compare the action against what else one could be doing. **Corollaries:**

    - It is possible to intend to do something bad. This happens when all other possible outcomes are worse.
    - It is possible to not intend to do something incredibly good. This happens when an alternative, mutually exclusive outcome is even better.

24. It is possible to intend to perform something one doesn't expect to perform. These arguments are explained further in [Mye88b].

    - Expected utility argument. Even though the expectation may be almost nil, one might not have anything better to do.
    - Modeling error argument. If one is wrong about the probabilities, it can be worth it.

25. Decisions are based on comparison of expected outcome *situations*, not just outcome states.

26. Expected outcome situations and expectation uncertainties are derived from beliefs. The set of possible expected outcome situations is fixed, and cannot be modified without modifying the set of beliefs. In other words, a rational agent should not change its expectations just because it wants them to be different. The expectations are derived honestly from beliefs, and must not be changed arbitrarily.

27. The intention to obtain a derived-goal-situation is different, and is derived from, the intention to obtain a goal-state.

    - Goal-states come from wants.

- Derived-goal-situations come from goal-states, after evaluating the global situation.
- Goal-states are state-intentions, and should not be confused with action-intentions.
- Derived-goal-situations are pursued as long as they are the best alternative.
- Derived-goal-situations can be reconsidered if the global situation changes and more possible outcomes appear.
- Goal-states will not be reconsidered at this level of reasoning.
- Side-effects are probabilistic, and may be not expected to happen. It is possible to perform an action, and expect to achieve (A), or possibly (A and B). In this case, the specific outcome situation (A and B) is not chosen as such. Thus, the "principle of holistic conclusion" is weakened.
- Intentional division is a false concept. A derived-goal-situation corresponds to (A and B); a goal-state corresponds to A. It is common to intend to achieve (A and B), while not intending to achieve (B). Even if A and B are each within the agent's control, if the situation (A and B) is chosen (by derivation from the goal to A), then the agent will work towards that situation until a better alternative is offered. The "problem of the package deal" comes from misunderstanding how intentional action works; it is not a problem. [12]

28. Remaining areas of research:

    - Intentionally NOT performing an action.
    - Group Intentions.

29. Motivation is an important part of the intentional action cycle, and must be studied.

30. This theory covers intentional actions only. It does not say anything about nonintentional actions, including at least:

    (a) Unintentional consequential actions and effects.

    (b) Renaming performed actions or unintentionally creating superactions out of performed subactions.

    (c) Random actions.

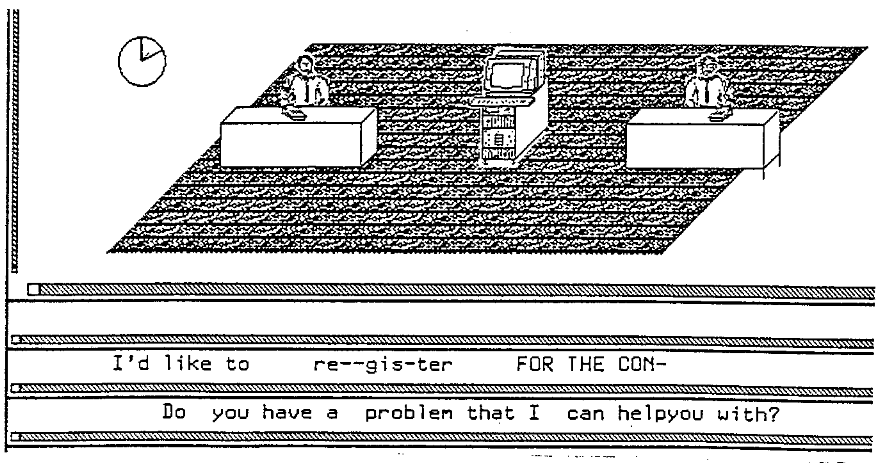
    (d) Nonvolitional actions.

    (e) Emotionally-driven actions and responses.

    (f) Habitual actions.

    (g) Action without an intention (doing things "by accident").

[12]The package deal states that `Intend-Derived-Situation(A∧B) = Intend-State(A) ∧ Intend-State(B)`. This is obviously not necessarily true. Even if only a simple `Intend()` operator is used, intend() is a modal operator, and there is no reason why the $\wedge$ operator must be able to be associated out of the scope of the operator.



I'd like to re--gis-ter FOR THE CON-

Do you have a problem that I can helpyou with?

Figure 2: Example Demonstrating the AGENT and IAM Simulators

11.5 The Intentional Agent Architecture AGENT.001

AGENT.001 stands for Architecture for General ENTities, a full design for agents that operate using beliefs, desires, and intentions. The AGENT.001 design embodies the theory specified in the previous section. Research is continuing at a low level on extending the design, implementing the full design in software, and testing the theory. A limited version of AGENT.001 was used to implement the minds of the discussing agents presented in a paper on handling interruptions [Mye90c]. An agent had the intention of speaking a message, but stopped speaking if another agent forcefully interrupted it. The agent then resumed executing its intention when it was able to do so and still be polite. An example from the paper is reproduced in Figure 2. Research on AGENT.001 was reported at an invited talk at Kyuushuu Institute of Technology [Mye90g]. The current design for AGENT.001 is presented in Appendix E.

It is anticipated that the AGENT.001 intentional agent design will be useful in testing out and proving a realistic theory of intentions. It will also be useful in understanding and predicting the intentional actions of speakers, thus helping the interpreting telephone system to understand the intent behind utterances and to interpret them better. The intentional agent architecture will play a key role in dynamically specifying the nodes and causal connections required by the evidential reasoning system for the disambiguation system presented in Section 7.

11.6 The Intentional Agent Moderator (IAM) Situation Simulator

Research is continuing at a low level on extending the IAM intentional-agent situation simulator. The IAM is a system that supports pseudo-real-time, pseudo-parallel simulation of multiple limited-resource intentional agents interacting in a situation simultaneously. The agent's minds are separate from the world and from their bodies. Limitations can be placed on sensing, computation, and actuation capabilities, to make the simulation realistic. For instance, agents require time to think about concepts, and sometimes agents do not perceive messages completely. The agents can be mistaken about actions in the world, which forces the agents to be careful and to verify what they are doing. The system handles agent character initialization, and takes care of simulating the agents as they interact.

The IAM simulator was used to simulate two humans and a computer agent talking simultaneously, listening to each other, and interrupting each other [Mye90c] (again, see Figure 2). Each agent had to talk, listen, and think at the same time. Although the AGENT.001 simulator handles what happens inside a single mind, the IAM simulator handles time, distance, communication between multiple agents, and what happens in the "real world". The simulators thus are quite different.

It is anticipated that the IAM simulator will be useful in predicting the next utterances of speakers, and in initializing the probabilities for causal relationships required for evidential reasoning. The IAM simulator is required to understand situations in which more than one speaker (e.g., a human and a computer) are talking. It is also useful for testing out "real-time" conversations, where the human might become bored, might not hear or correctly understand an utterance made by the computer, or might not wait his turn.

12 Research in Uncertain Non-Deterministic Actions (UNDA)

12.1 Description

The major work of this year has been basic research in a theory of working with Uncertain Non-Deterministic Actions (UNDA) [Mye90e]. A nondeterministic action is one that has several different possible outcomes, only one of which will come true. An uncertain nondeterministic action is a nondeterministic action where the possible outcomes are labeled with "uncertainties". An "uncertainty" is a new mathematical entity that subsumes probabilities, fuzzy numbers, second-order probabilities, Dempster-Shafer evidences, interval probabilities, and complete uncertainty. Uncertainties are used to form "uncertain variables", analogous to "random variables" but more powerful. Uncertain variables are used to represent "uncertain quantities" of real-world measurable items. A new mathematics, "the MU calculus", has been developed to reason with uncertainties. A representation and computation theory, called "UNDA", has been developed to represent and reason with uncertain nondeterministic actions. Using UNDA, it is possible to make decisions about which actions should be chosen, and also to plan new series of actions. UNDA thus offers an original unified view of planning theory and decision theory under uncertainty.

The following are two example of the type of problems that UNDA is designed to be able to represent and solve.

> The computer is trying to understand and represent the conversation between the conference applicant and the conference office. The applicant wants to cancel his application and get a refund, but he is completely uncertain as to whether he will be allowed to get a refund or not. If not, he wants to send a replacement person. If he asks for permission to send the replacement, he believes that there is about an 80% chance that he will be turned down. The applicant wants to be polite, and does not want to waste time talking about things that are unnecessary. At the same time, the applicant wants to make sure that either he receives a refund, or that a replacement is allowed to attend. The computer must model the situation and predict the next speech acts.

The computer is doing plan recognition to understand the actions of the applicant. The applicant wants to get from Kyoto station to the conference hall. There is a 50% chance that there is a taxi waiting at the station already; otherwise, there is a 50% chance that a taxi will arrive in time with an exponential probability distribution $\theta = 0.5$ minutes. The applicant is uncertain how much the taxi fare is, but he thinks it's probably between $30 and $60. The taxi will take between 20 and 25 minutes to get to the conference hall. He could also take a bus, which would probably cost $5-$10 but take about 40-60 minutes to arrive. He'd have to wait 22 minutes for the bus, though. Or, the applicant could ask for more information–perhaps (3% chance) there's a new subway that goes straight to the conference hall, taking 15 minutes and costing somewhere around $15, that he hasn't heard about yet. It would take about one minute to ask for more information. The computer must recognize and understand the plan of the applicant, and predict what the conference applicant will say next.

A journal paper has almost been finished on the UNDA theory and the MU calculus; this may be referenced for a more complete discussion [Mye90e].

12.2 Applications

UNDA and the MU calculus allow ATR to (a) represent uncertain actions, (b) represent uncertain quantities, (c) plan using uncertain actions and quantities, (d) make decisions about multiple possible uncertain courses of action. These capabilities can be used in at least the following manners:

12.2.1 Representation of Nondeterministic Outcomes in Actions

Using UNDA, the computer is able to represent an action that has more than one possible outcome. This is important when representing real-world dialogs. For instance, the office might grant a request, or it might not. The ability to represent such actions is a prerequisite for being able to reason with them.

12.2.2 Recognition of Plans with Conditionals and Nondeterministic Outcomes

The UNDA theory allows the computer to perform plan recognition and inference on peoples' plans which contain conditionals or uncertain outcomes. People work with the real world, and constantly make contingency plans based on what might occur. In order to understand these plans, it is necessary to have an uncertain action theory such as UNDA. The previous usage of simple sequential plans in plan recognition is insufficient for good understanding. UNDA allows the construction of an uncertain-plan inference system in a straightforward manner; a forthcoming conference paper will be written on such a system.

12.2.3 Planning Information-Gathering Activities

The UNDA theory is ideal for planning information-gathering activities, where the exact kind or amount of information that will be gained is uncertain, and the activity costs effort.

One example of this might be a scenario where a dictionary, or a world-knowledge encyclopedia, is split up between two storage media: a fast main memory, and a slower but larger backup memory. The computer must decide whether to query the larger backup memory or not. In some cases, the results will be unimportant, and it would be a waste of time to wait; in some cases, the backup memory will not know the answer to a query, so this would also be a waste of time. However, in some cases, the backup memory would contribute significant information, and it would be worth it to wait for the results. Choosing when to query the backup system and when to finish computation using only the existing information is an uncertain decision problem.

This problem is related to the following problem.

12.2.4 Interactive Understanding

No matter how good a language understanding system is, it will never be perfect, because the language and the number of concepts in the world are constantly growing. For this reason, it is impossible to have a good automatic interpretation system that operates in a forward direction only. Instead, the system should query the speaker, in an interactive manner, wherenever there is something important that the system does not understand. [13] If the system is allowed to have a small conversation with the speaker in order to determine exactly what the speaker means, then in theory the computer should be able to translate any kind of language, even new concepts, without problems.

However, there is a practical problem with such a system. If the system queries the user about *everything*, including unimportant items, then the user will soon become frustrated. The speaker should only be consulted on important items, that are necessary to the interpretation. Unimportant unclear items should be left alone, even if the system does not understand them.

If there is more than one important item in an utterance that the system does not understand, it must plan which item to discuss first. Anything that the system says to the speaker will change the focus of the computer-human dialog, so it is important to schedule items in a correct order.

Complicating the problem is the question of time. In some cases, if the system thinks for itself for a while, it can understand a difficult problem. In other cases, it might be faster to ask the speaker, even if the computer could disambiguate and understand the utterance itself. So, the system must decide whether to take some extra time, and perhaps succeed in understanding an utterance well, or to quit and ask the speaker.

Finally, there is the matter of seriousness. If the computer hears the speaker utter something with serious consequences, the computer should check with the speaker to make certain that that is what the speaker actually intends should be interpreted. Otherwise, if the computer heard the person incorrectly, or if the speaker did not really mean what he or she said, then the computer could cause much trouble by going ahead and translating the supposed utterance directly.

Thus, when presented with an utterance that it does not yet understand, the system has three alternatives. It can translate what it does understand, and leave the rest ambiguous. It can initiate an interactive dialog with the speaker, and hope to resolve its ignorance. Or, it can keep processing an utterance, and hope that with more searching, perhaps it will be able to understand the utterance better. In all of these cases, the results are uncertain, and the choice of the system will consume time and cause significant results.

[13]The system can use an "interpreter voice" instead of the normal "speaker voice" during voice-generation, to distinguish such questions.

Thus, this is a problem in planning and decision-making with uncertain variables. It is anticipated that the UNDA method will be useful in this area.

12.2.5 Planning Word Choice and Phrasing During Generation

Generation is a planning task. Generation can be divided up at least into "deep generation" and "shallow generation". The task of deep generation is to plan how to communicate a given illocutionary force effectively, given a model of what the hearer *might* know and understand. An advanced program might even plan how to achieve a given perlocutionary force. The task of shallow generation is to choose the particular words and phrasings to communicate the particular concepts that have been decided on, again given a model of the hearer. In both of these cases, the computer's model of the hearer's knowledge is *uncertain*–the computer does not know exactly what the hearer believes. Thus, the computer must make plans with fuzzy information.

Generation in general has two conflicting goals. The system must be as precise as possible, and communicate all information clearly and to sufficient depth. At the same time, the system must not be boring–it must not repeat or mention any information that is "obvious", because the hearer will not like it. It appears that the task of deciding how deeply to expand information in order to meet the conflicting goals of precision and liveliness in an optimal manner, will be helped by the decision-theoretic and uncertainty aspects of UNDA.

12.2.6 Scheduling and Working with Limited Resources

The UNDA theory can be used to decide which actions should be performed, when an action consumes limited resources such as time. Examples of limited-resource action decisions include whether to parse the third or fourth output from the speech-recognition module, whether to perform understanding on the tenth output of the parsing module, whether to start transfer work on preliminary results of the understanding module or wait for full results, and whether to devote much generation resources to an utterance, in order to get a very clean output, or whether to stop polishing the generated utterance and use the existing results. Further discussion of this type of problem can be found in Appendix G.

12.2.7 Understanding Ill-formed Spontaneous Speech Input

The MU calculus can, in principle, be used to determine the most likely intended meaning of an ill-formed or mis-recognized spontaneous utterance. This can be done by combining the evidence of the observed utterance words, along with the expected possible words and illocutionary forces, to determine the most likely interpretation of the utterance. This approach would be similar to the disambiguation approach used to demonstrate understanding of "unagi-da" sentences in Appendix A. Further research would be required.

13 Future Research

The combination of statistical trajectories, causal evidential reasoning, the MU uncertainty calculus, intentional agent simulation and prediction, and the UNDA planning and decision-making scheme, offer a powerful and comprehensive method for dialog understanding.

Several research items must be explored and completed in the near future. These include finishing the construction of the MU calculus routines, and the UNDA planning and decision-making engine. This can then be adapted for recognizing plans with conditionals, and plans dealing with nondeterministic actions. At the same time, the causal evidential reasoning engine for trees must be finished. After this is working well, it must be expanded to deal with multiple parents, networks with loops, and explanation sets, in that order, before it can represent the realistic types of problems that ATR deals with in the sample conversations.

Both the UNDA and the causal evidential reasoning engines will be implemented on the Sequent parallel-process computer, for speed.

The next item must be to build a disambiguation system based on the causal evidential reasoning and the UNDA plan recognition methods. The disambiguation system will be weak at first, because it will not be supplied with general rules. Disambiguation must then be augmented with an utterance/speech-act prediction system that generates alternatives, based on a trajectory representation system and the intentional agent simulator. More research must be performed on understanding and predicting the actions of an intentional agent, so as to build a good (general-purpose) predictor. In this way, the disambiguation system will become stronger, and will not be so fragile as to only be able to process our example conversations.

It will be necessary for the UNDA, causal evidential reasoning, and trajectories systems to have the ability to train themselves, using statistics extracted automatically from conversations. This will be required in order to get realistic probabilities. Therefore, after these reasoning engines have been implemented and their capabilities demonstrated, it will in the future be necessary to research statistical updating and automatic learning methods for these systems.

14 Conclusion

This paper has reported on the main parts of the research in dialog understanding performed in 1990. More detailed discussions of specific projects can be found in individual papers, e.g. [Mye90e], [Mye89c], [Mye90f], [Mye90a], etc. Research on dialog understanding is continuing. Special emphasis is being provided to the end goal of constructing a working disambiguation/prediction system, based on causal evidential reasoning and uncertain non-deterministic action theory.

A Sketch of Handling Unagi-da-bun by a Causal Evidential Reasoning System

As was discussed in Section 7, most problems in natural-language understanding can be classified as disambiguation problems. Some problems, such as zero-pronoun anaphora, require supplying a missing noun for the sentence. Other problems, such as interpreting "onegaishimasu" or "sore o kudasai", require supplying a missing verb for the sentence. One of the classic missing-verb cases in Japanese is a class of sentences known as "unagi-da-bun" ("eel-sentences"), where the verb and the verbal object case markers are elided and replaced by the copula. The name comes from the following famous example:

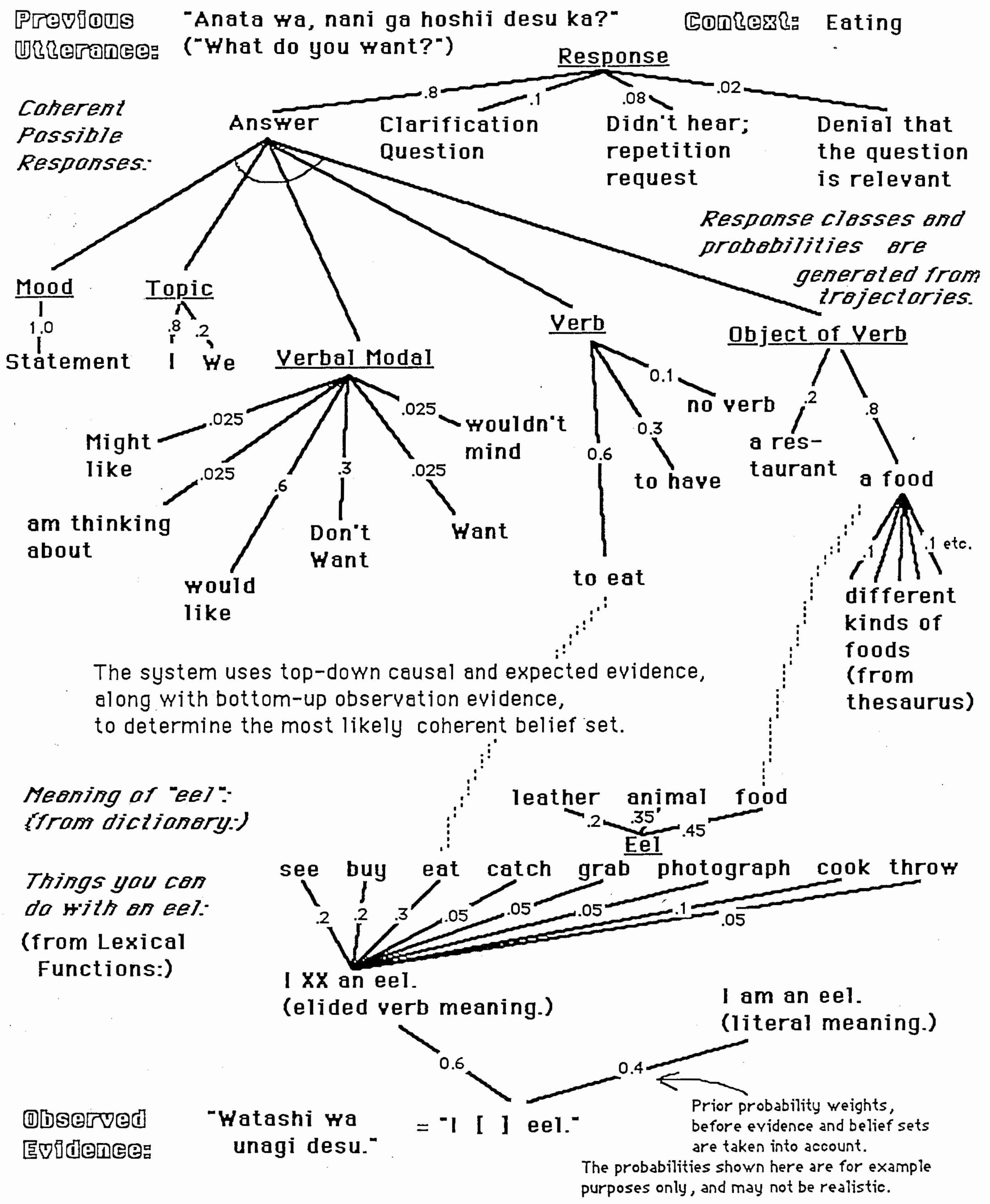
```
Anata wa nan no tabemono ga suki desu ka?
  (What kind of food would you like [to eat]?)
Watashi wa unagi da.
  (lit., I am an eel.) (I would like to eat eel.)
```

This section will provide a brief sketch of how a causal evidential reasoning system could disambiguate and understand the meaning of such a sentence. It is important to note that none of the algorithms that are discussed in this appendix have been implemented, nor has any of the theory been tested under realistic conditions. This is a sketch only, not a working model. Nonetheless, it offers a practical target and conceptual demonstration of how such a system is envisioned to work. It is anticipated that the disambiguation system described in this report, when implemented, will solve such problems in such a manner.

Probabilities in general are specified by dynamic beliefs arising from the situation, by prior probabilities and expectations, and by evidence provided from observations. An evidential reasoning system provides a good match for the problem of disambiguation of meaning. When the observed utterance "Watashi wa unagi desu" comes in, it provides evidence for both the literal meaning, "I am an eel", and also the elided-verb meaning, "I *something* an eel". If it is indeed an elided verb, the verb must be filled in by something that one can do with an eel. There are only a handful of likely verbs that apply to eels; each one has a prior statistical probability, that could be adjusted due to context. The very fact that a human can answer the question, "What kinds of things can one do with an eel?", provides evidence that some kind of similar process seems to be occuring when humans understand sentences. These verbs can be provided by using a Lexical Function network [SS90].

At the same time, the meanings (intensions) of the nouns in the sentence must be disambiguated. For instance, the word "eel" can be used as a reference to indicate a type of leather, a type of animal, or a type of food. These meanings can be taken from a thesaurus or a dictionary. They will also have prior probabilities based on statistical usages.

At the same time that bottom-up reasoning is proceeding, the system is also performing top-down reasoning. The system uses trajectories, and eventually agent models, to predict the general classes of coherent response from the speaker. Once again, there are only a handful of types of responses, each with an associated statistical probability: the speaker could answer the question, he could ask a clarification question, he could ask the questioner to repeat the question, or he could deny that the question is relevant. All possibilities are explored. If the response takes the form of an answer, it will have a conjunction of qualities, including such things as mood, topic, verbal modals, verb, and object of the verb,



Previous Utterance:
"Anata wa, nani ga hoshii desu ka?"
("What do you want?")
Context: Eating
Response
.8
.1
.08
.02
Coherent Possible Responses:
Answer
Clarification Question
Didn't hear; repetition request
Denial that the question is relevant
Response classes and probabilities are generated from trajectories.
Mood
1.0
Statement
Topic
.8
.2
I
We
Verbal Modal
.025
Might like
.025
am thinking about
.6
would like
.3
Don't Want
.025
Want
.025
wouldn't mind
Verb
0.1
no verb
0.3
to have
0.6
to eat
Object of Verb
.2
a restaurant
.8
a food
.1
.1 etc.
different kinds of foods (from thesaurus)
The system uses top-down causal and expected evidence,
along with bottom-up observation evidence,
to determine the most likely coherent belief set.
Meaning of "eel":
(from dictionary:)
leather
animal
food
.2
.35
.45
Eel
Things you can do with an eel:
(from Lexical Functions:)
see
buy
eat
catch
grab
photograph
cook
throw
.2
.2
.3
.05
.05
.05
.1
.05
I XX an eel.
(elided verb meaning.)
I am an eel.
(literal meaning.)
0.6
0.4
Prior probability weights, before evidence and belief sets are taken into account.
The probabilities shown here are for example purposes only, and may not be realistic.
Observed Evidence:
"Watashi wa unagi desu."
= "I [ ] eel."

Figure 3: Sketch of an Evidential Reasoning System Disambiguating an "Unagi-Da" Utterance

etc. Although it will eventually be important to form coherent sets of these possibilities, a first approximation allows each quality to be considered separately. For instance, the verbal modal might be "might like", "want", or "wouldn't mind". Each possibility again has an associated probability, based on prior probabilities and the situation.

The probabilistic expectations based on the previous utterance get mixed with the probabilistic evidence provided by the current utterance. Even though there are many different possibilities, the system is able to pick the most probable consistent set. In this way, the causal evidential reasoning system, combined with other supporting systems, will be able to disambiguate utterances with elided significant information.

B Hand-Classification of Illocutionary Forces

B.1 Classification of Conversation 1

LM01:>myers>con>illoc-conv1-JREF.txt.2

```
CONVERSATION #1                    ILLOCUTIONARY FORCE

Guest:   もしもし
Guest:   Moshi-moshi.
         Hello.
[[RELN   もしもし-OPEN_DIALOGUE]
assertive: Greet

         そちらは会議事務局ですか
         Sochira wa kaigi jimu-kyoku desu ka?
         Is this the Conference office?
[[RELN   S-REQUEST]
 [OBJE [[RELN  INFORMIF]
commissive: Askq

Office:  はい
         Hai.
         Yes,
[[RELN   はい-AFFIRMATIVE]
declarative: Confirm

         そうです
         Sou desu.
         that's right.
[[RELN   そうです-CONFIRMATION]
declarative: Confirm

         どのようなご用件でしょうか
         Dono youna go-youken deshou ka?
         May I help you?
[[RELN   S-REQUEST]
 [OBJE [[RELN  INFORMREF]
commissive: Offer

Guest:   会議に申し込みたいのですが
         Kaigi ni moushikomitai no desu ga.
         I would like to attend the Conference.
[[RELN   が-MODERATE]
 [OBJE [[RELN  たい-DESIRE]
assertive: Inform
indirect directive: Request (help, advice)

         どのような手続きをすればよろしいのでしょうか
         Dono youna tetsuzuki o sureba yoroshii no deshou ka?
         How can I apply?
[[RELN   S-REQUEST]
 [OBJE [[RELN  INFORMREF]
directive: Askq (factual question)
indirect directive: Request (help)

Office:  登録用紙で手続きをして下さい
         Touroku-youshi de tetsuzuki o shite-kudasai.
         Please fill out a registration form.
[[RELN   下さい-REQUEST]
directive: Direct

         登録用紙は既にお持ちでしょうか
         Touroku-youshi wa sude-ni o-mochi deshou ka?
         Do you have one?
[[RELN   S-REQUEST]
 [OBJE [[RELN  INFORMIF]
directive: Askq

Guest:   いいえ
         Iie.
         No,
[[RELN   いいえ-NEGATIVE]
assertive: Inform

         まだです
         Mada desu.
         not yet.
[[RELN   だ-STATEMENT]
 [OBJE [[PARM ! X01[]]
         [RESTR [[RELN  まだ-1]
assertive: Inform
```

```
indirect directive: Request
possible indirect expresive: Complain*

Office: 分かりました
        Wakarimashita.
        I see.
[[RELN  た-PERFECTIVE]
 [OBJE [[RELN  分かる-1]
expressive: Acknowledge
assertive: Inform

        それでは登録用紙をお送り致します
        Soredewa, touroku-youshi o o-okuriitashimasu.
        Then, I'll send you a registration  form.
[[RELN  送る-1]
commissive: Promise

        ご住所とお名前をお願いします
        Go-juusho to o-namae o onegaishimasu.
        Would you please give me your name and address?
[[RELN  願う-REQUEST]
directive: Askq

Guest:  住所は大阪市北区茶屋町二十三です
        Juusho wa Oosaka-shi Kita-ku Chaya-machi 23 (ni-juu san) desu.
        My address  is  23  Chayamachi, Kita-ku, Osaka.
[[RELN  だ-IDENTICAL]
assertive: Inform

        名前は鈴木真弓です
        Namae wa Suzuki Mayumi desu.
        My name is Mayumi Suzuki.
[[RELN  だ-IDENTICAL]
assertive: Inform

Office: 分かりました
        Wakarimashita.
        I've got it.
[[RELN  た-PERFECTIVE]
 [OBJE [[RELN  分かる-1]
expressive: Acknowledge

        登録用紙を至急送らせて頂きます
        Touroku-youshi o shikyuu okurasete-itadakimasu.
        I'll send you the form.immediately.
[[RELN  てもらう-RECEIVE_FAVOR]
 [OBJE [[RELN  させる-PERMISSIVE]
        [OBJE [[RELN  送る-1]
commissive: Promise

Guest:  よろしくお願いします
        Yoroshiku onegaishimasu.
        Thank you very much.
[[RELN  願う-REQUEST]
 [INFMANN [[PARM !X01[]]
           [RESTR [[RELN  よろしく-1]
expressive: Thank/Say-Goodbye

        それでは失礼します
        Soredewa shitsurei-shimasu.
        Good-bye.
[[RELN  失礼する-CLOSE_DIALOGUE]
expressive: Say-Goodbye
```

* If the guest EXPECTED to have received a form before now,
and believes that it is UNFAIR or NOT-RIGHT to have not received a form,
then the guest FEELS-INJURED.

If the guest FEELS-INJURED, and the guest communicates that fact,
then the guest is COMPLAINING.

If the guest FEELS-INJURED,
and believes that it is the FAULT (= CAUSE + NOT-RIGHT) of the office,
then the guest BLAMES the office.

It is possible to COMPLAIN to someone who is BLAMELESS.

B.2 Classification of Conversation 2

```
CONVERSATION 2                    ILLOCUTIONARY FORCE
Office: はい
        Hai.
        Hello.                          LM01:>myers>con>illoc-conv2-JREF.txt.2
[[RELN  はい-AFFIRMATIVE]
expressive: Greet

        こちらは会議事務局です
        Kochira wa kaigi-jimukyoku desu.
        The Conference office.
[[RELN  だ-IDENTICAL]
 [OBJE [[LABEL *SPEAKER*]]]
declarative: Self-Identify

Guest:  会議の参加料について教えて頂きたいのですが
        Kaigi no sanka-ryou ni tsuite oshiete-itadakitai no desu ga.
        Could you give me some information about the application fee for the Conference?
[[RELN  が-MODERATE]
 [OBJE [[RELN  たい-DESIRE]
        [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
directive: Request
assertive: Inform
indirect directive: Askq

        いま会議に申し込めば参加料はいくらですか
        Ima kaigi ni moushi-komeba sanka-ryou wa ikura desu ka?
        How much will it cost if I apply for the Conference right now?
[[RELN  S-REQUEST]
 [OBJE [[RELN  INFORMREF]
directive: Askq

Office: はい
        Hai.
        Well, let's see.
[[RELN  はい-AFFIRMATIVE]
expressive: Acknowledge

        参加料は現在お一人三万五千円です
        Sanka-ryou wa genzai o-hitori san-man go-sen-en desu.
        It costs 35,000 yen per person.
[[RELN  だ-IDENTICAL]
assertive: Inform

        来月お申し込みになりますと四万円です
        Rai-getsu omoushikomi ni narimasu to yon-man-en desu.
        But if you apply next month, it will cost you  40,000 yen.
[[RELN  だ-IDENTICAL]
 [COND [[PARM !X02[]]
        [RESTR [[RELN  と-CONDITIONAL]
assertive: Inform
directive: Advise

        参加料には、予稿集代と歓迎会費が含まれています
        Sanka-ryou niwa, yokou-shuudai to  kangei-kaihi ga  fukumarete-imasu.
        The proceedings and the reception are included in the application fee.
[[RELN  ている-STATIVE]
 [OBJE [[RELN  れる-PASSIVE]
assertive: Inform

Guest:  わたしは情報処理学会の会員なのですが
        Watashi wa Jouhousho-Rigakkai no  kaiin na no desu ga.
        I am a member of the Information Processing Society.
[[RELN  が-MODERATE]
 [OBJE [[RELN  だ-IDENTICAL]
        [OBJE [[LABEL  *SPEAKER*]]]]]]
assertive: Inform
indirect directive: Request (help)

        参加料の割引はないのですか
        Sanka-ryou no  waribiki wa nai no desu ka?
        Is there a discount for  members?
[[RELN  S-REQUEST]
 [OBJE [[RELN  INFORMIF]
directive: Askq
indirect directive: Request (discount)
```

Office: 今回は割引を行っておりません
Kon-kai wa waribiki o okonatte-orimasen.
No, there's no discount this time.
[[RELN NEGATE]
[OBJE [[RELN ている-PROGRESSIVE]
commissive: Refuse
assertive: Inform
indirect expressive: Apologize

Guest: そうですか
Sou desu ka.
I understand.
[[RELN そうですか-CONFIRMATION]
expressive: Acknowledge
(expressive: Complain)

参加料はどのようにお支払いしたらよいのですか
Sanka-ryou wa dono youni oshiharai-shitara yoi no desu ka?
How can I pay?
[[RELN S-REQUEST]
[OBJE [[RELN INFORMREF]
directive: Askq

Office: 参加料は銀行振り込みです
Sanka-ryou wa ginkou-furikomi desu.
Payment should be made by bank-transfer.
[[RELN だ-IDENTICAL]
directive: Direct
assertive: Inform

案内書に載せられている口座番号に振り込んで下さい
Annaisho ni kisaisarete-iru kouza-bangou ni furikonde-kudasai.
Please remit to our bank account which is mentioned in the announcement.
[[RELN 下さい-REQUEST]
directive: Request/direct

また期限は今年いっぱいです
Mata kigen wa kotoshi ippai desu.
And, the deadline is the end of the year.
[[RELN だ-IDENTICAL]
directive: Advise
assertive: Inform

Guest: 分かりました
Wakarimashita.
I see.
[[RELN た-PERFECTIVE]
expressive: Acknowledge

どうもありがとうございました
Doumo arigatou gozaimashita.
Thank you very much.
[[RELN た-PERFECTIVE]
[OBJE [[RELN ありがとう-THANKING]
expressive: Thank

Office: どういたしまして
Dou itashimashite.
You're welcome.
[[RELN どういたしまして-GREETING]
expressive: Welcome

分からない点がございましたらいつでもお聞き下さい
Wakaranai ten ga gozaimashitara itsudemo o-kiki-kudasai.
Please feel free to ask if there's anything you don't understand.
[[RELN 下さい-REQUEST]
commissive: Offer
expressive: Say-Goodbye

失礼致します
Shitsurei-itashimasu.
Good-bye.
[[RELN 失礼する-CLOSE_DIALOGUE]
expressive: Say-Goodbye

B.3 Classification of Conversation 3

LM01:>myers>con>illoc-conv3-JREF.txt.2

```
CONVERSATION 3                      ILLOCUTIONARY FORCE
Office: はい
        Hai.
        Hello.
[[RELN  はい-AFFIRMATIVE]
expressive: Greet

        こちらは会議事務局です
        Kochira wa kaigi-jimukyoku desu.
        The Conference office.
[[RELN  だ-IDENTICAL]
 [OBJE [[LABEL  *SPEAKER*]]]]
declarative: Self-identify

Guest:  会議に論文を発表したいと思っているのですが
        Kaigi ni ronbun o happyou-shitai to omotte-iru no desu ga.
        I would like to contribute a paper to the Conference.
[[RELN  が-MODERATE]
 [OBJE [[RELN  ている-STATIVE]
assertive: Inform (want)
indirect directive: Request (help)

        会議の内容について教えて下さい
        Kaigi no naiyou ni tsuite oshiete-kudasai.
        Would you please tell me the topic of the Conference?
[[RELN  下さい-REQUEST]
directive: Request/Askq

Office: 今回の会議は通訳電話に関連する広範な研究分野を含んでいます
        Kon-kai no kaigi wa tsuuyaku-denwa ni kanren suru kouhan-na kenkyuu-bunya o fukunde-imasu
.
        This Conference covers a wide area of research related to Interpreting Telephony.
[[RELN  ている-STATIVE]
assertive: Inform

        言語学や心理学を専攻する方にも参加して頂く予定です
        Gengo-gaku ya shinri-gaku o senkou-suru  kata nimo sanka-shite-itadaku yotei desu.
        We are also expecting linguistists and psychologists as participants.
[[RELN  予定だ-1]
 [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
assertive: Inform

Guest:  分かりました
        Wakarimashita.
        I see.
[[RELN  分かった-CONFIRMATION]
expressive: Acknowledge

        ところで会議での公式言語は何ですか
        Tokorode, kaigi de no koushiki-gengo wa nan desu ka?
        By the way, what is the official language of the Conference?
[[RELN  S-REQUEST]
 [OBJE [[RELN  INFORMREF]
directive: Askq

Office: 英語と日本語です
        Eigo to Nihongo desu.
        English and Japanese.
[[RELN  だ-IDENTICAL]
assertive: Inform

Guest:  わたしは日本語が全然分からないのですが
        Watashi wa Nihongo ga zenzen wakaranai no desu ga.
        I don't understand Japanese at all.
[[RELN  が-MODERATE]
 [OBJE [[RELN  NEGATE]
        [OBJE [[RELN  分かる-2]
assertive: Inform
(expressive: Complain)
possible indirect directive: Request

        発表が日本語で行われる場合英語への同時通訳はあるのですか
        Happyou ga Nihongo de okonawareru  baai  eigo e no douji-tsuuyaku wa aru no desu ka?
        Is there simultaneous interpretation into English when the presentation is made in Japane
se?
[[RELN  S-REQUEST]
```

```
 [OBJE [[RELN  INFORMIF]
directive: Askq

Office: はい
        Hai.
        Yes.
[[RELN  はい-AFFIRMATIVE]
assertive: Inform?
expressive: Acknowledge?

        英語への同時通訳を用意しております
        Eigo e no douji-tsuuyaku o youi-shite-orimasu.
        We have simultaneous interpretation service into English.
[[RELN  ている-PROGRESSIVE]
 [OBJE [[RELN  用意する-1]
assertive: Inform
(expressive: Welcome)*

Guest:  分かりました
        Wakarimashita.
        That would be helpful for me.
[[RELN  た-PERFECTIVE]
 [OBJE [[RELN  分かる-1]
expressive: Acknowledge
expressive: Thank

        どうもありがとうございました
        Doumo arigatou gozaimashita.
        Thank you very much.
[[RELN た-PERFECTIVE]
 [OBJE [[RELN ありがとう-THANKING]
expressive: Thank
expressive: Say-Goodbye

        さようなら
        Sayounara.
        Good-bye.
[[RELN  さようなら-GOOD-BYE]
expressive: Say-Goodbye
```

*WELCOME: You are coming to someplace that is mine.
I tell you that I will provide for your needs.

B.4 Classification of Conversation 4

```
CONVERSATION 4                    ILLOCUTIONARY FORCE

Office: こちらは会議事務局です
        Kochira wa kaigi-jimukyoku desu.
        The Conference office.               LM01:>myers>con>illoc-conv4-JREF.txt.2
[[RELN  だ-IDENTICAL]
 [OBJE [[LABEL *SPEAKER*]]]
declarative:  Self-identify
expressive:   Greet

Guest:  会議について詳しいことを教えて下さい
        Kaigi ni tsuite kuwashii koto o oshiete-kudasai.
        I would like to know the details of the Conference.
[[RELN  下さい-REQUEST]
 [OBJE [[RELN  教える-1]
directive: Request/Askq

Office: 会議の案内書はお持ちですか
        Kaigi no annai-sho wa o-mochi desu ka?
        Do you have an announcement of the Conference?
[[RELN  S-REQUEST]
 [OBJE [[RELN  INFORMIF]
directive: Askq

Guest:  いいえ
        Iie.
        No.
[[RELN  いいえ-NEGATIVE]
assertive: Inform

        持っていません
        Motte-imasen.
        I don't.
[[RELN  NEGATE]
 [OBJE [[RELN  ている-STATIVE]
assertive: Inform

Office: そうですか
        Sou desu ka.
        OK.
[[RELN  そうですか-CONFIRMATION]
expressive: Acknowledge

        会議は8月22日から25日まで京都国際会議場で開催されます
        Kaigi wa hachi-gatsu ni-juu-ni-nichi kara ni-juu-go-nichi made Kyouto Kokusai-Kaigi-Jou d
e kaisaisaremasu.
        The Conference will take place from August 22nd to 25th at the Kyoto International Confer
ence Center.
[[RELN  れる-PASSIVE]
 [OBJE [[RELN  開催する-1]
assertive: Inform

        参加料は4万円です
        Sanka-ryou wa yon-man-en desu.
        The fee for participation is 40,000 yen.
[[RELN  だ-IDENTICAL]
assertive: Inform

        発表を希望されるのでしたら3月20日までに要約を提出して下さい
        Happyou o kibousareru no deshitara  san-gatsu (20)hatsuka made ni youyaku o teishutsu-shi
te-kudasai.
        If you intend to present a paper, please submit a summary by March 20th.
[[RELN  下さい-REQUEST]
 [OBJE [[RELN  提出する-1]
directive: Request
directive: Advise
assertive: Inform

        会議の案内書をお送り致しますのでそれをご覧下さい
        Kaigi no annai-sho o o-okuri-itashimasu no de, sore o goran-kudasai.
        I'll send the Conference announcement to you soon, please look at it.
[[RELN  下さい-REQUEST]
 [OBJE [[RELN  ご覧-1]
commissive: Promise
assertive: Inform
```

```
directive: Direct

        失礼ですがお名前とご住所をお願いします
        Shitsurei desu ga  o-namae to go-juusho o onegai-itashimasu.
        Would you mind telling me your name and address?
[[RELN  願う-REQUEST]
directive: Request/Askq

Guest:  アダムスミスです
        Adamu Sumisu desu.
        My name is Adam Smith.
[[RELN  だ-IDENTICAL]
declarative: Self-identify
assertive:  Inform

        住所は大阪市東区玉造2丁目27の7です
        Juusho wa Oosaka-shi Higashi-ku Tamatsukuri Ni-choume 27-7(nijuunanano nana) desu.
        My address is 2-27-7 Tamatsukuri, Higashi-ku, Osaka.
[[RELN  だ-IDENTICAL]
assertive: Inform

Office: 分かりました
        Wakarimashita.
        OK.
[[RELN  た-PERFECTIVE]
 [OBJE [[RELN  分かる-1]
expressive: Acknowledge

        電話番号もお聞きしたいのですが
        Denwa-bangou mo  o-kiki-shitai no desu ga.
        Could I have your phone number too?
[[RELN  が-MODERATE]
 [OBJE [[RELN  たい-DESIRE]
        [OBJE [[RELN  聞く-1]
directive: Request/Askq

Guest:  はい
        Hai.
        Yes.
[[RELN  はい-AFFIRMATIVE]
expressive: Acknowledge

        三七二の八〇一八です
        372-8081 (Sannananino hachizeroichihachi) desu.
        372-8018.
[[RELN  だ-IDENTICAL]
assertive: Inform

Office: 三七二の八〇一八でございますね
        372-8081 (Sannananino hachizeroichihachi) de gozaimasu ne.
        372-8018.  Is that correct?
[[RELN  ね-CONFIRMATION]
 [OBJE [[RELN  だ-IDENTICAL]
directive: Askq

Guest:  はい
        Hai.
        Yes.
[[RELN  はい-AFFIRMATIVE]
declarative: Confirm

        そうです
        Sou desu.
        It is.
[[RELN  そうです-CONFIRMATION]
declarative: Confirm

        それではよろしくお願いします
        Soredewa  yoroshiku onegaishimasu.
        Thank you very much.
[[RELN  願う-REQUEST]
 [MANN [[PARM !X01[]]
        [RESTR [[RELN  よろしく-1]
expressive: Thank/Say-Goodbye

        失礼します

        Shitsurei-shimasu.
        Good-bye.
[[RELN  失礼する-CLOSE_DIALOGUE]
expressive: Say-Goodbye
```

B.5 Classification of Conversation 5

LM01:>myers>con>illoc-conv5-JREF.txt.2

```
CONVERSATION 5                  ILLOCUTIONARY FORCE

Office: はい
        Hai.
        Hello.
[[RELN  はい-AFFIRMATIVE]
```
expressive: Greet

```
        こちらは会議事務局でございます
        Kochira wa kaigi-jimukyoku de gozaimasu.
        The Conference office.
[[RELN  だ-IDENTICAL]
 [OBJE [[LABEL  *OFFICE*]]]
```
declarative: Self-Identify
expressive: Greet

```
Guest:  ちょっとお願いがあるのですが
        Chotto onegai ga aru no desu ga.
        I wonder if you could help me.
[[RELN  が-MODERATE]
 [OBJE [[RELN  ある-1]
```
assertive: Inform
directive: Request

```
        私は会議に申込みをした者です
        Watashi wa kaigi ni moushikomi o shita mono desu.
        I sent in the registration form for the Conference.
[[RELN  だ-IDENTICAL]
 [OBJE [[LABEL  *GUEST*]]]
```
assertive: Inform

```
        参加を取り消したいのですが
        Sanka o torike-shitai no desu ga.
        But I can't attend the Conference, so I would like to cancel.
[[RELN  が-MODERATE]
 [OBJE [[RELN  たい-DESIRE]
```
assertive: Inform
indirect directive: Request/Direct

```
Office: お名前をお伺いできますでしょうか
        Onamae o o-ukagai dekimasu deshou ka?
        Could you please give me your name?
[[RELN  S-REQUEST]
 [OBJE [[RELN  INFORMIF]
```
directive: Request/Askq

```
Guest:  はい
        Hai.
        Yes.
[[RELN  はい-AFFIRMATIVE]
```
expressive: Acknowledge

```
        ベル研のジムワイベルです
        Beru-ken no Jimu Waiberu desu.
        This is Jim Waibel from Bell  Labs.
[[RELN  だ-IDENTICAL]
```
assertive: Inform
declarative: Self-Identify

```
Office: 既に登録料の8万5千円を振り込まれておられますね
        Sude-ni touroku-ryou no hachi-man go-sen-en o furikomarete-oraremasu ne.
        Mr. Waibel, you have already paid 85,000 yen for your registration fee, haven't you?
[[RELN  ね-CONFIRMATION]
 [OBJE [[RELN  ている-PROGRESSIVE]
```
directive: Askq

```
Guest:  はい
        Hai.
        Yes,
[[RELN  はい-AFFIRMATIVE]
```
declarative: Confirm

```
        そうです
        Sou desu.
        I have.
[[RELN  そうです-CONFIRMATION]
```

declarative: Confirm

参加料を払い戻して頂けますか
Touroku-ryou o harai-modoshite-itadakemasu ka?
Is it possible for you to refund the registration fee?
[[RELN S-REQUEST]
[OBJE [[RELN INFORMIF]
directive: Ask
indirect directive: Request/Direct (refund)

Office: お気の毒ですができません
O-ki no doku desu ga dekimasen.
I am sorry, we can't.
[[RELN NEGATE]
[OBJE [[RELN できる-POSSIBLE]
expressive: Apologize
assertive: Inform
commissive: Reject/Refuse

案内書にも書いていますが 9月27日以後の取り消しに対する払い戻しできません
Annai-sho nimo kaite-imasu ga ku-gatsu ni-juu-nana-nichi igono torike-shinitai-suru ha
rai-modoshi wa dekimasen.
As noted in the announcement, cancellation after September 27th precludes a refund.
[[RELN が-MODERATE]
[OBJE [[RELN ている-STATIVE]
[[RELN NEGATE]
[OBJE [[RELN できる-POSSIBLE]
assertive: Inform
(commissive: Reject/Refuse)

後日プログラムと予稿集をお送り致します
Gojitsu puroguramu to yokoushuu o o-okuri-itashimasu.
We'll send you the programs and proceedings later.
[[RELN 送る-1]
assertive: Inform
directive: Promise

Guest: では誰かが私の代わりに参加することはできますか
Dewa, dare-ka ga watashi no kawari ni sanka-suru koto wa dekimasu ka?
Will somebody else be able to attend the Conference instead of me, then?
[[RELN S-REQUEST]
[OBJE [[RELN INFORMIF]
directive: Askq (information)
directive: Request (permission, help)

Office: それは別に問題ありません
Sore wa betsu ni mondai arimasen.
That's all right.
[[RELN NEGATE]
[MANN [[RELN 別だ-1]
[OBJE [[RELN 問題ある-1]
commissive: Accept/Consent
directive: Permit

代理人が参加する場合はあらかじめこちらまでお知らせ下さい
Dairi-nin ga sanka-suru baai wa, araka-jime kochira made oshirase-kudasai.
Please let me know in advance who is going to attend instead of you.
[[RELN 下さい-REQUEST]
directive: Request/Direct

Guest: 分かりました
Wakarimashita.
I understand.
[[RELN た-PERFECTIVE]
[OBJE [[RELN 分かる-1]
expressive: Acknowledge

代理人が決まりましたらお知らせ致します
Dairi-nin ga kimarimashitara, oshirase-itashimasu.
I'll let you know when it's decided.
[[RELN せる-CAUSATIVE]
[OBJE [[RELN 知る-1]
assertive: Inform
commissive: Promise

では失礼します
Dewa shitsurei-shimasu.
Good-bye.
[[RELN 失礼する-CLOSE_DIALOGUE]
expressive: Say-Goodbye

B.6 Cross-Index of Classifications

```
ILLOCUTIONARY-FORCE CROSS-INDEX
```

LM01:>myers>con>illoc-cross-index-F.txt.3

```
[[RELN  ある-1]
       [OBJE [[PARM ! X01[]]
              [RESTR [[RELN  お願い-1]
assertive: Inform
directive: Request


[[RELN  だ-IDENTICAL]
assertive: Inform
-----------------
assertive: Inform
-----------------
assertive: Inform
-----------------
assertive: Inform
-----------------
assertive: Inform
-----------------
assertive: Inform
-----------------
assertive: Inform
-----------------
directive: Direct
assertive: Inform
-----------------
directive: Advise
assertive: Inform
-----------------
declarative: Self-identify
assertive:  Inform
------------------
assertive: Inform
declarative: Self-Identify
------------------
ね: directive: Askq


[[RELN  だ-IDENTICAL]
 [OBJE [[LABEL *SPEAKER*]]]
declarative: Self-Identify
--------------------------
declarative: Self-identify
--------------------------
declarative:  Self-identify
expressive:  Greet
--------------------------
declarative: Self-Identify
expressive: Greet
---------------------------
assertive: Inform
indirect directive: Request (help)
---------------------------
assertive: Inform


[[RELN  だ-IDENTICAL]
 [COND [[PARM ! X02[]]
        [RESTR [[RELN  と-CONDITIONAL]
assertive: Inform
directive: Advise


[[RELN  だ-STATEMENT]
 [OBJE [[PARM ! X01[]]
        [RESTR [[RELN  まだ-1]
assertive: Inform
indirect directive: Request
possible indirect expresive: Complain


[[RELN  できる-POSSIBLE]
(NEGATE)expressive: Apologize
```

```
assertive: Inform
commissive: Reject/Refuse
-------------------------
assertive: Inform
(commissive: Reject/Refuse)


[[RELN どういたしまして-GREETING]
expressive: Welcome


[[RELN  が-MODERATE]
 [OBJE [[RELN  ある-1]
        [OBJE [[PARM !X01[]]
               [RESTR [[RELN  お願い-1]
assertive: Inform
directive: Request


[[RELN  が-MODERATE]
 [OBJE [[RELN  たい-DESIRE]
assertive: Inform
indirect directive: Request (help, advice)
-------------------------------------------
directive: Request/Askq
-----------------------
assertive: Inform
indirect directive: Request/Direct


[[RELN  が-MODERATE]
 [OBJE [[RELN  たい-DESIRE]
        [OBJE [[RELN  聞く-1]
directive: Request/Askq


[[RELN  が-MODERATE]
 [OBJE [[RELN  たい-DESIRE]
        [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
directive: Request
assertive: Inform
indirect directive: Askq


[[RELN  が-MODERATE]
 [OBJE [[RELN  だ-IDENTICAL]
        [OBJE [[LABEL  *SPEAKER*]]]]]]
assertive: Inform
indirect directive: Request (help)


[[RELN  が-MODERATE]
 [OBJE [[RELN  NEGATE]
        [OBJE [[RELN  分かる-2]
assertive: Inform
(expressive: Complain)
possible indirect directive: Request


[[RELN  が-MODERATE]
 [OBJE [[RELN  ている-STATIVE]
assertive: Inform (want)
indirect directive: Request (help)


[[RELN  はい-AFFIRMATIVE]
declarative: Confirm
--------------------
declarative: Confirm
--------------------
declarative: Confirm
--------------------
expressive: Greet
------------------
expressive: Greet
------------------
```

```
expressive: Greet
-------------------
expressive: Acknowledge
------------------------
expressive: Acknowledge
------------------------
expressive: Acknowledge
------------------------
assertive: Inform?
expressive: Acknowledge?


[[RELN  いいえ-NEGATIVE]
assertive: Inform
------------------
assertive: Inform


 [OBJE [[RELN  開催する-1]
assertive: Inform


[[RELN  下さい-REQUEST]
directive: Direct
------------------
directive: Request/direct
--------------------------
directive: Request/Direct
--------------------------
directive: Request/Askq
------------------------
directive: Request/Askq
------------------------
commissive: Offer
expressive: Say-Goodbye
------------------------
directive: Request
directive: Advise
assertive: Inform
-------------------
directive: Direct
  * Compound sentence:
commissive: Promise
assertive: Inform


[[RELN  下さい-REQUEST]
 [OBJE [[RELN  ご覧-1]
directive: Direct
  * Compound sentence:
commissive: Promise
assertive: Inform


[[RELN  下さい-REQUEST]
 [OBJE [[RELN  教える-1]
directive: Request/Askq


[[RELN  下さい-REQUEST]
 [OBJE [[RELN  提出する-1]
directive: Request
directive: Advise
assertive: Inform


[[RELN  問題ある-1]
(NEGATIVE)commissive: Accept/Consent
directive: Permit


[[RELN  もしもし-OPEN_DIALOGUE]
assertive: Greet


[[RELN  ね-CONFIRMATION]
```

```
 [OBJE [[RELN  だ-IDENTICAL]
directive: Askq

[[RELN  ね-CONFIRMATION]
 [OBJE [[RELN  ている-PROGRESSIVE]
directive: Askq


[[RELN  NEGATE]
 [OBJE [[RELN  できる-POSSIBLE]
expressive: Apologize
assertive: Inform
commissive: Reject/Refuse
-------------------------
assertive: Inform
(commissive: Reject/Refuse)


[[RELN  NEGATE]
 [MANN [[RELN  別だ-1]
 [OBJE [[RELN  問題ある-1]
commissive: Accept/Consent
directive: Permit


[[RELN  NEGATE]
 [OBJE [[RELN  ている-PROGRESSIVE]
commissive: Refuse
assertive: Inform
indirect expressive: Apologize

[[RELN  NEGATE]
 [OBJE [[RELN  ている-STATIVE]
assertive: Inform


 [OBJE [[RELN  NEGATE]
        [OBJE [[RELN  分かる-2]
assertive: Inform
(expressive: Complain)
possible indirect directive: Request


[[RELN  送る-1]
commissive: Promise
-------------------
commissive: Promise
-------------------
assertive: Inform
directive: Promise


[RESTR [[RELN  お願い-1] (が ある)
assertive: Inform
directive: Request

[[RELN  願う-REQUEST]
directive: Askq
-----------------------
directive: Request/Askq


[[RELN  願う-REQUEST]
 [INFMANN [[PARM ! X01[]]
           [RESTR [[RELN  よろしく-1]
expressive: Thank/Say-Goodbye
-----------------------------
expressive: Thank/Say-Goodbye


[[RELN  れる-PASSIVE]
 [OBJE [[RELN  開催する-1]
assertive: Inform


[[RELN  S-REQUEST]
```

```
 [OBJE [[RELN  INFORMIF]
directive: Askq
----------------
directive: Askq
----------------
directive: Askq
----------------
directive: Askq
----------------
directive: Request/Askq
----------------
directive: Askq
indirect directive: Request (discount)
----------------
directive: Askq
indirect directive: Request/Direct (refund)
----------------
directive: Askq (information)
directive: Request (permission, help)


[[RELN  S-REQUEST]
 [OBJE [[RELN  INFORMREF]
commissive: Offer
--------------------
directive: Askq (factual question)
indirect directive: Request (help)
----------------------------------
directive: Askq
---------------
directive: Askq
---------------
directive: Askq


[[RELN  せる-CAUSATIVE]
 [OBJE [[RELN 知る-1]
assertive: Inform
commissive: Promise

 [OBJE [[RELN  させる-PERMISSIVE]
        [OBJE [[RELN  送る-1]
commissive: Promise


[[RELN  さようなら-GOOD-BYE]
expressive: Say-Goodbye

[[RELN 知る-1]
(CAUS) assertive: Inform
commissive: Promise


[[RELN  失礼する-CLOSE_DIALOGUE]
expressive: Say-Goodbye
-----------------------
expressive: Say-Goodbye
-----------------------
expressive: Say-Goodbye
-----------------------
expressive: Say-Goodbye


[[RELN  そうです-CONFIRMATION]
declarative: Confirm
--------------------
declarative: Confirm
--------------------
declarative: Confirm


[[RELN  そうですか-CONFIRMATION]
[[RELN  そうですか-CONFIRMATION]
expressive: Acknowledge
-----------------------
expressive: Acknowledge
```

```
(expressive: Complain)


[[RELN た-PERFECTIVE]
 [OBJE [[RELN ありがとう-THANKING]
expressive: Thank
-----------------
expressive: Thank
expressive: Say-Goodbye


 [OBJE [[RELN  たい-DESIRE]
assertive: Inform
indirect directive: Request (help, advice)


 [OBJE [[RELN  たい-DESIRE]
        [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
directive: Request
assertive: Inform
indirect directive: Askq


 [OBJE [[RELN  ている-PROGRESSIVE]
ね: directive: Askq


[[RELN  ている-PROGRESSIVE]
 [OBJE [[RELN  用意する-1]
assertive: Inform
(expressive: Welcome)


 [[RELN  ている-STATIVE]
assertive: Inform (want)
indirect directive: Request (help)
-----------------------------------
assertive: Inform
------------------
assertive: Inform


[[RELN  ている-STATIVE]
 [OBJE [[RELN  れる-PASSIVE]
assertive: Inform


[[RELN  てもらう-RECEIVE_FAVOR]
 [OBJE [[RELN  させる-PERMISSIVE]
        [OBJE [[RELN  送る-1]
commissive: Promise


[OBJE [[RELN  てもらう-RECEIVE_FAVOR]
directive: Request
assertive: Inform
indirect directive: Askq
--------------------------
assertive: Inform


[OBJE [[RELN  分かる-2]
assertive: Inform
(expressive: Complain)
possible indirect directive: Request


[[RELN  分かった-CONFIRMATION]
------------------------
[[RELN  た-PERFECTIVE]
 [OBJE [[RELN  分かる-1]
expressive: Acknowledge
assertive: Inform
------------------------
expressive: Acknowledge
------------------------
```

```
expressive: Acknowledge
------------------------
expressive: Acknowledge
------------------------
expressive: Acknowledge
------------------------
expressive: Acknowledge
------------------------
expressive: Acknowledge
expressive: Thank


[[RELN  予定だ-1]
 [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
assertive: Inform


 [OBJE [[RELN  用意する-1]
assertive: Inform
(expressive: Welcome)
```

C Translation of Logical Forms from Feature Structures

This section presents an example of the FS-LF version 1.3 system translating conversation 1 from Nadine feature structures into FOPC-style logical forms. The information contained in the logical forms is based on examples taken from a paper by Hobbs and Kameyama [HK90]. For instance, the following is an excerpt from this paper:

> "(2) The Tokyo office called.
>
> ...The logical form is something like
>
> (3) $(\exists e, x, o, b) call'(e, x) \wedge person(x) \wedge rel(x, o) \wedge office(o) \wedge nn(t, o) \wedge Tokyo(t)$
>
> That is, there is a calling event e by a person x related somehow (possibly by identity) to the explicit subject of the sentence o, which is an office and bears some unspecified relation nn to t which is Tokyo."

This general format was used in constructing the translation rules (Appendix D). For instance, each verb is followed by case information (such as `person(x)` that describes the type of its arguments. Also, constants are described by truth functions $(\exists t(Tokyo(t)))$ instead of simply being listed as they are $(\exists Tokyo)$. However, the quantified event variable accompanying every instantiated verb e was ignored, as it seemed to be especially redundant. The verb functions shown here implicitly reference events. If it is desired, each verb function can be modified to explicitly reference an event by inserting an e variable into every verb function and an $\wedge event(e)$ case function after every verb function call.

It is important to note that the style of the logical forms employed here is *not* fixed, but can be changed to suit the tastes of the system that will use the FS-LF translator's output. For instance, it would be just as easy to use an infix notation (`(person x)`) instead of a prefix notation (`person(x)`), or to change the logical forms so that they don't present verb-argument case information, or to insert other required information. However, it is not very useful to experiment with this, until a definite system to use the output of the FS-LF translator has been specified. Further comments on the use of the FS-LF system are presented in Section 4.3.

```
**   FS-LF DEMO   **
**Translating utterances from number 0 to 19.**


0:(FS-to-LF

 [[RELN もしもし-OPEN_DIALOGUE]
  [AGEN [[LABEL *GUEST*]]]
  [RECP [[LABEL *OFFICE*]]]]
  )
    ...gives:

((∃ X Y) SAYS-MOSHI-MOSHI (X Y) ∧ PERSON (X) ∧ PERSON (Y) ∧ IS (X (*GUEST*)) ∧ IS
 (Y (*OFFICE*)))



1:(FS-to-LF

 [[RELN S-REQUEST]
  [AGEN !X3[[LABEL *GUEST*]]]
  [RECP !X2[[LABEL *OFFICE*]]]
  [OBJE [[RELN INFORMIF]
         [AGEN !X2]
         [RECP !X3]
         [OBJE [[RELN だ-IDENTICAL]
                [OBJE [[RESTR !X2]]]
                [IDEN [[PARM !X1[]]
                       [RESTR [[RELN NAMED]
                               [ENTITY !X1]
                               [IDEN 会議事務局-1]]]]]]]]]]]
  )
    ...gives:

((∃ X Y Z) REQUEST (X Y Z) ∧ PERSON (X) ∧ PERSON (Y) ∧ ACTION (Z) ∧ IS (X (*GUEST*)) ∧ IS
 (Y (*OFFICE*)) ∧ IS
 (Z
  ((∃ X2 Y2 Z2) INFORMING-IF-ACTION (X2 Y2 Z2) ∧ PERSON (X2) ∧ PERSON (Y2) ∧ PROPOSITION (Z2)
   ∧ IS (X2 (*OFFICE*)) ∧ IS (Y2 (*GUEST*)) ∧ IS
   (Z2
    ((∃ M N) IDENTICAL (M N) ∧ IS (M (*OFFICE*)) ∧ IS
     (N ((∃ N2) ENTITY (N2) ∧ HAS-NAME (N2 会議事務局-1))))))))



2:(FS-to-LF

 [[RELN はい-AFFIRMATIVE]
  [AGEN [[LABEL *OFFICE*]]]
  [RECP [[LABEL *GUEST*]]]]
  )
    ...gives:

((∃ X Y) SAYS-HAI-AFFIRMATIVE (X Y) ∧ PERSON (X) ∧ PERSON (Y) ∧ IS (X (*OFFICE*)) ∧ IS
 (Y (*GUEST*)))



3:(FS-to-LF

 [[RELN そうです-CONFIRMATION]
  [AGEN [[LABEL *OFFICE*]]]
  [RECP [[LABEL *GUEST*]]]]
  )
    ...gives:

((∃ X Y) SAYS-SOU-DESU-CONFIRMATION (X Y) ∧ PERSON (X) ∧ PERSON (Y) ∧ IS (X (*OFFICE*)) ∧ IS
 (Y (*GUEST*)))



4:(FS-to-LF

 [[RELN S-REQUEST]
```

```
  [AGEN ! X6[[LABEL *OFFICE*]]]
  [RECP ! X5[[LABEL *GUEST*]]]
  [OBJE [[RELN INFORMREF]
         [AGEN ! X5]
         [RECP ! X6]
         [OBJE [[PARM ! X4[[PARM ! X2[[PARM ! X1[]]
                                     [RESTR [[RELN どのよう-1]
                                             [ENTITY ! X1]]]]]
                          [RESTR [[RELN だ-IDENTICAL]
                                  [OBJE ! X2]
                                  [IDEN [[PARM ! X3[]]
                                         [RESTR [[RELN 用件-1]
                                                 [ENTITY ! X3]]]]]]]]]
                [RESTR [[RELN だ-IDENTICAL]
                        [OBJE ! X4]
                        [IDEN []]]]]]]]]
  )
    ...gives:

((∃ X Y Z) REQUEST (X Y Z) ∧ PERSON (X) ∧ PERSON (Y) ∧ ACTION (Z) ∧ IS (X (*OFFICE*)) ∧ IS
 (Y (*GUEST*)) ∧ IS
 (Z
  ((∃ X2 Y2 Z2) INFORMING-REFERENCE-ACTION (X2 Y2 Z2) ∧ PERSON (X2) ∧ PERSON (Y2) ∧
   PROPOSITION (Z2) ∧ IS (X2 (*GUEST*)) ∧ IS (Y2 (*OFFICE*)) ∧ IS
   (Z2 ((∃ M N) IDENTICAL (M N) ∧ IS (M (どのよう-WHAT-KIND-OF)) ∧ IS (N (用件-1))))))


5:(FS-to-LF

 [[RELN が-MODERATE]
  [OBJE [[RELN たい-DESIRE]
         [EXPR ! X2[]]
         [OBJE [[RELN 申込む-1]
                [AGEN ! X2]
                [SLOC [[PARM ! X1[]]
                       [RESTR [[RELN 会議-1]
                               [ENTITY ! X1]]]]]]]]]]
  )
    ...gives:

(GA-MODERATION
 (((∃ X Z) WANTS-TO-DO (X Z) ∧ PERSON (X) ∧ ACTION (Z) ∧ IS (X NIL) ∧ IS
   (Z
    ((∃ X W) ATTEND-PLACE (X W) ∧ PERSON (X) ∧ SPATIAL-LOCATION (W) ∧ IS (X NIL) ∧ IS (W 会議-1))
    ))))


6:(FS-to-LF

 [[RELN S-REQUEST]
  [AGEN ! X7[[LABEL *GUEST*]]]
  [RECP ! X6[[LABEL *OFFICE*]]]
  [OBJE [[RELN INFORMREF]
         [AGEN ! X6]
         [RECP ! X7]
         [OBJE [[PARM ! X5[[PARM ! X2[[PARM ! X1[]]
                                     [RESTR [[RELN どのよう-1]
                                             [ENTITY ! X1]]]]]
                          [RESTR [[RELN だ-IDENTICAL]
                                  [OBJE ! X2]
                                  [IDEN [[PARM ! X3[]]
                                         [RESTR [[RELN 手続-1]
                                                 [AGEN []]
                                                 [ENTITY ! X3]]]]]]]]]
                [RESTR [[RELN ばよい-SHOULD]
                        [AGEN ! X4[]]
                        [OBJE [[RELN する-1]
                               [AGEN ! X4]
                               [OBJE ! X5]]]]]]]]]]
  )
    ...gives:

((∃ X Y Z) REQUEST (X Y Z) ∧ PERSON (X) ∧ PERSON (Y) ∧ ACTION (Z) ∧ IS (X (*GUEST*)) ∧ IS
 (Y (*OFFICE*)) ∧ IS
 (Z
  ((∃ X2 Y2 Z2) INFORMING-REFERENCE-ACTION (X2 Y2 Z2) ∧ PERSON (X2) ∧ PERSON (Y2) ∧
   PROPOSITION (Z2) ∧ IS (X2 (*OFFICE*)) ∧ IS (Y2 (*GUEST*)) ∧ IS
   (Z2 ((∃ M N) IDENTICAL (M N) ∧ IS (M (どのよう-WHAT-KIND-OF)) ∧ IS (N (手続-1))))))


7:(FS-to-LF

 [[RELN 下さい-REQUEST]
  [AGEN [[LABEL *OFFICE*]]]
  [RECP !X3[[LABEL *GUEST*]]]
  [OBJE [[RELN する-1]
         [AGEN !X3]
         [OBJE [[PARM !X2[]]
                [RESTR [[RELN 手続-1]
                        [AGEN []]
                        [ENTITY !X2]]]]]
         [INST [[PARM !X1[]]
                [RESTR [[RELN 登録用紙-1]
                        [ENTITY !X1]]]]]]]]
  )
   ...gives:

((∃ X Y Z) REQUEST (X Y Z) ∧ PERSON (X) ∧ PERSON (Y) ∧ ACTION (Z) ∧ IS (X (*OFFICE*)) ∧ IS
 (Y (*GUEST*)) ∧ IS
 (Z
  ((∃ X2 Z2 W2) DO-WITH (X2 Z2 W2) ∧ PERSON (X2) ∧ ACTION (Z2) ∧ INSTRUMENT (W2) ∧ IS
   (X2 (*GUEST*)) ∧ IS (Z2 手続-1) ∧ IS (W2 登録用紙-1))))


8:(FS-to-LF

 [[RELN S-REQUEST]
  [AGEN !X3[[LABEL *OFFICE*]]]
  [RECP !X4[[LABEL *GUEST*]]]
  [OBJE [[RELN INFORMIF]
         [AGEN !X4]
         [RECP !X3]
         [OBJE [[RELN 持つ-1]
                [AGEN !X3]
                [OBJE [[PARM !X2[]]
                       [RESTR [[RELN 登録用紙-1]
                               [ENTITY !X2]]]]]
                [TLOC [[PARM !X1[]]
                       [RESTR [[RELN 既に-1]
                               [ENTITY !X1]]]]]]]]]]
  )
   ...gives:

((∃ X Y Z) REQUEST (X Y Z) ∧ PERSON (X) ∧ PERSON (Y) ∧ ACTION (Z) ∧ IS (X (*OFFICE*)) ∧ IS
 (Y (*GUEST*)) ∧ IS
 (Z
  ((∃ X2 Y2 Z2) INFORMING-IF-ACTION (X2 Y2 Z2) ∧ PERSON (X2) ∧ PERSON (Y2) ∧ PROPOSITION (Z2)
   ∧ IS (X2 (*GUEST*)) ∧ IS (Y2 (*OFFICE*)) ∧ IS
   (Z2
    ((∃ X2 Z2) HAVE-OBJECT (X2 Z2) ∧ PERSON (X2) ∧ OBJECT (Z2) ∧ IS (X2 (*OFFICE*)) ∧ IS
     (Z2 登録用紙-1))))))


9:(FS-to-LF

 [[RELN いいえ-NEGATIVE]
  [AGEN [[LABEL *GUEST*]]]
  [RECP [[LABEL *OFFICE*]]]]
  )
   ...gives:

((∃ X Y) SAYS-IIE-NEGATIVE (X Y) ∧ PERSON (X) ∧ PERSON (Y) ∧ IS (X (*GUEST*)) ∧ IS
 (Y (*OFFICE*)))


10:(FS-to-LF
 [[RELN だ-STATEMENT]
  [OBJE [[PARM !X1[]]
         [RESTR [[RELN まだ-1]
                 [ENTITY !X1]]]]]]
  )
    ...gives:

((∃ X Y) SAYS-MADA-DESU (X Y) ∧ PERSON (X) ∧ PERSON (Y))


11:(FS-to-LF

 [[RELN た-PERFECTIVE]
  [OBJE [[RELN 分かる-1]
         [EXPR []]
         [OBJE []]]]]
  )
    ...gives:

(PERFECTIVE-TENSE (((∃ X Z) UNDERSTAND (X Z) ∧ PERSON (X) ∧ CONCEPT (Z))))


12:(FS-to-LF

 [[RELN 送る-1]
  [AGEN []]
  [RECP []]
  [OBJE [[PARM !X2[]]
         [RESTR [[RELN 登録用紙-1]
                 [ENTITY !X2]]]]]
  [INFMANN [[PARM !X1[]]
            [RESTR [[RELN それでは-1]
                    [ENTITY !X1]]]]]]
  )
    ...gives:

((∃ X Y Z) SENDS (X Y Z) ∧ PERSON (X) ∧ PERSON (Y) ∧ OBJECT (Z) ∧ IS (X NIL) ∧ IS (Y NIL) ∧ IS
 (Z 登録用紙-1))


13:(FS-to-LF

 [[RELN 願う-REQUEST]
  [RECP []]
  [EXPR []]
  [OBJE [[RELN と-COORDINATE]
         [ARG-1 [[PARM !X2[]]
                 [RESTR [[RELN 住所-1]
                         [ENTITY !X2]]]]]
         [ARG-2 [[PARM !X1[]]
                 [RESTR [[RELN 名前-1]
                         [ENTITY !X1]]]]]]]]
  )
    ...gives:

FS-LF: handle-mapping-problem:  ERROR: No value found!!
((∃ X Y Z) REQUESTING-ACTION (X Y Z) ∧ MANNER-OF-RESPONSE (NIL) ∧ PERSON (X) ∧ PERSON (Y) ∧
 ACTION (Z) ∧ SOMEHOW-RELATED (Z ((住所-1) ∧ (名前-1))))


14:(FS-to-LF

 [[RELN だ-IDENTICAL]
  [OBJE [[PARM ?X05]
         [RESTR [[RELN 住所-1]
                 [ENTITY ?X05]]]]]
  [IDEN [[PARM !X4[[PARM !X3[[PARM !X2[[PARM !X1[]]
                                        [RESTR [[RELN NAMED]
                                                [ENTITY !X1]
                                                [IDEN 二十三]]]]]
                              [RESTR [[RELN NAMED]
                                      [ENTITY !X2]
```

```
                                   [IDEN 茶屋町]]]]]
                   [RESTR [[RELN NAMED]
                           [ENTITY ! X3]
                           [IDEN 北区]]]]]
         [RESTR [[RELN NAMED]
                 [ENTITY ! X4]
                 [IDEN 大阪市]]]]]]
 )
   ...gives:

((∃ M N) IDENTICAL (M N) ∧ IS (M (住所-1)) ∧ IS (N ((∃ N2) ENTITY (N2) ∧ HAS-NAME (N2 大阪市))))


15:(FS-to-LF

 [[RELN だ-IDENTICAL]
  [OBJE [[PARM ?X02]
         [RESTR [[RELN 名前-1]
                 [ENTITY ?X02]]]]]
  [IDEN [[PARM ! X1[]]
         [RESTR [[RELN NAMED]
                 [ENTITY ! X1]
                 [IDEN 鈴木真弓]]]]]]
 )
   ...gives:

((∃ M N) IDENTICAL (M N) ∧ IS (M (名前-1)) ∧ IS (N ((∃ N2) ENTITY (N2) ∧ HAS-NAME (N2 鈴木真弓))))


16:(FS-to-LF

 [[RELN た-PERFECTIVE]
  [OBJE [[RELN 分かる-1]
         [EXPR []]
         [OBJE []]]]]
 )
   ...gives:

(PERFECTIVE-TENSE (((∃ X Z) UNDERSTAND (X Z) ∧ PERSON (X) ∧ CONCEPT (Z))))


17:(FS-to-LF

 [[RELN てもらう-RECEIVE_FAVOR]
  [AGEN []]
  [RECP ! X4[]]
  [OBJE [[RELN させる-PERMISSIVE]
         [AGEN ! X4]
         [RECP ! X3[]]
         [OBJE [[RELN 送る-1]
                [AGEN ! X3]
                [RECP []]
                [OBJE [[PARM ! X1[]]
                       [RESTR [[RELN 登録用紙-1]
                               [ENTITY ! X1]]]]]
                [MANN [[PARM ! X2[]]
                       [RESTR [[RELN 至急に-1]
                               [ENTITY ! X2]]]]]]]]]]
 )
   ...gives:

((∃ X1 Y1 Z1) RECEIVES-FAVOR-FROM (X1 Y1 Z1) ∧ PERSON (X1) ∧ PERSON (Y1) ∧ FAVOR (Z1) ∧ IS
 (X1 NIL) ∧ IS (Y1 NIL) ∧ IS
 (Z1
  ((∃ X2 Y2 Z2) PERMITS-TO-DO (X2 Y2 Z2) ∧ PERSON (X2) ∧ PERSON (Y2) ∧ ACTION (Z2) ∧ IS
   (X2 NIL) ∧ IS (Y2 NIL) ∧ IS
   (Z2
    ((∃ X Y Z) SENDS (X Y Z) ∧ PERSON (X) ∧ PERSON (Y) ∧ OBJECT (Z) ∧ IS (X NIL) ∧ IS (Y NIL)
     ∧ IS (Z 登録用紙-1))))))


18:(FS-to-LF
[[RELN 願う-REQUEST]
 [RECP []]
 [EXPR []]
 [OBJE []]
 [INFMANN [[PARM ! X1[]]
           [RESTR [[RELN よろしく-1]
                   [ENTITY ! X1]]]]]]
 )
   ...gives:

((∃ X Y Z) REQUESTING-ACTION (X Y Z) ∧ MANNER-OF-RESPONSE (よろしく-1) ∧ PERSON (X) ∧ PERSON (Y) ∧
 ACTION (Z) ∧ SOMEHOW-RELATED (Z NIL))


19:(FS-to-LF

 [[RELN 失礼する-CLOSE_DIALOGUE]
  [AGEN [[LABEL *GUEST*]]]
  [RECP [[LABEL *OFFICE*]]]
  [INFMANN [[PARM ! X1[]]
            [RESTR [[RELN それでは-1]
                    [ENTITY ! X1]]]]]]
  )
    ...gives:

((∃ X Y) CLOSES-DIALOG-WITH-SHITSUREI-SURU (X Y) ∧ PERSON (X) ∧ PERSON (Y) ∧ IS (X (*GUEST*))
 ∧ IS (Y (*OFFICE*)))


Time taken to translate 20 utterances using Fast-FS-TO-LF, version 1.3:
FS-LF: handle-mapping-problem:  ERROR: No value found!!
Evaluation of (LOOP FOR UTTERANCE-NUM FROM 0 TO ...) took 0.684091 seconds of elapsed time includ
ing:
  0.013 seconds processing sequence breaks,
  0.204 seconds in the storage system (including 0.160 seconds waiting for pages):
    0.194 seconds processing 132 page faults including 2 fetches,
    0.010 seconds in creating and destroying pages, and
    0.000 seconds in miscellaneous storage system tasks.
1,176 list, 2,492 structure, 198 stack words consed in WORKING-STORAGE-AREA.
6 list words consed in *WHO-CALLS-DATABASE-AREA*.

<end of demo>
```

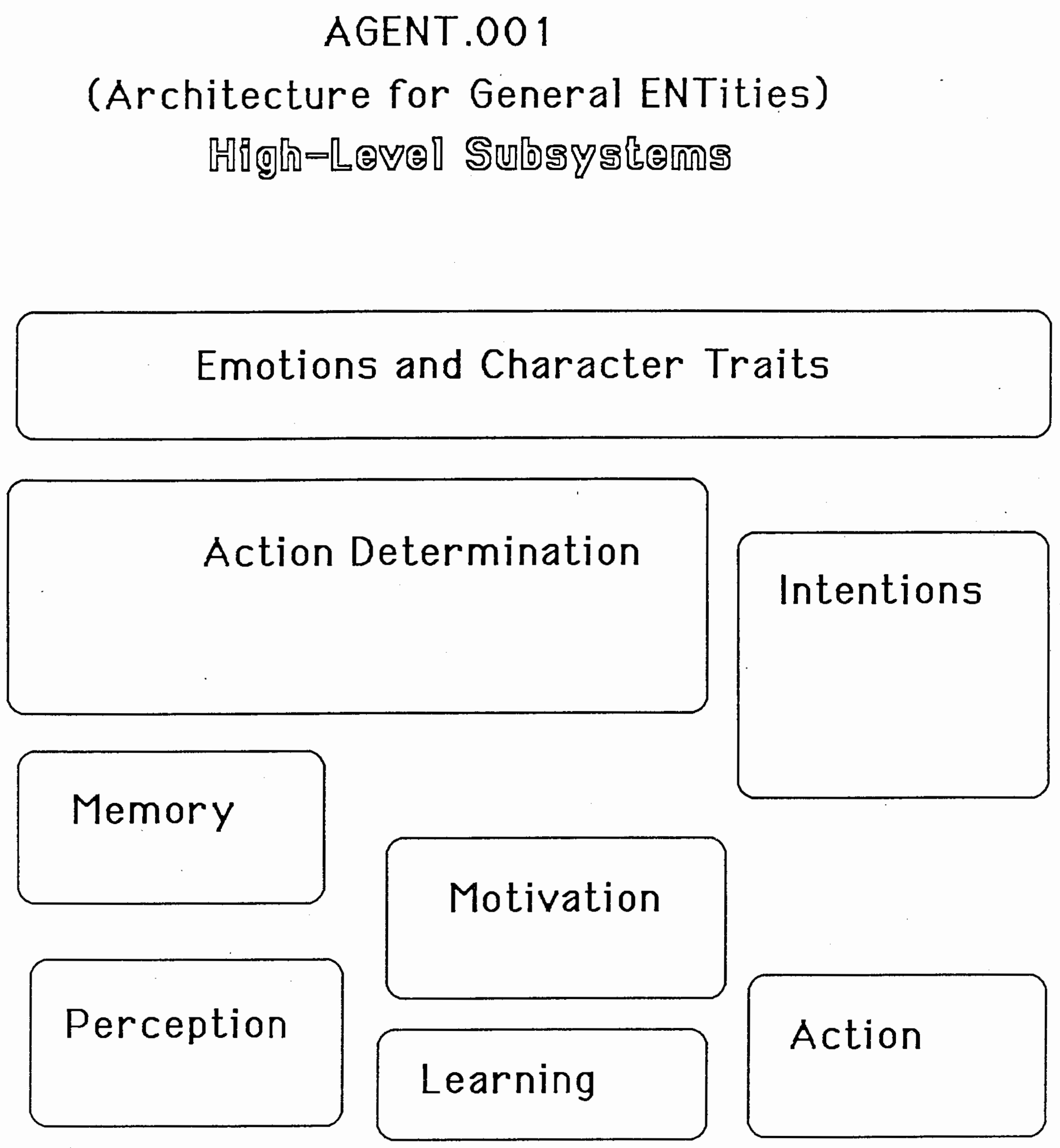
D FS-LF Specifications Used to Translate The Example in Appendix C

```
;;;  -*- Mode: LISP; Syntax: Common-lisp; Package: USER; Base: 10 -*-

;;;      NAME:            FS-LF-SPECS-EX4                LM01:>myers>fs-lf-specs-ex4.lisp.12
;;;
;;;      VERSION:         1.0
;;;
;;;      WHAT IT DOES:    Provides an example of the FS-LF translation specifications.
;;;
;;;
;;;      PROBLEMS IT:     Show an example of how FOPC logical forms could be used.
;;;      IS DESIGNED      ***Note that the actual format of the logical forms is a matter****
;;;      TO SOLVE         ***of taste, and can be changed to fit the needs of the customer***
;;;                       *******************************************************************
;;;
;;;      WHAT YOU NEED:   LM01:>myers>convn>conv1-ex-real.lisp
;;;                       LM01:>myers>FS-2-LF.bin   plus supporting files.
;;;
;;;      USR INTERFACES:  Run this file using   fs-lf-demo5.
;;;
;;;      HOW IT WORKS:    Note that ?vars that are not in the [FS] are a no-no.
;;;
;;;      FUTURE EXTNS.:   See FS-2-LF.
;;;
;;;      HISTORY
;;;      -------
;;;      Mar 19 '90       New RWS reader does not like "&"s.  Changed all "&"s to "+"s.
;;;      Jul 17 '90       Converted a couple functions over to FOPC to test it out.
;;;                       Results:  ∃,∧ works nicely but : didn't work for some reason.
;;;                       Put a ' in front of them, did not get a chance to test it out again.
;;;                       FOPC appears to be monotonic and will have the traditional hairy problems
;;;                       of how to represent time.
;;;                       Also, semantic meaning is up for grabs.  $kolem constants? vars?
;;;      Sept '90         Copied over into EX4.
;;;      Nov 5 '90        Final test version of EX4.
;;;
;;; To Do:
;


(reset-FS-LF)

;-------------------------------------------------------------------

(fs-to-lf-spec

 [[LABEL ?name]]

  (?name)
)
;-------------------------------------------------------------------

(fs-to-lf-spec

[[RELN  もしもし-OPEN_DIALOGUE]
 [AGEN ?s1]
 [RECP ?s2]]

((∃ x y) says-moshi-moshi(x y) ∧ person(x) ∧ person(y) ∧ is(x ?s1) ∧ is(y ?s2))

)
;-------------------------------------------------------------------
(fs-to-lf-spec

[[RELN  S-REQUEST]
 [AGEN ?s1]
 [RECP ?s2]
 [OBJE ?s3]]


((∃ x y z) request(x y z) ∧ person(x) ∧ person(y) ∧ action(z)
 ∧ is(x ?s1) ∧ is(y ?s2) ∧ is(z ?s3))
)
;-----------------------------------------------------------------
(fs-to-lf-spec

  [[RELN  INFORMIF]
        [AGEN ?s1]
        [RECP ?s2]
        [OBJE ?s3]]

((∃ x2 y2 z2) informing-if-action(x2 y2 z2) ∧ person(x2) ∧ person(y2) ∧ proposition(z2)
 ∧ is(x2 ?s1) ∧ is(y2 ?s2) ∧ is(z2 ?s3))
)
;-----------------------------------------------------------------
(fs-to-lf-spec

[[RELN  だ-IDENTICAL]
  [IDEN [[PARM []]
         [RESTR ?something]]]
  [OBJE [[RESTR ?something-else]]]]]]]

((∃ m n) identical(m n) ∧ is(m ?something-else) ∧ is(n ?something))
)

;-----------------------------------------------------------------
(fs-to-lf-spec

[[RELN   NAMED]
 [ENTITY ?something]
 [IDEN   ?name]]

((∃ n2) entity(n2) ∧ has-name(n2 ?name) )

)
;-----------------------------------------------------------------
(fs-to-lf-spec

[[RELN  はい-AFFIRMATIVE]
 [AGEN ?sp1]
 [RECP ?sp2]]

((∃ x y) says-hai-affirmative(x y) ∧ person(x) ∧ person(y) ∧ is(x ?sp1) ∧ is(y ?sp2))

)
;----------------------------------------------------------------
(fs-to-lf-spec

[[RELN  そうです-CONFIRMATION]
 [AGEN ?sp1]
 [RECP ?sp2]]

((∃ x y) says-sou-desu-confirmation(x y) ∧ person(x) ∧ person(y) ∧ is(x ?sp1) ∧ is(y ?sp2))

)
;---------------------------------------------------------------
(fs-to-lf-spec


       [[RELN  INFORMREF]
        [AGEN ?s1]
        [RECP ?s2]
        [OBJE [[PARM !X05[[PARM !X04[[PARM !X03[]]
                                    [RESTR [[RELN ?wh-question]
                                            [ENTITY !X03]]]]]
                          [RESTR ?something-is-something]]]]]

((∃ x2 y2 z2) informing-reference-action(x2 y2 z2) ∧ person(x2) ∧ person(y2) ∧ proposition(z2)
 ∧ is(x2 ?s1) ∧ is(y2 ?s2) ∧ is(z2 ?something-is-something))
)
;----------------------------------------------------------------------------------
(fs-to-lf-spec
[[RELN どのよう-1]]

(どのよう-what-kind-of)
)
;----------------------------------------------------------------------------------
(fs-to-lf-spec
[[RELN 用件-1]
 [ENTITY ?entity]]
(用件-1)
)
;-------------------------------------------------------------------------------

(fs-to-lf-spec
[[RELN  が-MODERATE]
 [OBJE ?rest]]

(Ga-moderation (?rest))
)

;-------------------------------------------------------------------------------
(fs-to-lf-spec

 [[RELN  たい-DESIRE]
        [EXPR ?agent]
        [OBJE ?action]]

((∃ x z) wants-to-do(x z) ∧ person(x) ∧ action(z)
 ∧ is(x ?agent) ∧ is(z ?action))

)

;-------------------------------------------------------------------------------
(fs-to-lf-spec

[[RELN  申込む-1]
 [AGEN ?agent]
 [SLOC [[PARM ! X02[]]
        [RESTR [[RELN  ?place]
                [ENTITY ! X02]]]]]]]

((∃ x w) attend-place(x w) ∧ person(x) ∧ spatial-location(w)
 ∧ is(x ?agent) ∧ is(w ?place) )

)

;-------------------------------------------------------------------------------
(fs-to-lf-spec
[[RELN  手続-1]
 [AGEN []]
 [ENTITY ?entity]]

(手続-1)
)

;---------------------------------------------------------------------------------
(fs-to-lf-spec

[[RELN  下さい-REQUEST]
 [AGEN ?s1]
 [RECP ?s2]
 [OBJE ?s3]]

((∃ x y z) request(x y z) ∧ person(x) ∧ person(y) ∧ action(z)
 ∧ is(x ?s1) ∧ is(y ?s2) ∧ is(z ?s3))

)
;---------------------------------------------------------------------------------
(fs-to-lf-spec

[[RELN  する-1]
 [AGEN ?s1]
 [OBJE [[PARM ! X01[]]
        [RESTR [[RELN  ?s3]
                [AGEN []]
                [ENTITY ! X01]]]]]
 [INST [[PARM ! X02[]]
        [RESTR [[RELN  ?s2]
                [ENTITY ! X02]]]]]]]

((∃ x2 z2 w2) do-with(x2 z2 w2) ∧ person(x2) ∧ action(z2) ∧ instrument(w2)
 ∧ is(x2 ?s1) ∧ is(z2 ?s3) ∧ is(w2 ?s2))
```

```
)
;----------------------------------------------------------------------
(fs-to-lf-spec

              [[RELN  持つ-1]
               [AGEN ?s1]
               [OBJE [[PARM ! X05[]]
                      [RESTR [[RELN ?s3]
                              [ENTITY ! X05]]]]]]

((∃ x2 z2) have-object(x2 z2) ∧ person(x2) ∧ object(z2) ∧ is(x2 ?s1) ∧ is(z2 ?s3))

)
;----------------------------------------------------------------------
(fs-to-lf-spec

[[RELN  いいえ-NEGATIVE]
 [AGEN ?s1]
 [RECP ?s2]]

((∃ x y) says-iie-negative(x y) ∧ person(x) ∧ person(y) ∧ is(x ?s1) ∧ is(y ?s2))

)
;---------------------------------------------------------------------------
(fs-to-lf-spec

[[RELN  だ-STATEMENT]
 [OBJE [[PARM ! X01[]]
        [RESTR [[RELN  まだ-1]
                [ENTITY ! X01]]]]]]

((∃ x y) says-mada-desu(x y) ∧ person(x) ∧ person(y))
)
;----------------------------------------------------------------------
(fs-to-lf-spec

[[RELN  送る-1]
 [AGEN ?s1]
 [RECP ?s2]
 [OBJE [[PARM ! X02[]]
        [RESTR [[RELN ?s3]
                [ENTITY ! X02]]]]]]

((∃ x y z) sends(x y z) ∧ person(x) ∧ person(y) ∧ object(z) ∧ is(x ?s1) ∧ is(y ?s2) ∧ is(z ?s3))

)
;----------------------------------------------------------------------
(fs-to-lf-spec

[[RELN  願う-REQUEST]
 [EXPR []]
 [RECP []]
 [OBJE ?action]
 [INFMANN [[PARM ! X01[]]
           [RESTR [[RELN  ?manner]
                   [ENTITY ! X01]]]]]]

  ((∃ x y z) requesting-action(x y z) ∧ manner-of-response(?manner)
   ∧ person(x) ∧ person(y) ∧ action(z)
   ∧ somehow-related(z ?action) )

)

;----------------------------------------------------------------------
(fs-to-lf-spec
       [[RELN  と-COORDINATE]
        [ARG-1 [[PARM ! X01[]]
                [RESTR   ?s1]]]
        [ARG-2 [[PARM ! X02[]]
                [RESTR   ?s2]]]]
(?s1 ∧ ?s2)
)
;----------------------------------------------------------------------
(fs-to-lf-spec
[[RELN  住所-1]]
(住所-1)
```

```
)
;-----------------------------------------------------------------------
(fs-to-lf-spec
[[RELN  名前-1]]
 (名前-1)
)
;--------------------------------------------------------------
(fs-to-lf-spec

[[RELN  た-PERFECTIVE]
 [OBJE ?s]]

 (perfective-tense(?s))
)
;---------------------------------------------------------------
(fs-to-lf-spec
[[RELN  分かる-1]
        [EXPR []]
        [OBJE []]]

((∃ x z) understand(x z) ∧ person(x) ∧ concept(z))

)

;----------------------------------------------------------------
(fs-to-lf-spec

[[RELN  てもらう-RECEIVE_FAVOR]
 [AGEN ?s1]
 [RECP ?s2]
 [OBJE ?s3]]

((∃ x1 y1 z1) receives-favor-from(x1 y1 z1) ∧ person(x1) ∧ person(y1) ∧ favor(z1)
 ∧ is(x1 ?s1) ∧ is(y1 ?s2) ∧ is(z1 ?s3))

)

;----------------------------------------------------------------
(fs-to-lf-spec

[[RELN  させる-PERMISSIVE]
        [AGEN ?s1]
        [RECP ?s2]
        [OBJE ?s3]]

((∃ x2 y2 z2) permits-to-do(x2 y2 z2) ∧ person(x2) ∧ person(y2) ∧ action(z2)
 ∧ is(x2 ?s1) ∧ is(y2 ?s2) ∧ is(z2 ?s3))

)

;----------------------------------------------------------------
(fs-to-lf-spec

[[RELN  失礼する-CLOSE_DIALOGUE]
 [AGEN ?s1]
 [RECP ?s2]]

((∃ x y) closes-dialog-with-shitsurei-suru(x y) ∧ person(x) ∧ person(y)
 ∧ is(x ?s1) ∧ is(y ?s2))

)
```

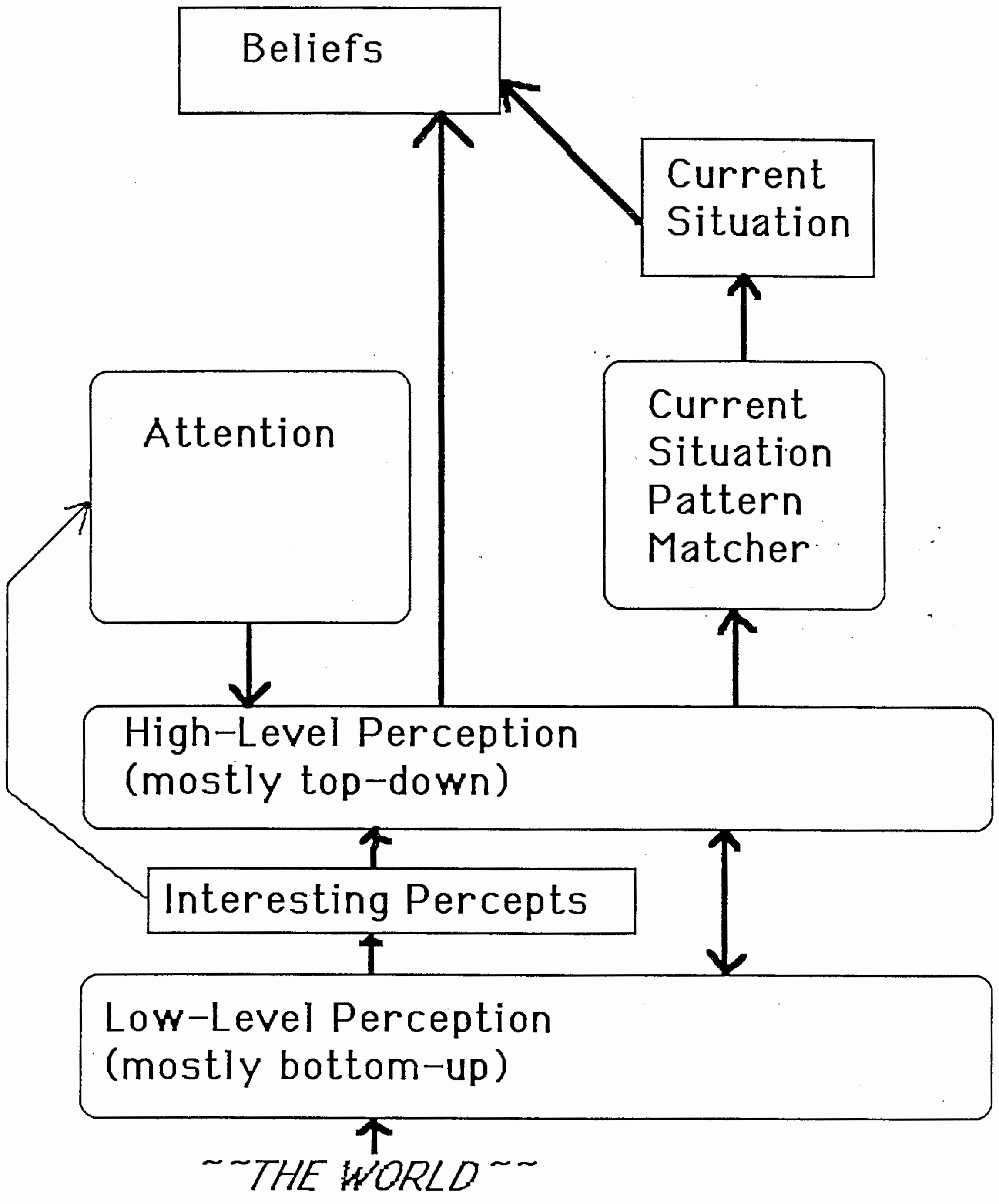
E The Design for AGENT.001: an Architecture for General ENTitites.



AGENT.001

(Architecture for General ENTities)

The Perception Subsystem

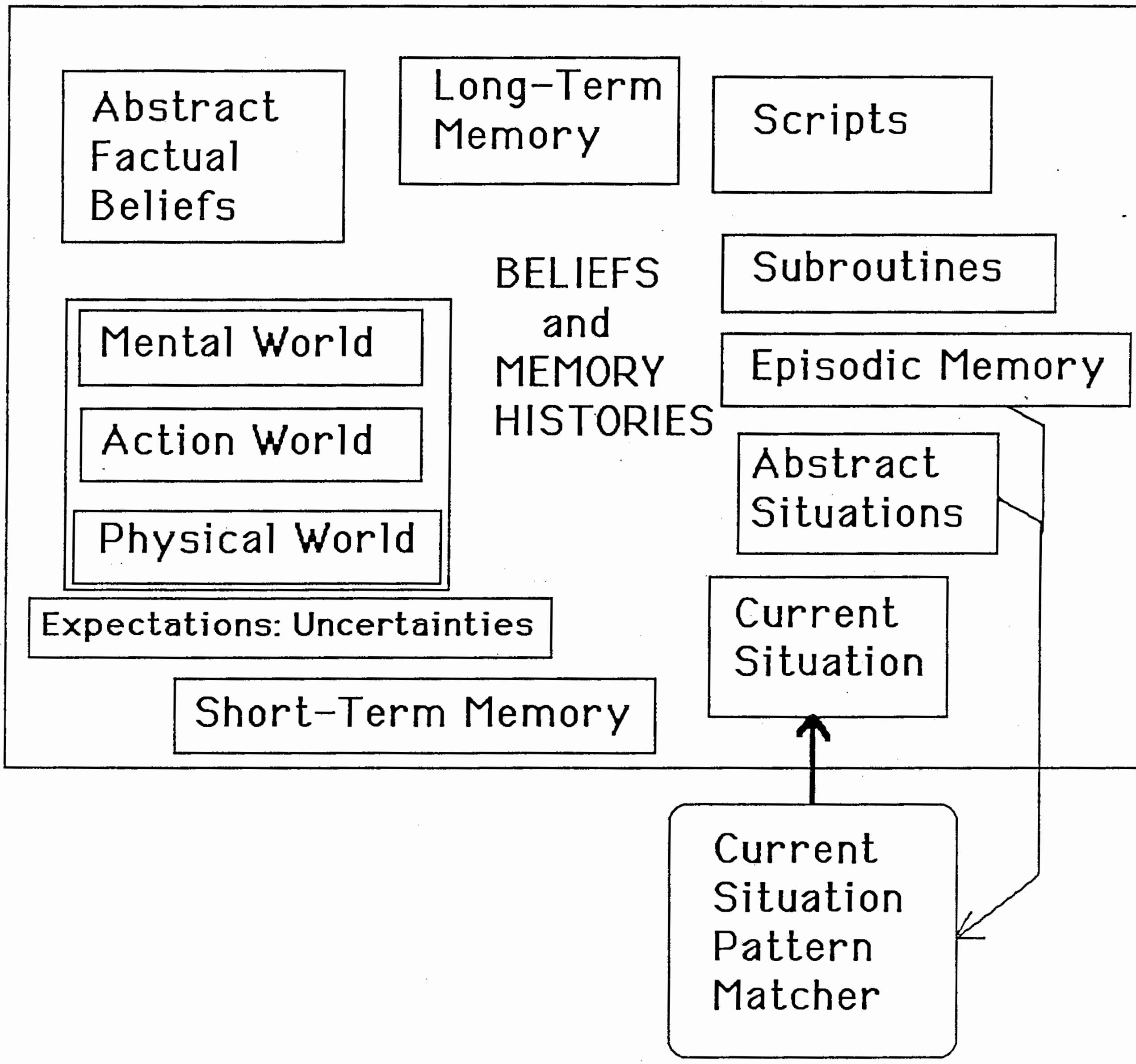


Beliefs
Current Situation
Attention
Current Situation Pattern Matcher
High-Level Perception (mostly top-down)
Interesting Percepts
Low-Level Perception (mostly bottom-up)
~~THE WORLD~~

AGENT.001

(Architecture for General ENTities)

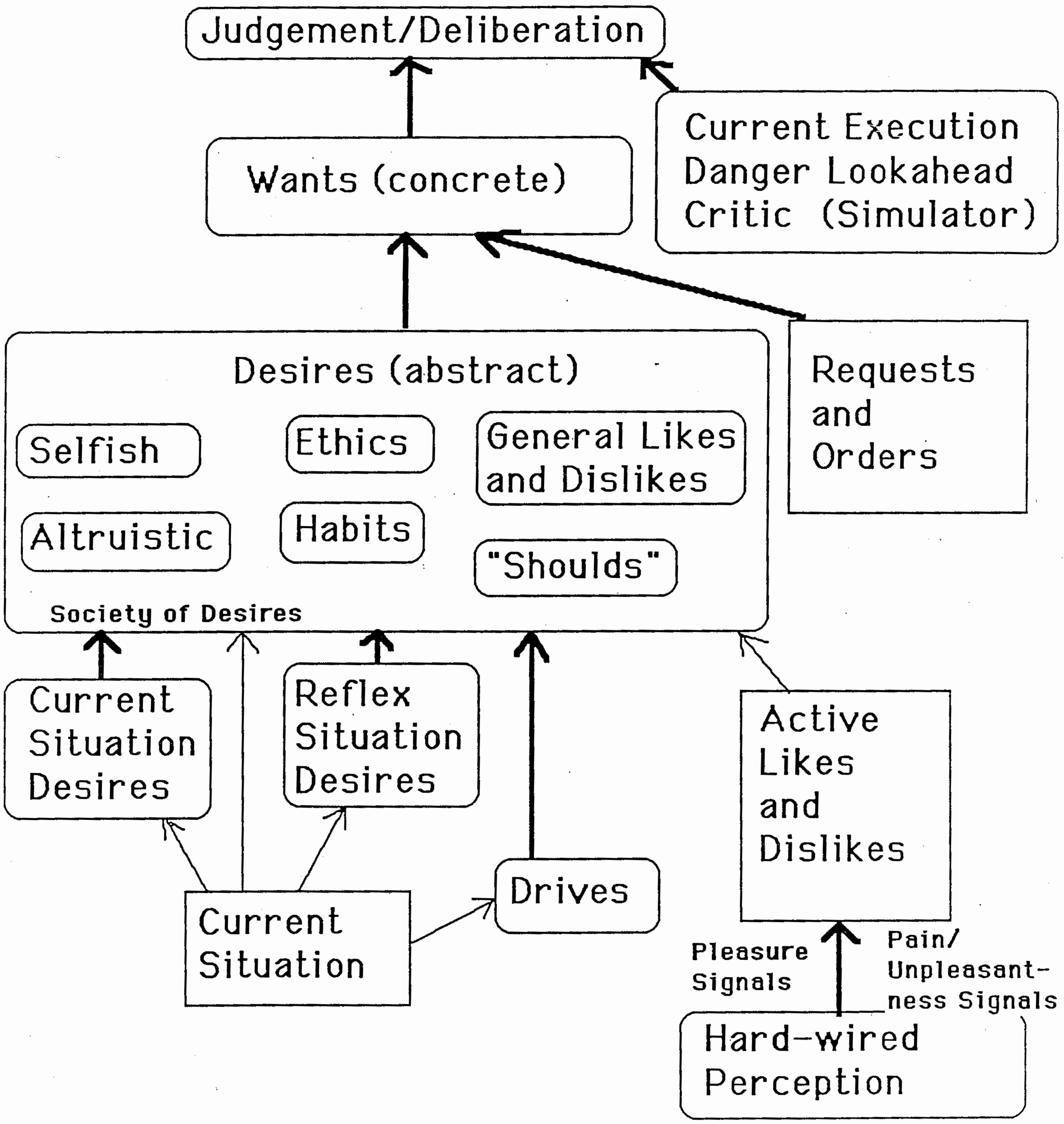
The Memory Subsystem



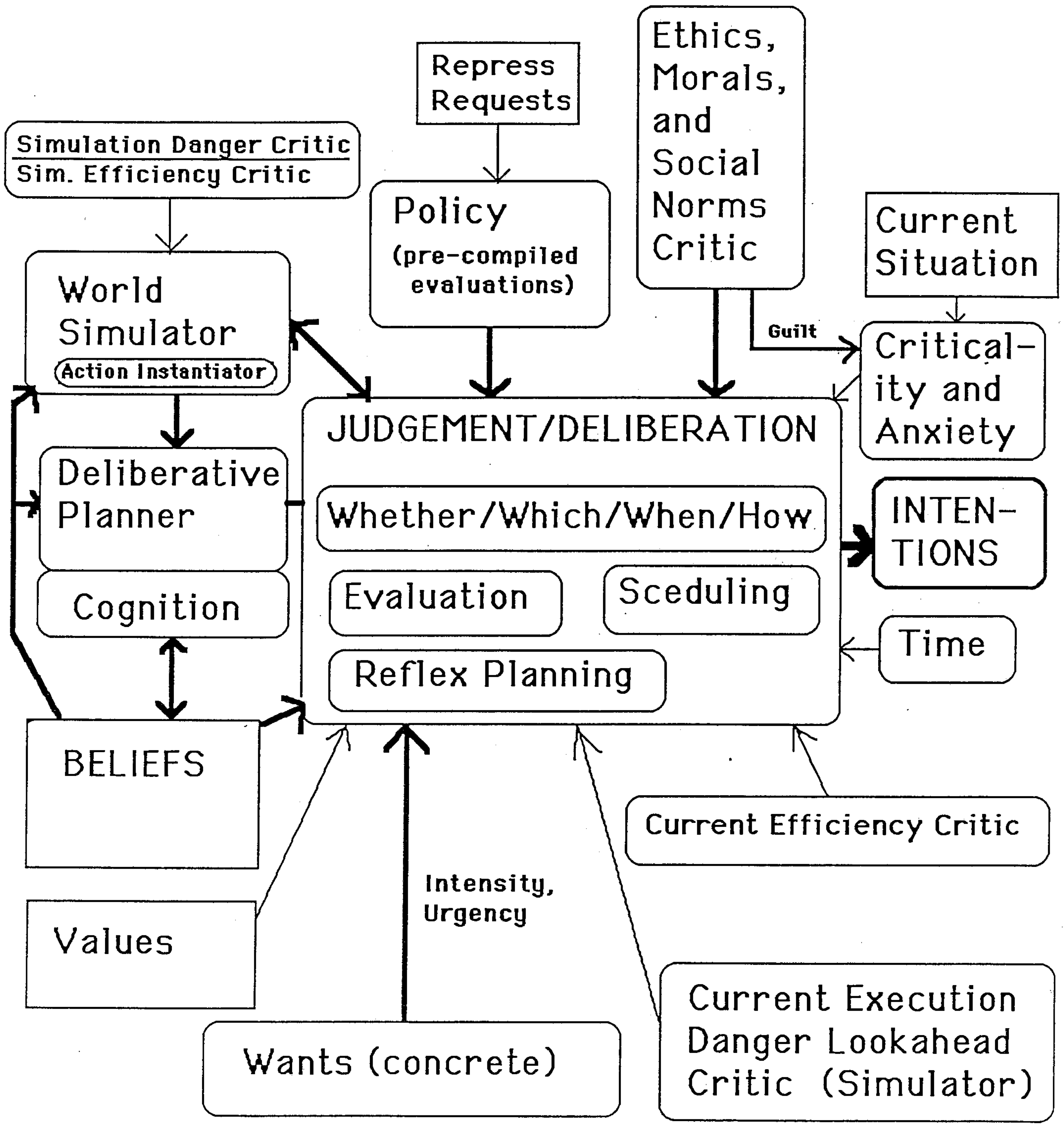
AGENT.001

(Architecture for General ENTities)

The Motivation Subsystem



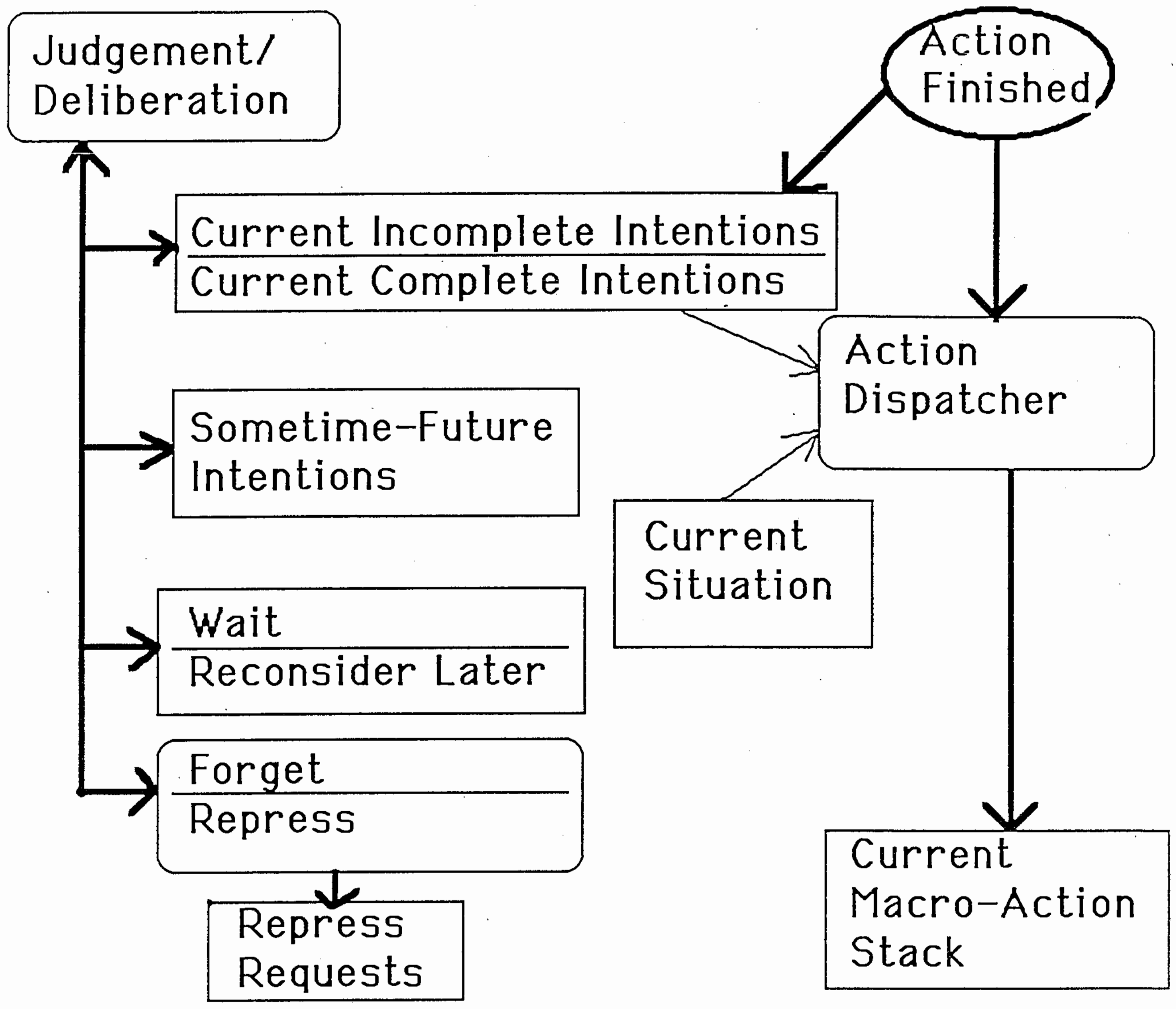
AGENT.001
(Architecture for General ENTities)
The Action Determination Subsystem



AGENT.001

(Architecture for General ENTities)

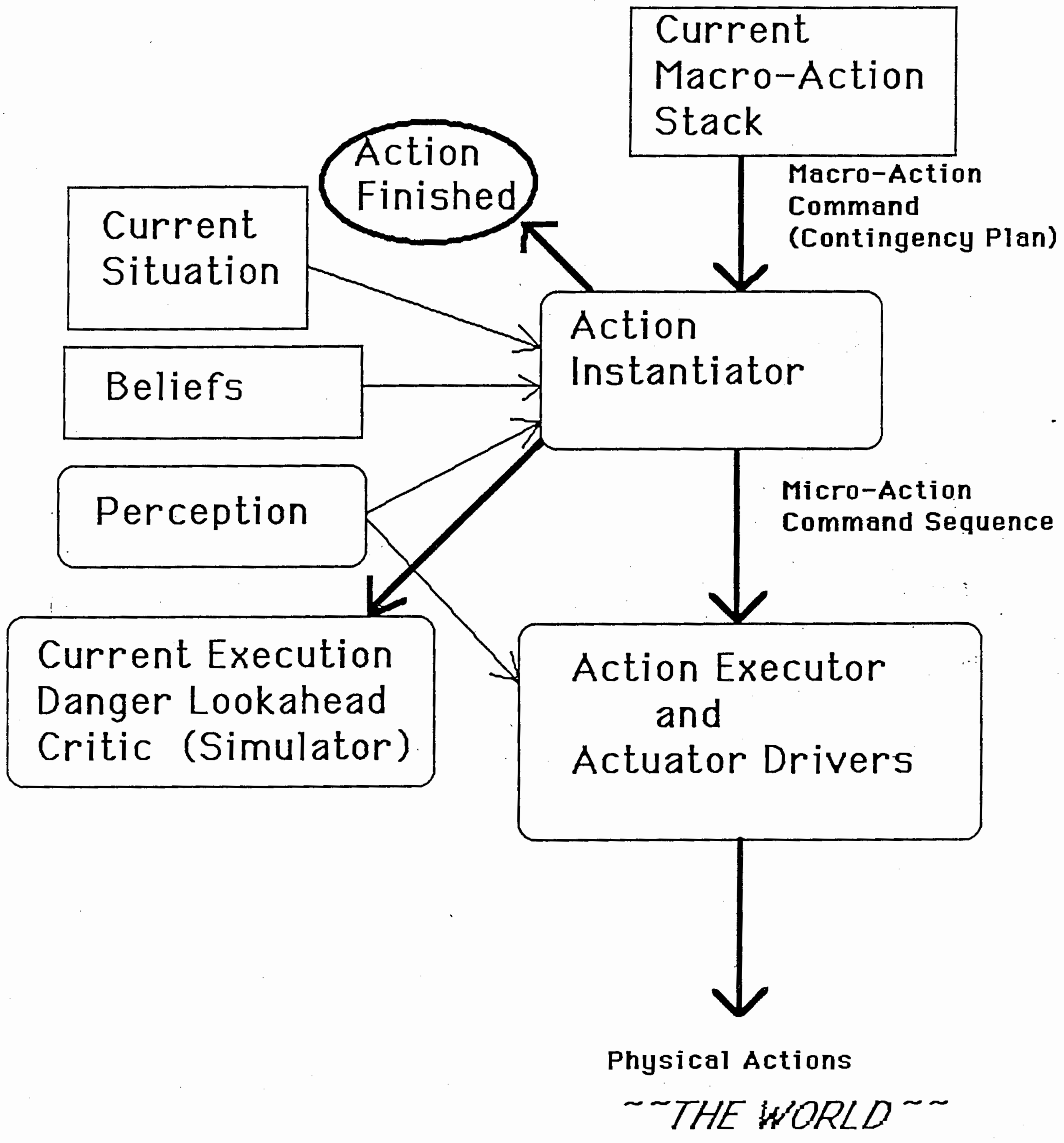
The Intentions Subsystem



AGENT.001

(Architecture for General ENTities)

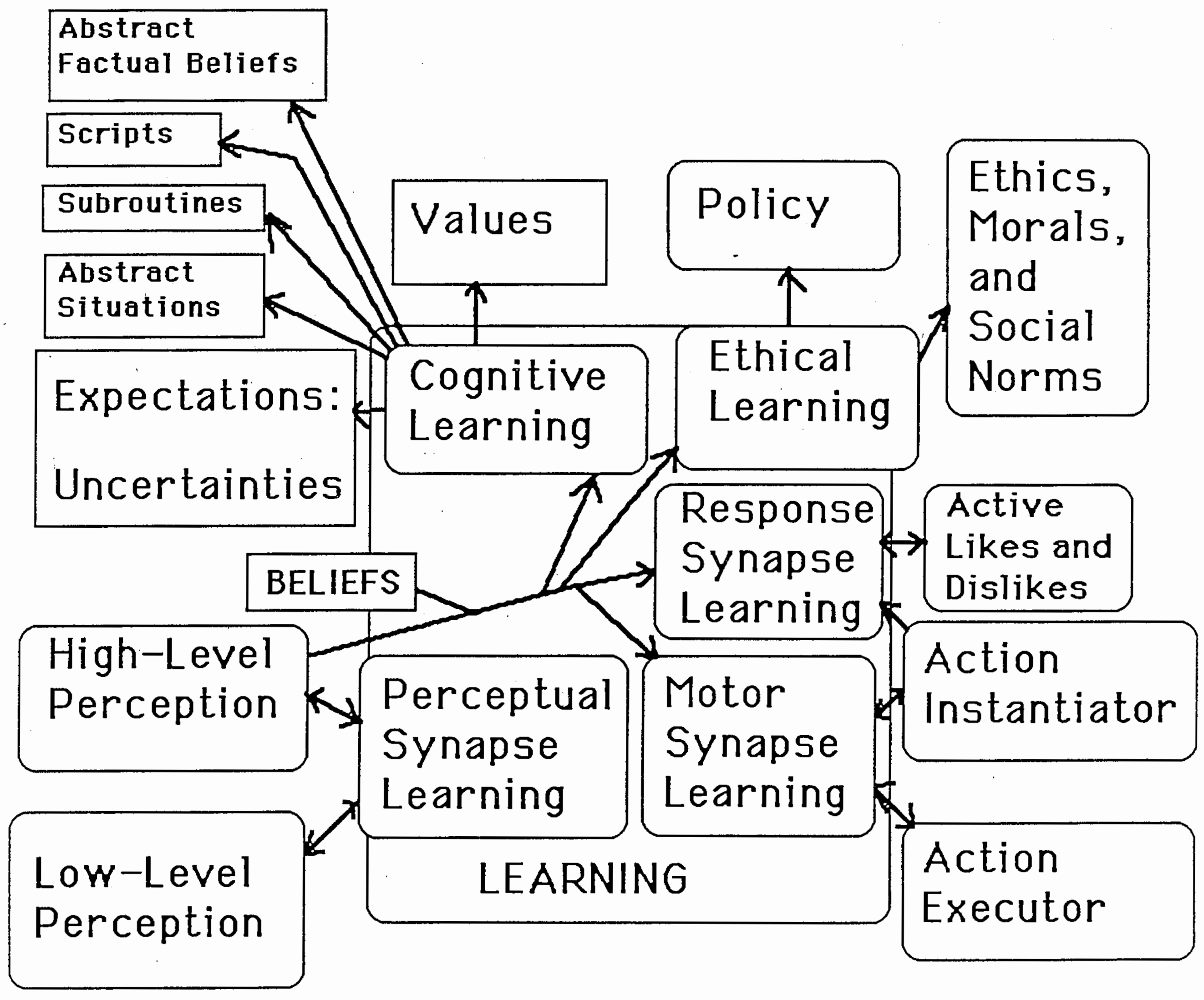
The Action Subsystem


Current Macro-Action Stack
Macro-Action Command (Contingency Plan)
Action Finished
Current Situation
Beliefs
Perception
Action Instantiator
Micro-Action Command Sequence
Current Execution Danger Lookahead Critic (Simulator)
Action Executor and Actuator Drivers
Physical Actions
~~THE WORLD~~

AGENT.001

(Architecture for General ENTities)

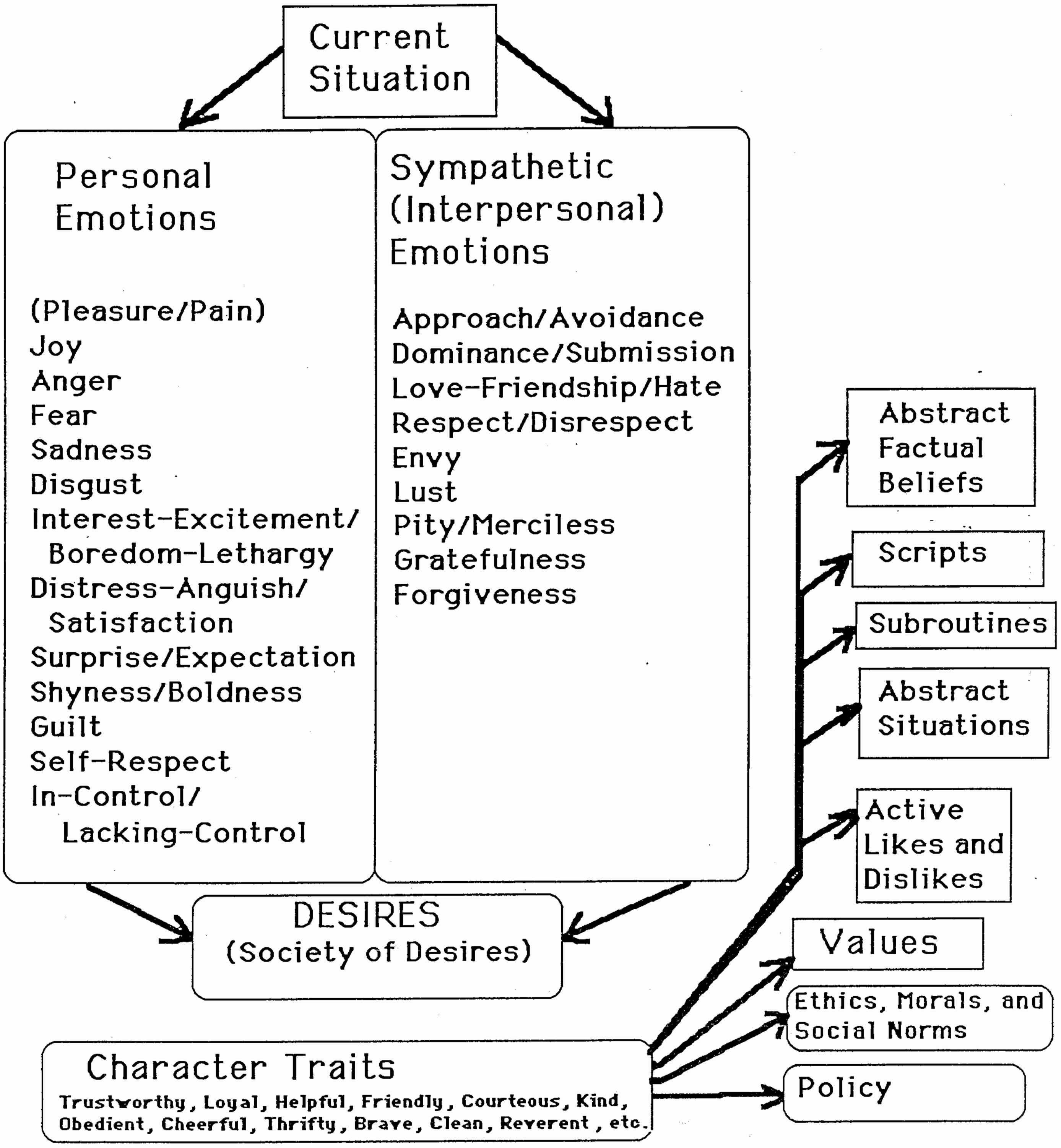
The **Learning** Subsystem



AGENT.001

(Architecture for General ENTities)

The Emotions and Character Traits Subsystem



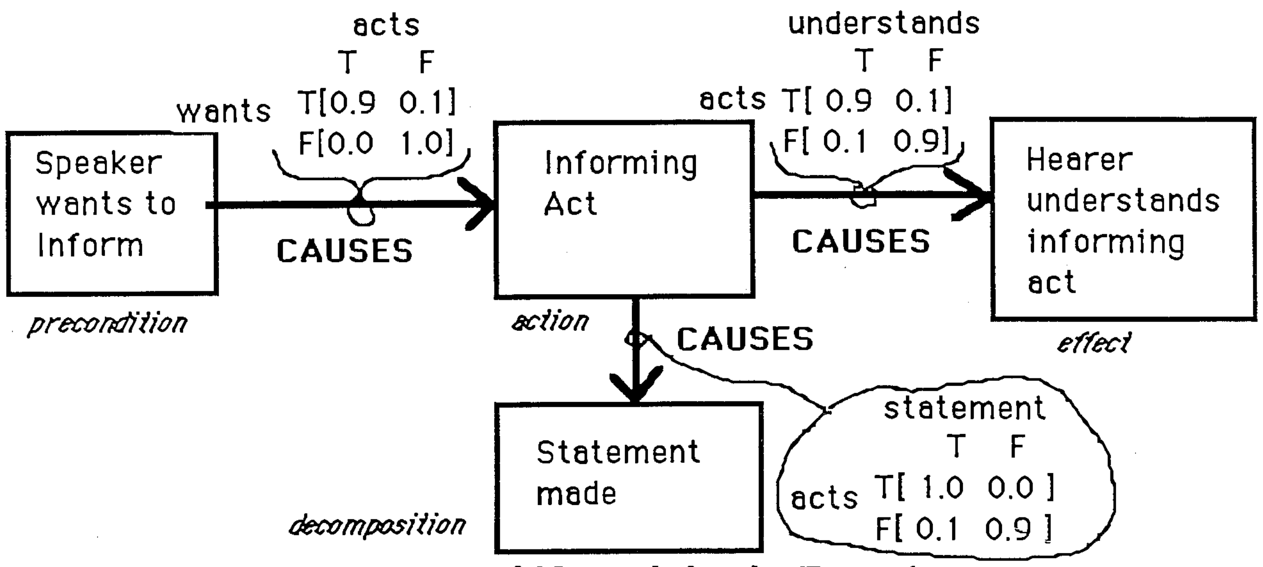


Figure 4: **Causal Network for the Example**

F A Preliminary Demonstration of CAER, the CAusal Evidential Reasoning System

The CAER system is shown performing probabilistic deductive (predictive) and abductive (diagnostic) evidential reasoning. The system performs belief updating on a causal network. CAER version 0.5 is restricted to the representation of causal trees, where each conceptual node is allowed to have only a single parent. Nevertheless, it is possible to demonstrate an example from a realistic problem.

The problem is one instance of the "unobservable precondition" difficulty discussed in Section 3.2. The current NP system, which is based on deductive logic requiring observable preconditions, is not powerful enough to handle this example. The problem concerns an action `informing-act`. To simplify the example, there is one precondition, one decomposition, and one effect for this action. The precondition is `the-speaker-wants-to-inform`. Note that this is a mental attitude of the speaker, and is unobservable by an outside system. The decomposition is `a-statement-is-made-by-the-speaker`. The speaker performs the illocutionary informing act by means of a locutionary statement act. The effect is `the hearer-understands-the-informing-act`. The effect follows from the illocutionary action having been performed.

A CAER network consists of a set of conceptual nodes, that are linked in a network using "causes" links. The nodes in this example will be the four concepts discussed above. Each node can take on one of a disjunctive set of symbolic values from its own value-set. For the purposes of this demonstration, each node will take on a value from the two-member set {TRUE, FALSE}. However, it is not necessary that the symbolic values of a value-set be logical, and there is no logical restriction to the finite number of values in the set. For instance, a value-set could be {WANTS, DOES-NOT-WANT, CAN'T-DECIDE}. Each conceptual node has its own value-set; there is no need for all the nodes to take on the same values.

The nodes are linked using causal links. It is important to specify the causal links' directions so that they correspond with the direction of causation in reality. In the example, the speaker wanting-to-inform causes the speaker to perform an informing-act. The need to perform an informing-act causes the locutionary statement-made action to be performed. The performance of the informing-act causes the hearer to understand the informing-act. Thus, the directions of the causal links are as shown in the illustration.

Each causal link is labeled with a conditional-probability causal matrix. In a causal relationship where node A causes node B ($a \rightarrow b$), the matrix represents the conditional probabilities $p(B|A)$, or, more strictly, $p(B = b \mid A = a)$, permuted for each of the values a and b contained in the respective value-sets $\{a\}$ and $\{b\}$. For instance, in the first link where `speaker-wants-to-inform` (or `wants` for short) implies `informing-act-performed` (or `act` for short), we have the value p(act=TRUE | want=TRUE) in the upper left corner of the matrix, p(act=FALSE | want=TRUE) in the upper right, p(act=TRUE | want=FALSE) in the lower left, and p(act=FALSE | want=FALSE) in the lower right. See Figure 4. The values for the antecedent A vary along the left side of the matrix; the values for the consequent B vary along the top of the matrix. The values for the matricies have been filled in by hand, using reasonable assumptions.

The only other information needed to specify the unactivated network is the background prior probabilities of the root node of the tree, in this case `the-speaker-wants-to-inform`. This has been set to $[0.2, 0.8]$, to reflect the fact that about 20% of the time the speaker will typically want to inform the hearer of something. This information represents what is known without *any* observations having been taken. This prior probability propagates through the tree, and causes the prior probabilities of all of the other nodes.

Note that unlike most logical systems, which have a node take on only one value at a time, an evidential reasoning system must consider all the possible values of a node and their corresponding probabilities. Thus, when reporting the value of a node, it is insufficient to simply output `wants-to-inform:  TRUE`. Instead, it is necessary to output `wants-to-inform:  TRUE = 0.8, FALSE = 0.2`.

F.1 Predictive (Deductive) Reasoning Example

In the first example, the system is first loaded and initialized with its background (prior) probabilities. Then, a certain observation is made that the speaker definitely wants to inform the hearer about something. Such a certain observation could come from internal reasoning, for instance, if it were certainly known that the hearer had just previously requested the speaker to inform the hearer of something. The first print-out shows the prior beliefs of each node in the network. Note that all of the TRUE probabilities are low, and all of the FALSE probabilities are high. Next, the certain observation is made, which changes the belief of the `speaker-wants-to-inform` node. This belief propagates through the system, in a deductive manner. Note that after this observation has been made, all of the downstream nodes have changed their probabilities: now the TRUE probabilities are all high, while the FALSE probabilities are all low.

```
Command: (load "LM01:>myers>CAER-test.lisp")
Loading LM01:>myers>CAER-test.lisp.newest into package USER (really COMMON-LISP-USER)
[CAER-node 1]: SPEAKER-WANTS-TO-INFORM
   TRUE: 0.2  FALSE: 0.8
[CAER-node 2]: INFORMING-ACTION-PERFORMED
   TRUE: 0.17999999  FALSE: 0.82
[CAER-node 3]: STATEMENT-MADE
   TRUE: 0.262  FALSE: 0.738
[CAER-node 4]: HEARER-UNDERSTANDS-INFORMING-ACT
   TRUE: 0.24399999  FALSE: 0.756
T
Command: (certain-observation wants-to-inform-node 1 0)
```

```
[CAER-node 1]: SPEAKER-WANTS-TO-INFORM
   TRUE: 1.0  FALSE: 0.0
[CAER-node 2]: INFORMING-ACTION-PERFORMED
   TRUE: 0.9  FALSE: 0.1
[CAER-node 3]: STATEMENT-MADE
   TRUE: 0.91  FALSE: 0.09
[CAER-node 4]: HEARER-UNDERSTANDS-INFORMING-ACT
   TRUE: 0.82  FALSE: 0.18
NIL
```

F.2 Diagnostic (Abductive) Reasoning Example

In the following example, the same network is again loaded and initialized, as reflected in the first print-out. Then, some uncertain evidence for the `statement-made` node is observed. Since this is an uncertain observation, it combines with the prior probabilities for that node to produce the new beliefs. The new evidence then propagates up the tree in an abductive manner. Notice also that the belief in the `hearer-understands-informing-act` node has been changed, because its antecedent is now believed. Also notice that, even though all the TRUE beliefs are now greater than 0.5, they are still not as strong as the previous example. This results from two reasons. First, in this second case, the prior probabilities still play a significant part in the network's beliefs, because the observation was uncertain. Second, due to the general nature of abductive reasoning, the evidential weight will tend to become more diffuse sooner.

```
(load "LM01:>myers>CAER-test.lisp")
Loading LM01:>myers>CAER-test.lisp.newest into package USER (really COMMON-LISP-U
[CAER-node 1]: SPEAKER-WANTS-TO-INFORM
   TRUE: 0.2  FALSE: 0.8
[CAER-node 2]: INFORMING-ACTION-PERFORMED
   TRUE: 0.17999999  FALSE: 0.82
[CAER-node 3]: STATEMENT-MADE
   TRUE: 0.262  FALSE: 0.738
[CAER-node 4]: HEARER-UNDERSTANDS-INFORMING-ACT
   TRUE: 0.24399999  FALSE: 0.756
T
Command: (uncertain-observation statement-node .99 .01)
[CAER-node 1]: SPEAKER-WANTS-TO-INFORM
   TRUE: 0.6761134  FALSE: 0.32388663
[CAER-node 2]: INFORMING-ACTION-PERFORMED
   TRUE: 0.6680162  FALSE: 0.33198377
[CAER-node 3]: STATEMENT-MADE
   TRUE: 0.9723346  FALSE: 0.027665315
[CAER-node 4]: HEARER-UNDERSTANDS-INFORMING-ACT
   TRUE: 0.634413  FALSE: 0.36558703
NIL
```

These preliminary examples demonstrate that the current CAER system is capable of performing causal evidential reasoning on networks with single-parent nodes. Research is continuing on expanding and strengthening the system.

G Examples of Preliminary Results from Parallel Process Scheduling Research

These subsections show some preliminary results from research in the parallel process scheduling problem. It is assumed that the Automatic Interpretation system is divided up into six subsystems: Speech Recognition, Parsing, Understanding, Transfer, Language Generation, and Speech Generation. Each subsystem may have one or more processes. The subsystem processes are simulated by trivial routines that print out when they start and end.

The system output is governed by an additional output process written by the author that guarantees First In, First Out (FIFO) output for multiple processes. However, for some reason, the Sequent computer occasionally drops characters in its output, so the output is not guaranteed to actually happen nor to be complete. This explains why the output lines occasionally look a little strange. Without the output process, FIFO output does not happen, and the output is very hard to read.

G.1 Fork-And-Join Example

In this example, each subsystem is given three processes. No subsystem can start processing until all of the previous subsystem's processes are finished. Notice that the order of each process in the fork is not determined.

This example demonstrates the ability to start different copies of a process in parallel, and also the ability to wait and force different copies to finish in parallel.

```
<Initial lwp> (test4)
Using 5 processors

What is your input utterance? Kaigi ni moshikomitai no desu ga.

Starting Speech Recognition Process #2.
Starting Speech Recognition Process #1.
Finishing Speech Recognition Process #2.  Took 1/100 seconds.
Finishing Speech Recognition Process #1.  Took 1/100 seconds.
Starting Speech Recognition Process #0.
Finishing Speech Recognition Process #0.  Took 0 seconds.
Starting Parsing Process #2.
Finishing Parsing Process #2.  Took 0 seconds.
Starting Parsing Process #0.
Finishing Parsing Process #0.  Took 1/100 seconds.
Starting Parsing Process #1.
Finishing Parsing Process #1.  Took 0 seconds.
Starting Understanding Process #2.
Finishing Understanding Process #2.  Took 1/100 seconds.
Starting Understanding Process #1.
Finishing Understanding Process #1.  Took 0 seconds.
Starting Understanding Process #0.
Finishing Understanding Process #0.  Took 0 seconds.
```

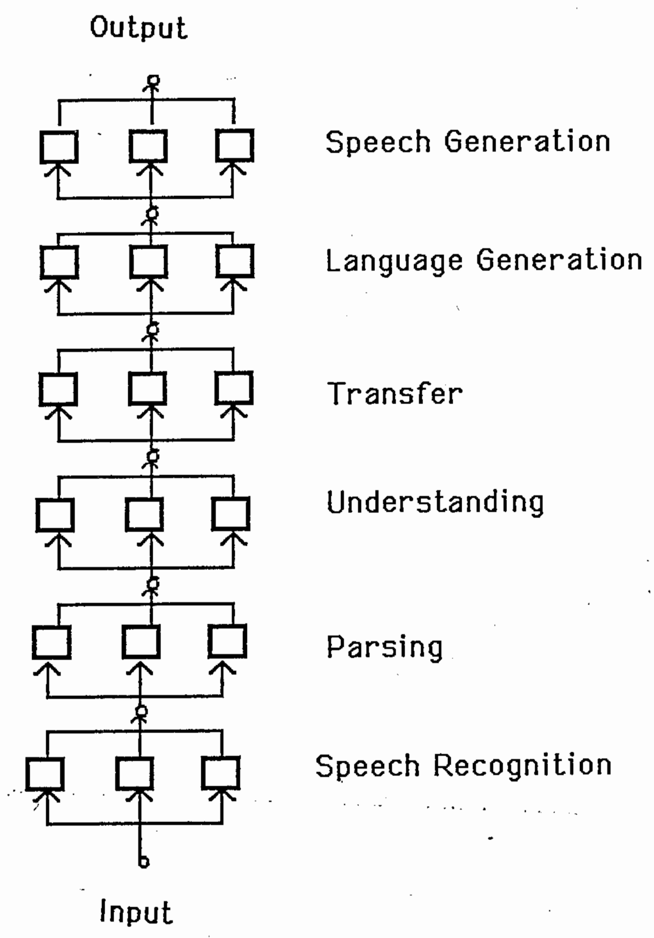


Figure 4: **Fork-And-Join Model Static Scheduling Example**

```
Starting Transfer Process #2.
Finishing Transfer Process #2.  Took 0
Starting Transfer Process #1.
Finishing Transfer Process #1.  Took 0 seconds.
Starting Transfer Process #0.
Finishing Transfer Process #0.  Took 0 seconds.
Starting Language Generation Process #2.
Finishing Language Generation Process #2.  Took 0 seconds.
Starting Language Generation Process #1.
Finishing Language Generation Process #1.  Took 0 seconds.
Starting Language Generation Process #0.
Finishing Language Generation Process #0.  Took 0 seconds.
Starting Speech Generation Process #2.
Finishing Speech Generation Process #2.  Took 0 seconds.
Starting Speech Generation Process #1.
Finishing Speech Generation Process #1.  Took 0 seconds.
Starting Speech Generation Process #0.
Finishing Speech Generation Process #0.  Took 0 seconds.
Finished processing utterance:    Kaigi ni moshikomitai no desu ga. .


What is your input utterance?
```

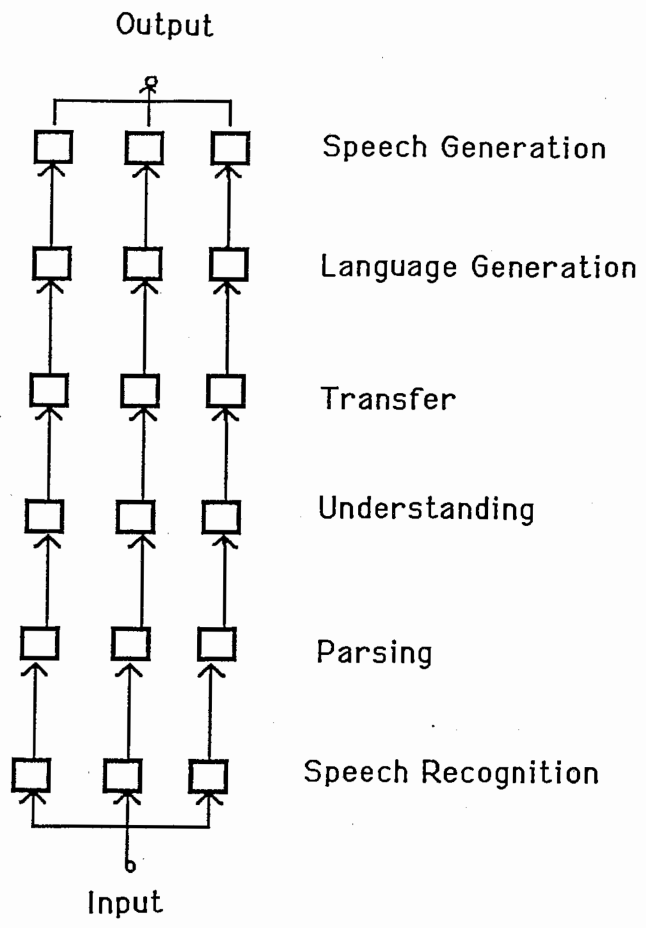


Figure 5: **Parallel-Pipeline Model Static Scheduling Example**

G.2 Parallel-Pipeline Example

In this example, the processing is again split up into three processes for each subsystem. However, the processing proceeds in three parallel pipelines, corresponding to three separate messages. For each message, the processing in one subsystem cannot start until the previous subsystem's processing of that message is finished. However, since the parallel pipelines are asynchronous, there are no constraints on the progress of the other messages through the system. For instance, messages #0 and #2 start parsing before message #1 is finished in speech recognition.

The pipelines are augmented by a three-way fork at the beginning, and a three-way join at the end, so that all the results are collected. Times are from counting to 10,000 in each process.

This example demonstrates the ability to process different messages along asynchronous pipelines in parallel.

```
<Initial lwp> (test5)
Using 5 processors

What is your input utterance? Kaigi ni moshikomitai

Starting Speech Recognition Process #1 (ni).
Starting Speech Recognition Process #2 (moshikomitai).
Starting Speech Recognition Process #0 (Kaigi).
Finishing Speech Recognition Process #2.  Took 227/20 seconds.
Finishing Speech Recognition Process #0.  Took 1137/100 seconds.
Starting Parsing Process #2  (moshikomitai).
Starting Parsing Process #0  (Kaigi).
Finishing Speech Recognition Process #1.  Took 1149/100 seconds.
Starting Parsing Process #1  (ni).
Finishing Parsing Process #0.  Took 541/50 seconds.
Finishing Parsing Process #2.  Took 217/20 seconds.
Starting Understanding Process #0  (Kaigi).
Starting Understanding Process #2  (moshikomitai).
Finishing Parsing Process #1.  Took 56/5 seconds.
Starting Understanding Process #1  (ni).
Finishing Understanding Process #2.  Took 1109/100 seconds.
Starting Transfer Process #2 (moshikomitai).
Finishing Understanding Process #0.  Took 56/5 seconds.
Finishing Understanding Process #1.  Took 268/25 seconds.
Starting Transfer Process #0 (Kaigi).
Starting Transfer Process #1 (ni).
Finishing Transfer Process #2.  Took 1117/100 seconds.
Starting Language Generation Process #2 (moshikomitai).
Finishing Transfer Process #1.  Took 1113/100 seconds.
Starting Language Generation Process #1 (ni).
Finishing Transfer Process #0.  Took 1117/100 seconds.
Starting Language Generation Process #0 (Kaigi).
Finishing Language Generation Process #2.  Took 1119/100 seconds.
Starting Speech Generation Process #2 (moshikomitai).
Finishing Language Generation Process #1.  Took 56/5 seconds.
Starting Speech Generation Process #1 (ni).
Finishing Language Generation Process #0.  Took 56/5 seconds.
Starting Speech Generation Process #0 (Kaigi).
Finishing Speech Generation Process #2.  Took 217/20 seconds.
Finishing Speech Generation Process #1.  Took 1083/100 seconds.
Finishing Speech Generation Process #0217/20 seconds.
Finished processing utterance:    Kaigi ni moshikomitai .


What is your input utterance?
```

G.3 Realistic Static Example

This example combines the fork-and-join technology with the parallel-pipe technology, and presents a realistic example of the type of parallel processing that will probably have to be done in the ATR automatic interpretation system. A single speech-recognition process accepts input, and creates five top candidates for parsing. Each parsing candidate has a score, simulated by choosing a random number between 0 and 99. Each candidate is parsed in a separate parallel parsing process. Each parsing process results in from zero to five successful parses; this is simulated by a number chosen randomly between 0 and 6. Each of the parsed utterances (there are about 15 of them on average) is assigned an

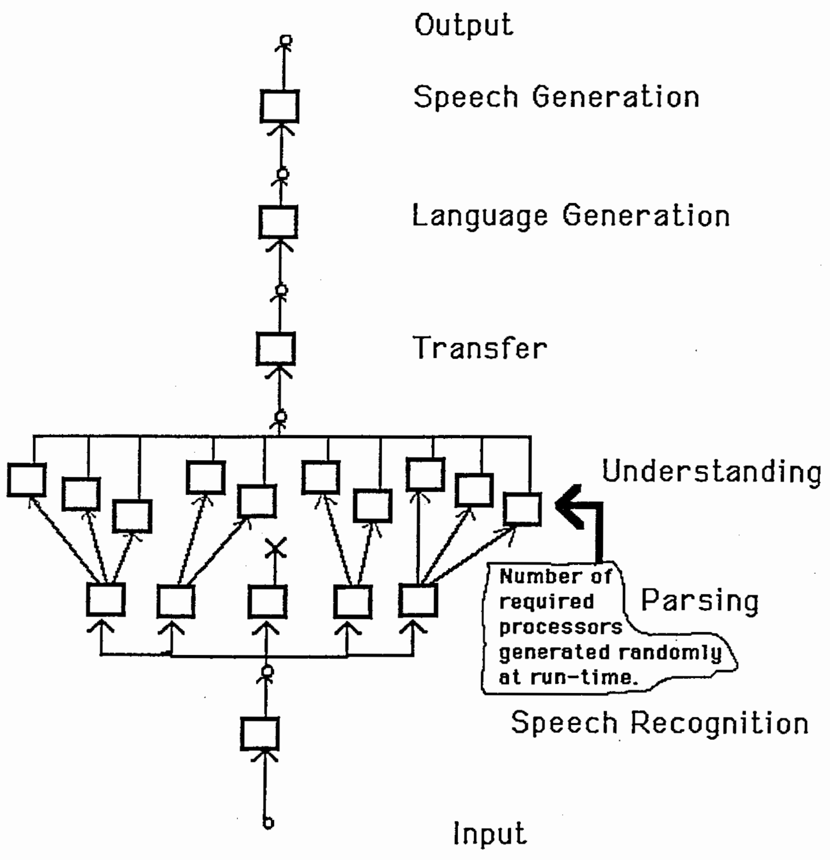


Figure 6: **Realistic Static Scheduling Example, with Random Allocation**

understanding process. The understanding process assigns a new total score to the parsed utterance; this is simulated by adding a random number between 0 and 99 to the score from the speech-recognition process for that candidate, resulting in a random score from 0 to 198. When all of the understanding processes are finished, the top-scoring candidate is chosen and the transfer process starts processing it. Since there is only one top choice, only one transfer process is required. After the transfer process is finished, the language and speech generation subsystems each run once using one process.

The simulation shows that the Understanding subsystem will have to run the most processes of any subsystem in the interpreting telephone, an average of around 15 processes. This means that the Understanding system will have to have the most responsibility for running quickly, or risk causing a bottle-neck. The second-most processed subsystem is the Parser: it might run about 5 times on the average for a single utterance. The Parser must also be especially efficient.

It is important to point out that this is a static scheduling simulation. Although the number of parses and therefore the number of Understanding processes is randomly determined at run-time, each process runs to completion and the system waits for all the processes to report their answers before finishing. A dynamic scheduling algorithm might stochastically decide to stop some processes in the middle, or finish before all the processes have reported their answers. The static scheduling simulation provides a preliminary exercise and demonstration of some of the techniques that will be useful in dynamic scheduling.

```
<Initial lwp> (test6)
Using 5 processors

What is your input utterance? Kaigi ni moshikomitai

Starting Speech Recognition Process #0.
Finishing Speech Recognition Process #0.  Took 1/25 seconds.
Speech Recognition Process #0 has 5 answers.  Creating 5 processes to process the:
Starting Parsing Process #0: score = 46.
Finishing Parsing Process #0.  Took 1/25 seconds.
Parsing Process #0 has 3 answers.  Creating 3 processes to process these.
Starting Parsing Process #1: score = 14.
Finishing Parsing Process #1.  Took 1/25 seconds.
Starting Parsing Process #2: score = 97.
Parsing Process #1 has 2 answers.  Creating 2 processes to process these.
Finishing Parsing Process #2.  Took 1/25 seconds.
Parsing Process #2 has 0 answers.  Creating 0 processes to process these.
Starting Understanding Process #0-0: score = 46 + 20 = 66.
Finishing Understanding Process #0-0.  Score = 66.  Took 1/25 seconds.
Starting Parsing Process #3: score = 51.
Starting Understanding Process #0-1: score = 46 + 42 = 88.
Starting Understanding Process #1-0: score = 14 + 50 = 64.
Finishing Parsing Process #3.  Took 1/25 seconds.
Parsing Process #3 has 2 answers.  Creating 2 processes to process these.
Starting Parsing Process #4: score = 64.
Finishing Understanding Process #0-1.  Score = 88.  Took 3/100 seconds.
Finishing Understanding Process #1-0.  Score = 64.  Took 3/100 seconds.
Finishing Parsing Process #4.  Took 1/25 seconds.
Parsing Process #4 has 3 answers.  Creating 3 processes to process these.
Starting Understanding Process #1-1: score = 14 + 6 = 20.
Starting Understanding Process #0-2: score = 46 + 32 = 78.
Finishing Understanding Process #1-1.  Score = 20.  Took 1/25 seconds.
Finishing Understanding Process #0-2.  Score = 78.  Took 3/100 seconds.
Starting Understanding Process #3-0: score = 51 + 6 = 57.
Starting Understanding Process #3-1: score = 51 + 66 = 117.
Starting Understanding Process #4-0: score = 64 + 93 = 157.
Finishing Understanding Process #3-0.  Score = 57.  Took 3/100 seconds.
Finishing Understanding Process #3-1.  Score = 117.  Took 3/100 seconds.
Finishing Understanding Process #4-0.  Score = 157.  Took 3/100 seconds.
Starting Understanding Process #4-1: score = 64 + 37 = 101.
Finishing Understanding Process #4-1.  Score = 101.  Took 3/100 seconds.
Starting Understanding Process #4-2: score = 64 + 5 = 69.
Finishing Understanding Process #4-2.  Score = 69.  Took 1/25 seconds.
Starting Transfer Process #0.


Transfer: Top Score 157, from Understanding process 4-0.
```

```
Finishing Transfer Process #0.  Took 3/100 seconds.
Starting Language Generation Process #0 (157).
Finishing Language Generation Process #0.  Took 1/25 seconds.
Starting Speech Generation Process #0 (157).
Finishing Speech Generation Process #0.  Took 3/100 seconds.
Finished processing utterance:    Kaigi ni moshikomitai .


What is your input utterance?
```

References

[AI89] Hidekazu Arita and Hitoshi Iida. Tri-layered plan recognition model for dialogue machine translation. Technical Report TR-1-0067, ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, 1989. (in Japanese).

[All87] James Allen. *Natural Language Understanding.* Benjamin/Cummings Publishing Co., Menlo Park, CA, 1987.

[BD89] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In Michael Fehling et al., editor, *Working Notes of the AI and Limited Rationality Symposium, AAAI Spring Symposium Series*, pages 6–9, Stanford University, March 1989.

[BP83] Jon Barwise and John Perry. *Situations and Attitudes.* The MIT Press, Cambridge, Mass., 1983.

[Bra87] Michael E. Bratman. *Intention, Plans, and Practical Reason.* Harvard Univ. Press, Cambridge, MA, 1987.

[Bra90] Michael E. Bratman. What is intention? In Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, chapter 2, pages 15–31. The MIT Press, Cambridge, MA, 1990.

[CL86] Philip R. Cohen and Hector J. Levesque. Persistence, intention, and commitment. In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions & Plans: Proceedings of the 1986 Workshop*, Los Altos, CA, 1986. Morgan Kaufmann Publishers, Inc. Also report CSLI-87-88.

[CL87] Philip R. Cohen and Hector J. Levesque. Intention = choice + commitment. In *AAAI'87: The Sixth National Conference on Artificial Intelligence*, pages 410–415, Seattle, WA, 1987.

[CL90] Philip R. Cohen and Hector J. Levesque. Persistence, intention, and commitment. In Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, chapter 3, pages 33–69. The MIT Press, Cambridge, MA, 1990.

[DB88] Thomas Dean and Mark Boddy. An analysis of time dependent planning. In *AAAI'88: The Seventh National Conference on Artificial Intelligence*, St. Paul, MN, 1988.

[Den87] Daniel C. Dennett. *The Intentional Stance.* The MIT Press, Cambridge, Mass., 1987.

[dK86] Johan de Kleer. An assumption-based tms. *Artificial Intelligence*, 28(2):127–162, March 1986.

[Gol70] Alvin I. Goldman. *A Theory of Human Action.* Prentice-Hall Inc., Englewood Cliffs, NJ, 1970.

[HK90] Jerry R. Hobbs and Megumi Kameyama. Translation by abduction. In *COLING-90, the 13th International Conference on Computational Linguistics*, volume 3, pages 155–161, Helsinki, Finland, August 1990.

[HSME88] Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 95–103, Buffalo, New York, June 1988.

[IK88] Toshihide Ibaraki and Naoki Katoh. *Resource Allocation Problems.* The MIT Press, Cambridge, MA, 1988.

[MN86] Paul H. Morris and Robert A. Nado. Representing actions with an assumption-based truth maintenance system. In *AAAI'86: The Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. Also available as an IntelliCorp (Mountain View, CA) reprint.

[MT90] John K. Myers and Takashi Toyoshima. Known current problems in automatic interpretation: Challenges for language understanding. Technical Report TR-I-0128, ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, January 1990.

[Mye88a] John K. Myers. Action parameter determination under fallible execution. ATR International, Kyoto, Japan, December 1988.

[Mye88b] John K. Myers. Fallible execution. ATR International, Kyoto, Japan, December 1988.

[Mye88c] John K. Myers. The necessity of intentions under fallible execution. ATR International, Kyoto, Japan, December 1988.

[Mye89a] John K. Myers. An assumption-based plan inference system for conversation understanding. In *WGNL Meeting of the IPSJ*, pages 73–80, Okinawa, Japan, June 1989.

[Mye89b] John K. Myers. The atms manual (version 1.1). Technical Report TR-1-0074, ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, February 1989.

[Mye89c] John K. Myers. Np: An assumption-based feature-structure plan inference system. Technical report, ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, December 1989.

[Mye90a] John K. Myers. The fs-lf manual, version 1.2. Technical Report TR-I-0162, ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, April 1990.

[Mye90b] John K. Myers. The meanings of ability utterances with applications to dialog understanding. Technical report, ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, January 1990.

[Mye90c] John K. Myers. Methods for handling spoken interruptions for an interpreting telephone. In *IEICE Technical Report NLC-90*, pages 17–24, Tokyo, Japan, May 1990.

[Mye90d] John K. Myers. Np: An atms-based plan inference system that uses feature structures. In *Meeting of the IPSJ*, pages 438–439, Tokyo, Japan, March 1990.

[Mye90e] John K. Myers. Planning and decision-making with uncertain nondeterministic actions. Forthcoming paper, 1990.

[Mye90f] John K. Myers. A project report on np: An assumption-based nl plan inference system that uses feature structures. In *COLING-90, the 13th International Conference on Computational Linguistics*, volume 3, pages 428–430, Helsinki, Finland, August 1990.

[Mye90g] John K. Myers. Research on understanding at atr, with recommendations for multran (or, how to build intentional autonomous agents). Lecture notes., September 1990.

[NW90a] Peter Norvig and Robert Wilensky. Abduction models for semantic interpretation. In *COLING-90, the 13th International Conference on Computational Linguistics*, volume 3, pages 225–230, Helsinki, Finland, August 1990.

[NW90b] Peter Norvig and Robert Wilensky. Abduction models for semantic interpretation. In Paul O'Rorke et al., editor, *Working Notes of the Automated Abduction Symposium, AAAI Spring Symposium Series*, pages 18–22, Stanford University, March 1990.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.

[Sea85] John R. Searle. *Speech Acts.* Cambridge University Press, Cambridge, Mass., 1985.

[Sid83] Candace L. Sidner. Focusing in the comprehension of definite anaphora. In Michael Brady and Robert C. Berwick, editors, *Computational Models of Discourse*, pages 26–330. The MIT Press, Cambridge, MA, 1983.

[SR81] Roger C. Schank and Christopher K. Riesbeck, editors. *Inside Computer Understanding.* Lawrence Erlbaum Associates, Hillsdale, NJ., 1981.

[SS90] Ryo Stanwood and Masami Suzuki. Some computational applications of lexical functions. Technical Report TR-1-0179, ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, 1990.

[Sti90] Mark E. Stickel. A method for abductive reasoning in natural-language interpretation. In Paul O'Rorke et al., editor, *Working Notes of the Automated Abduction Symposium, AAAI Spring Symposium Series*, pages 5–9, Stanford University, March 1990.

[Suc87] Lucy A. Suchman. *Plans and Situated Actions.* Cambridge University Press, Cambridge, Mass., 1987.

[SV69] John R. Searle and Daniel Vanderveken. *Foundations of Illocutionary Logic.* Cambridge University Press, Cambridge, Mass., 1969.

[WE90] Bonnie Lynn Webber and Barbara Di Eugenio. Free adjuncts in natural language instructions. In *COLING-90, the 13th International Conference on Computational Linguistics*, volume 2, pages 395–400, Helsinki, Finland, August 1990.

[Wie87] Anna Wierzbicka. *English Speech Act Verbs.* Academic Press, Orlando, FL, 1987. Also Australia.