An Information-based Approach for Guiding Multi-Modal Human-Computer-Interaction

Matthias Denecke
Interactive Systems Laboratories
Carnegie Mellon University
Pittsburgh, PA 15217
denecke@cs.cmu.edu

Abstract

Much work has been done in dialogue modeling for spoken and multi-modal human-computer interaction. Problems can arise in situations that do not correspond to the dialogue model. For this reason, we propose *information-centered* dialogue processing in which the actions to be taken by the dialogue system are determined as a function of the information available in the discourse, the database and the domain model. In order to arrive at fully specified representations of the intended actions, the specificity of the representations is increased by unification, integrating information from multi-modal input, database access and domain knowledge. Our approach differs from other state-of-the-art systems in that it does not rely on explicit dialogue models. Instead, we show how partial and underspecified representations of the situation can be used in a spoken dialogue system to generate clarification questions and to guide the user to arrive at his or her communicative goal. We show furthermore how probabilistic information can be used to disambiguate without clarification questions. Evaluation results and dialogue examples demonstrate the flexibility and naturalness of our approach.

1 Introduction

Many state-of-the-art spoken dialogue systems rely on explicit dialogue modeling. Often, modeling is done by describing the dialogue states and the expected response using finite-state automata, dialogue grammars or rules. Typically, the models are built and evaluated using data from the Wizard-of-Oz technique.

However, modeling a dialogue explicitly as a sequence of actions has several drawbacks. First, dialogue models are costly to construct. The dialogue corpus collected have to be large enough to allow generalizations about the dialogue scenario, and considerable human effort is required to create and refine dialogue models. Second, even if the corpus is large enough for building good dialogue models, there is always the sparse data problem. Third, as the number of input modalities increases, so does the complexity of the dialogue model. This situation is exacerbated by the fact that different modalities provide different aspects of information, e.g. a touch screen device typically provides geometric information that must be dealt with appropriately. Fourth, instructions given by the user are often severely underspecified to the degree that the information conveyed by the request alone is not sufficient to perform the intended operation. Fifth, recognition errors in any of the modalities may cause partially inappropriate representations of requests to be generated. Here again, fall-back strategies have to be provided. All these difficulties increase modeling complexity.

To overcome these problems, we propose a departure from the model-based approach to dialogue processing in favor of an *information-centered* approach. We develop and use semantic representations that can be compared by the information they contain. Possible systems actions can thus be determined in function of the degree of specificity of the available information rather than a dialogue state. Additionally, these representations permit situated recovery strategies in cases where the user underspecifies the request or speech recognition errors occur. Furthermore, the information-oriented view allows for easy integration of multi-modal input. To demonstrate the feasibility of our approach,we implemented an interactive map program with spoken and multi-modal input.

2 Using Information in Interpreting Situations

We consider the semantic information that stems from a users' request, taken together with the context and domain modeling, as a *situation*. We claim that a situation in dialogue is non-hostile, since, in general, the dialogue partners are cooperating. Thus, in order to reduce dialogue modeling, we can exploit information we encounter in the present situation to guide the actions of our system.

We place ourselves in the context of task-oriented human-computer interaction in which the user wants the system to perform certain operations or to deliver certain information. The dialogue system is able to perform a given set of operations such as displaying objects, printing out information on objects, and the like. Each operation requires a set of parameters that have to meet lower bounds of information. For example, the operation `display_obj` might require the coordinates of all objects to be uniquely defined. The purpose of the lower bounds on the parameter values are twofold. First, they serve to

determine the minimal amount of information the user must provide in order to execute the intended action. Second, they also define communicative goals that have to be met if the user wants to execute an action. The system has to determine which operation the user wants to perform and sufficiently identify the required parameters.

3 Information-centered Representations

3.1 Domain Modeling

In many knowledge representation formalisms, knowledge is organized in a hierarchical way. To represent the background knowledge, we use a partial ordering of concepts. We call the concepts *types* and the ordering relation $\sqsubseteq$, the *subsumption relation*, according to Carpenter [1992]. Additionally, we describe what features a type consists of by so-called *appropriateness conditions* [Carpenter, 1992]. The terminological knowledge base allows us to express the IS-A relations (in the following noted in cursive letters) and IS-PART-OF relations (noted in capital letters) that hold between objects. We restrict the type hierarchy to be a rooted tree. We can then extend the hierarchy by adding probabilities along the $\mathsf{IS-A}$ links expressing the degree of confidence that an object of type θ is also of type θ' where θ is a direct supertype of θ' (noted $\theta \dot{\sqsubseteq} \theta'$). For the time being, the probabilities are supplied manually, but they could also be determined empirically[1]. We ensure that the probabilities on the $\mathsf{IS-A}$ links leaving a type θ sum up to one,

$$\sum_{\theta \dot{\sqsubseteq} \theta'} P(\theta' \mid \theta) = 1 \quad \forall \text{ types } \theta : \theta \text{ is not a leaf}$$

so that P is indeed a probability distribution. A part of the domain modeling we use in our interactive map implementation is shown in figure 1.

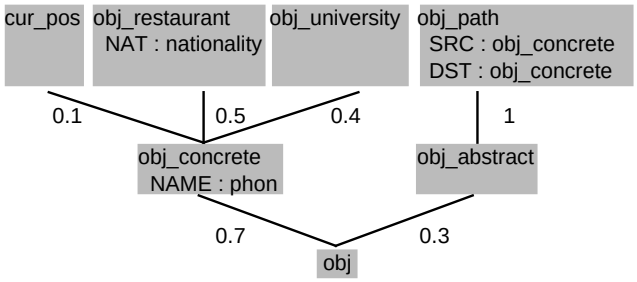

Figure 1: A part of the type hierarchy and its appropriateness conditions used in the map application. The least specific type is at the bottom of the tree.

Note that the domain model does not make any assumptions about the form of the dialogue, i.e. the type hierarchy only represents knowledge about the objects present in the chosen domain, but does not encode any assumptions on how interaction between user and system should take place.

[1] This can be done as follows: Among x objects in the data base, we find x_1 restaurants, thus we set $P(F \text{ is of type } \textit{obj\_restaurant} \mid F \text{ is of type } \textit{obj\_concrete}) = x_1/x$

3.2 The Semantic Representations

In many situations, the information conveyed in the users' request is not specific enough to determine the intended operation and its parameters entirely. The representations of the requests must reflect this fact in order to be able to represent partial information and to leave disjunctions unresolved.

Partially Specified Semantic Representations

We use typed feature structures [Carpenter, 1992] to represent the semantics of the users' requests. These structures easily combine domain-dependent knowledge provided by the type hierarchy and semantics of utterances. Moreover, they can represent partial information. We think of the semantic representations as partial descriptions of operations or objects. Furthermore, typed feature structures can – as can types – be ordered by subsumption. We can thus determine if the information provided is specific enough for the execution of an action.

The natural language input is analyzed by the PHOENIX parser developed by Ward [1994]. Its output, a partial semantic parse tree, is converted to a semantic representation by traversing the parse tree and applying construction rules to the nodes. The semantics of the utterance is given by a set of possibly partially specified feature structures that are stored in a discourse history. Each structure represents the semantics of a phrase of one of the main syntactic categories NP, VP, or PP.

Underspecified Representations

Typed feature structures are partial descriptions of objects that themselves are represented by typed feature structures over the same type hierarchy. This allows us to determine described objects by compatibility check as well for anaphoric reference as for accessing objects from a database.

However, in general, feature structures do not adequately represent unresolved disjunctions, since the generalization of a set of feature structures is in general only an approximation of the set. For this reason, we use *underspecified feature structures* to represent non uniquely referring expressions. Examples of underspecified feature structures are shown in figure 2. We think of an underspecified typed feature structure as a compact representation of a set of (possibly partial) descriptions. Underspecified feature structures are a generalization of the typed feature structure formalism. Our underspecified representations are optimal in the sense that they represent explicitly all information which is common to more than one structure. Since they explicitly factor out common information, they contain more information than the set of disjuncts itself. As described in the next section, this information is used when generating clarification questions.

In the attribute-value-matrix notation that we use to display feature structures, the type marked with an asterisk is the most specific lower bound of the types in its scope. The scope is indicated by curly brackets. The alternatives are represented inside curly brackets. Indices behind types identify the typed feature structure this information belongs to. If there are no indices, the information belongs to all feature structures. Features that are common to only a subset of all represented feature structures are in the scope of the most specific type that is in common to that subset. The feature PHONE in figure 2 (a) is such an example.

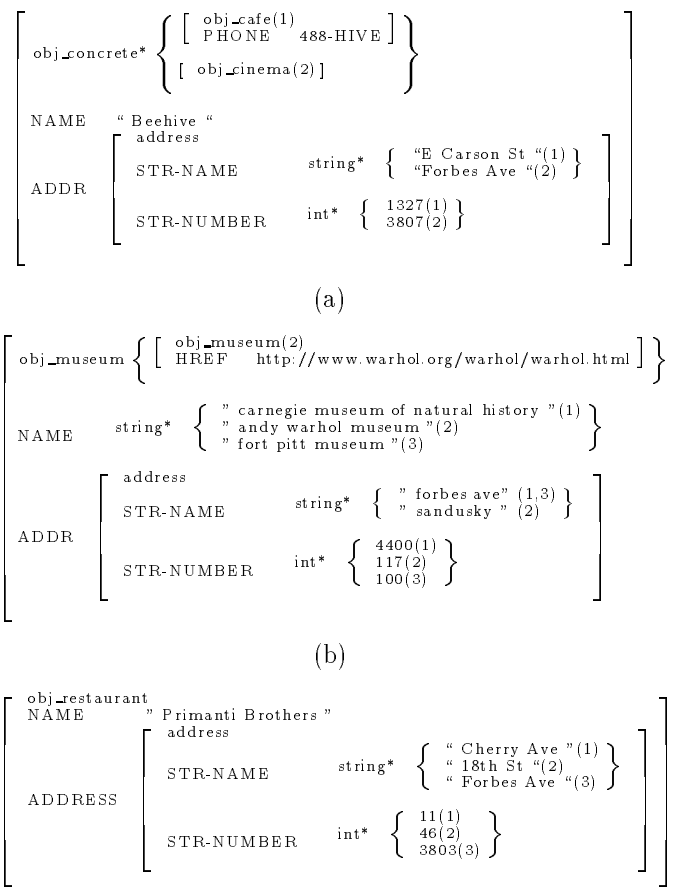

Figure 2: Three underspecified feature structures representing the objects referred to by the NPs "the museum", "the Beehive" and "Primanti Brothers". There are two objects called "Beehive" in our data base, one being a cinema, the other one a cafe. Moreover, we find three different museums and three restaurants called "Primanti Brothers".

The shape of the type hierarchy is of importance for the information represented in underspecified structures. The 'deeper' the type hierarchy, the more knowledge about the relations of the types is encoded. This is analogous to the information content of decision trees. Consequently, the specificity of underspecified feature structures increases with the amount of information in, or the depth of, the type hierarchy.

Moreover, the nodes of underspecified feature structures are arranged as decision trees themselves that are sub-trees of the original type hierarchy (see figure 3 for an example). The probabilities along the IS-A links have to be re-normalized to take missing types into account. In order to adapt the probabilities correctly, the underspecified feature structures must be optimal.

4 Information-centered Dialogue Processing

4.1 Integrating Information

The flexibility of a dialogue system increases with the capability of integrating information from different sources. Possible information sources in our system are domain knowledge, information from different input devices and database access. Integration of complementary information is achieved by type inference and unification.

Type Inference

Type inference ([Carpenter, 1992]) serves to increase a type θ in a feature structure to the least specific type θ', $\theta \sqsubseteq \theta'$ for which the features assigned to θ are appropriate. Since the type hierarchy encodes domain knowledge, type inference integrates domain knowledge into the feature structures. There are several ways in which type inference is useful.

First, type inference can provide useful information in integrating context information. Consider the sentence `Show me the Mexican place`. The noun `place` gets mapped to the type *obj_concrete*. However, in our domain model shown in figure 1, only a restaurant can have a nationality. Type inference can provide this information, as in the following equation, where type inference maps a non well-typed feature structure to a well-typed one.

$$TypInf\left(\begin{bmatrix} obj\_concrete \\ \text{NAT} \quad mexican \end{bmatrix}\right) = \begin{bmatrix} obj\_restaurant \\ \text{NAT} \quad mexican \end{bmatrix}$$

Second, in cases in which the speech recognizer does not recognize the spoken utterance well or the utterance is ill-formed, it is probable that the parser will skip part of the input. In these cases, it generates a semantic representation of the utterance that represents only partially what has been said. Typically, the information in the representation is not sufficient to trigger the intended operation. An example which occurred during our test sessions is the request `How can I get to the Elbow Room` that was recognized as `How can I get to [ zoom out ]`. The semantic parser skips the bracketed part of the input so that the generated feature structure is

$$\begin{bmatrix} speech\_act \\ \text{ACTION} \quad show\_path \end{bmatrix}$$

A total type inference procedure allows us to determine the least specific well-typed feature structure while retaining all present information. In this case, the type inference procedure yields

$$\begin{bmatrix} speech\_act \\ \text{ACTION} \quad show\_path \\ \text{OBJECT} \quad obj\_concrete \end{bmatrix}$$

This representation makes requests for missing values easier (or even unnecessary) to trigger, since all possible features are present *after* the type inference and furthermore, their value is set to be the most general type allowed for in this context. This means that strict domain modeling contributes to more specific representations.

Third, when unifying well-typed feature structures the result is not necessarily well-typed [Carpenter, 1992]. Type inference may yield more specific structures, thus allowing for more specific clarification dialogues. This is

important in cases in which the specificity of a parameter increases during the dialogue, and the associated operation also needs to be more specific.

Integrating Information from Different Input Sources

In systems that allow for discourse context and multimodal input at the same time, ambiguities between the deictic and anaphoric use of pronouns arise. The information supported by a pronoun is that it refers to an object the user expects to be uniquely identified by the context and possibly additional input. If complementary deictic information is present, this information is used to determine the object the user refers to. In case where more than one pronoun of possibly deictic use occur in the phrase, time information is used to correctly join deictic and acoustic information.

In our case, a gesture can be a point given by coordinates, a line given by the coordinates of the two endpoints, or an area given by a set of lines. On the semantic level, this information is represented in feature structures. If complementary semantic information coming from spoken input is available, the gestural and language representations are associated and unified. For example, this makes it possible to disambiguate the representation of an anaphor as shown in figure 3 by mouse click or drawn circles to refer to areas in which all restaurants should be shown.

Database Access

Data retrieval procedures are linked to the types of the roots of feature structures. For the semantic representation of every NP in the discourse an appropriate data retrieval procedure, if provided, is executed. Database retrieval procedures generate an underspecified feature structure on the object level representing all objects that are compatible with the information on the semantic level. Note that the database access also takes geographical information into account, for example when assigning to the semantic representation of `this restaurant + <mouse click>`

$$\begin{bmatrix} \textit{obj\_restaurant} \\ \text{P1} \begin{bmatrix} \textit{point} \\ \text{X} \quad \textit{254} \\ \text{Y} \quad \textit{746} \end{bmatrix} \end{bmatrix}$$

the restaurant that is displayed on the screen is closest to the given point as opposed to some other object such as the intersection that is even closer to the point than the restaurant.

The same retrieval procedures apply when resolving reference of anaphora. This makes it possible to generate representations of ambiguous referring anaphora.

4.2 Strategies for Disambiguation

Due to speech recognition errors or inappropriate input, only parts of the input may be used for interpretation. As an example, consider a database access that yields an underspecified feature structure representing the destination of a path. Another example is the case in which necessary information is not conveyed by the user. In both cases, the representation is not informative enough to execute the users' request.

We investigate two strategies to recover from this state. The first strategy, an unbiased one, is to ask a clarification question, a second, a biased strategy, is to automatically choose the most probable interpretation, thus avoiding the clarification question. Note that the proposed representations do not favor one strategy over another. Contrarily, they allow for determining all possible solutions of the request in a first step. Disambiguating using clarification questions yields an unbiased strategy. On the other hand, a selection according to some (domain specific) criterion implements the biased strategy. This gives a general domain-independent dialogue strategy that is parametrized by domain-specific constraints such as selecting the next place in a map task or always asking clarification questions in high-security environments.

Clarification dialogues

There are at least two ways of asking clarification questions both of which will be discussed in this section. In any case, underspecified representations are used as a basis for formulating the clarification question.

To make it possible for the user to distinguish between the different objects using descriptions, unique descriptions of the objects have to be generated. These are filled in templates of the sort `Do you mean` $<desc_1>, \ldots, <desc_{n-1}>$ `or` $<desc_n>$`?`. The user is then supposed to convey (parts of) the descriptions to disambiguate the structure.

The algorithm works as follows. Given an underspecified feature structure F to be disambiguated, we determine the set of all ambiguous paths in F and choose the path π whose feature values have maximum entropy. For example, in figure 2(b), the maximal entropy path is "[NAME]". We then select one of the types θ, for example "Andy Warhol Museum" and calculate the unification $F \sqcup \pi : \theta$. The type θ is mapped on to a string that will be part of the description. We iterate this procedure until F represents one object uniquely. The clarification question generated by this procedure for the underspecified structures shown in figure 2 are then:

```
Do you mean the cafe or the cinema?

Do you mean the Carnegie Museum of
Natural History, the Andy Warhol
Museum or the Fort Pitt Museum?
```

and

```
Do you mean the one Cherry Ave,
the one on 18th St or the on Forbes ave?
```

respectively. The first question makes use of the different type of the two objects, thus based on a path whose length is zero, the second question asks for the name of the object, since the type is the same (namely *obj_university*), and the third question asks for the address, since type (*obj_restaurant*) and name (`''Primanti Brothers''`) are the same. This example shows again that the information currently present in the situation determines the form of the question.

How then can the disambiguation be achieved? If all readings that are incompatible with the additional information are removed from the underspecified structure, only the structures satisfying the users' constraints remain.

The advantage of this method is that one question is sufficient to disambiguate the underspecified structure.

The drawback of this approach is that the question tend to appear unnatural if the number of disjuncts is large (greater than 4). In these cases, a second strategy is provided. One could ask the user to provide a certain bit of information that helps to reduce the number of possible entities and repeat the process as long as the expression refers uniquely. The caveat here is that the user does not necessarily know the answer to the question and often more than one question must be asked.

The information the user may provide to disambiguate the underspecified representations is determined by the underspecified representations and, since these are the result of preliminary processing, by the context. Moreover, all choices that are compatible with the present information are explicitly determined. Thus, the proposed strategy leaves all possible choices to be decided by the user. For this reason, this strategy is *unbiased*.

Note that a prerequisite for the approach are underspecified representations that factor out differences between the possible options. In underspecified feature structures it is easy to determine where the differences of a set of feature structures are and how different (in terms of entropy) the values are.

It should also be noted that the system does not expect the user to pick one of the proposed answers. Instead, the system is able to deal with any information that serves to disambiguate (not necessarily entirely) the underspecified representations[2].

Avoiding Clarification Dialogues

There are cases in which the unbiased strategy described above leads to tedious and lengthy dialogues. In these cases, a biased strategy is used. Consider for example the discourse`''Show me the path from here to Carnegie Mellon University''` and `''Can I have more information on that''`. A (simplified) representation of the information conveyed by the anaphora is given by the underspecified feature structure

$$\left[\begin{array}{l} \text{obj}^* \left\{ \begin{array}{l} \left[\text{obj\_concrete}^* \left\{ \begin{array}{l} \left[\begin{array}{l} obj\_university \\ \text{NAME } \text{“}CMU\text{“} \end{array}\right] \\ cur\_pos \end{array}\right\}\right] \\ \left[\begin{array}{ll} \text{obj\_path(3)} & \\ \text{DST} & \left[\begin{array}{ll} \text{obj\_university} & \\ \text{NAME} & \text{“CMU”} \end{array}\right] \\ \text{SRC} & \text{cur\_pos} \end{array}\right] \end{array}\right\} \end{array}\right]$$

The underspecified representation yields the following information: first, the most specific type ρ^* that subsumes the type of all objects (*obj* in the above example), and second, the n types ρ^k whose most specific lower bound is ρ^* of the objects (*obj_path* and *obj_concrete* in the above example). This applies recursively until the final types (those that are not marked with an asterisk) are reached. The paths from the type of the root to the final types yield a decision tree that is a subtree of the type hierarchy. Re-normalizing the probability distributions for the sub-hierarchy yields probability distributions again. The decision tree for the above example is shown in figure 3.

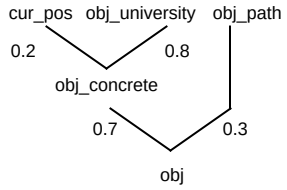

Figure 3: The decision tree extracted from the type hierarchy after re-normalization of the probability distributions

The representations allow us to calculate the probabilities $P\left(\rho^k \mid \rho^*\right)$ just by going from ρ^* to ρ^k in the sub-hierarchy while multiplying the conditional probabilities according to

$$P\left(\rho^k \mid \rho^*\right) = \prod_{i:\rho^* \sqsubseteq \rho^i \dot{\sqsubseteq} \rho^{i+1} \sqsubseteq \rho^k} P\left(\rho^{i+1} | \rho^i\right) \qquad (1)$$

The resulting distribution is indeed a probability distribution, since

$$\sum_{k=1}^{n} P\left(\rho^k \mid \rho^*\right) = 1$$

The result of this process is a probability distribution assigning probability to each object the anaphor may refer to as a function of the context, the domain modeling and the *a priori* probabilities. We use a heuristic to disambiguate the anaphora: if the difference between the largest probability and the second largest probability exceeds a certain threshold, there is a strong preference to one reading, thus this reading is assumed to hold. Otherwise a clarification dialogue is generated in order to ask for additional information.

This strategy is biased in that it disambiguates an underspecified representation such that it refers to the preferred reading. In other words, not all options are left to the user, since the system decides itself what the user meant. The intention behind this heuristics is that it is cheaper in average to repair an incorrect reading a few times than ask every time a question.

A prerequisite for correct probabilistic interpretation in eq. 1 is the optimality of underspecified representations w.r.t. representing all common used information. Let us assume for the sake of example that the underspecified structure does not represent the fact that both the current position and the university are subsumed by *obj_concrete*. This yields a structure that represents adequately the circumstances but is suboptimal in that the fact that both *obj_university* and *cur_pos* are of type *obj_concrete* is not represented.

$$\left[\begin{array}{l} \text{obj}^* \left\{ \begin{array}{l} \left[\begin{array}{ll} \text{obj\_university(1)} & \\ \text{NAME} & \text{CMU} \end{array}\right] \\ [\ \text{cur\_pos}] \\ \left[\begin{array}{ll} \text{obj\_path(3)} & \\ \text{DST} & \left[\begin{array}{ll} \text{obj\_university} & \\ \text{NAME} & \text{“CMU”} \end{array}\right] \\ \text{SRC} & \text{cur\_pos} \end{array}\right] \end{array}\right\} \end{array}\right]$$

[2]In the current implementation, the language model predicting the answer is generated on the fly which means that the speech recognizer assumes the answer to be one of the proposed options. However, this is a limitation imposed by the way we control the recognizer in the current implementation, not a limitation of the dialogue processing algorithms.

The decision tree represented by this underspecified structure is not optimal either.

5 An Example Application

To verify empirically the validity of our approach, we implemented an interactive map dialogue program whose purpose is to provide a test bed for interactively accessing information about places on a map and to generate path descriptions. As one of the input sources, we use the JANUS continuous speech recognizer [Waibel, 1996] using a 1600 word vocabulary.

The dialogue system is implemented as a multi-blackboard system. The currently used blackboards are the map database providing street and place information, the discourse blackboard storing the semantic representations of formerly uttered requests and dialogue answers and the multi-modal blackboard representing objects that can be referred to by gesture. The data base stores information of about 3000 streets (300 of which have been selected to be in the vocabulary of the speech recognizer) and of about 350 different locations. Knowledge sources may contribute complementary information. Each knowledge source implements a set of typed predicates that are used to form expert system style rules. The rules contain typed variables that are instantiated with typed feature structures stored in the discourse blackboard, such as word, parse tree, semantic representations or objects in the discourse module provided the type of their root is as least as specific as the type of the variable. The set of rules mediates the communication between different modules and the integration of information. The users input is added to the discourse blackboard. The behavior of the system, i.e. the interaction of the knowledge sources, the blackboard and the user is entirely determined by the information available in the blackboards and the rules together with the side effects of the predicates as implemented by the knowledge sources. This results in a stepwise refinement of the available representations.

This design allows for easy adaptation to a new domain since adaptation does not require new dialogue modeling. No assumptions on the domain have been hard-coded. New knowledge sources may easily be added so that new predicates may be made use of in the constraints, thus implementing a possibly completely different behavior of the system.

6 Evaluation

The implemented dialogue system can perform ten different actions that include panning or zooming the map, calculating paths, their lengths and travel times as well as scheduling hotel and restaurant reservations. Altogether, 223 request were presented to the system. In 53 cases, the system generated a representation of an incorrect operation. These errors were mostly caused by recognition errors. The length of the dialogues ranged from 0 system-initiated turns (for fully specified requests) to 5 system-initiated turns for hotel reservation.

In 20 cases, at least one feature was missing in the final representation. Most often, this was due to incorrectly interpreted prepositional phrases. In 17 cases, features were assigned incorrect types which were mostly wrong place names or street names which again was due to misrecognitions.

Type inference proved to be a useful feature since, due to recognition errors, the semantic parser often skipped parts of the input. In these cases, it was oftentimes possible to restore at least partially the original information as conveyed by the users request.

7 Summary

In this paper, we developed an information based approach to dialogue systems. The central idea is that actions are performed according to the specificity of the representations, not as a function of an explicit state the system is in. In particular, the system does not make any assumptions on what information in which order at what time using which input device the user should provide.

The chosen representations are particularly useful for representing spontaneous speech since spontaneous speech consists for the most part of utterance fragments for which no closed formula is derivable. Moreover, speech fragments may increase specificity not only by adding fillers but also by generating more specific types. The disambiguation of underspecified feature structures using fragmentary and elliptical input shows some similarities to the *Micro Conversational Events* described in [Poesio and Traum, 1997] since the unification of the semantic representations of answers incrementally increases the information in underspecified representations.

We showed that, in task-oriented domains, context information and domain model are informative enough to determine what the system should do. Since our approach is information-centered, multi-modal input can easily be implemented if all input information is represented using the same formalism.

We investigated two strategies to obtain complementary information of the user. The unbiased strategy leaves all action to the user at the expense of possibly many questions to answer, whereas the biased strategy restricts possible actions. Biased strategies are of particular interest if unintended actions can be avoided or repaired so that the mean effort is minimized.

Acknowledgements

I would like to thank Alex Waibel, Wayne Ward and Bernhard Suhm for discussions, advice and suggestions concerning the topics discussed in the paper. Also, I would like to thank the three anonymous reviewers for their helpful comments.

References

[Carpenter, 1992] Bob Carpenter. *The Logic of Typed Feature Structures.* Cambridge University Press, 1992.

[Poesio and Traum, 1997] Massimo Poesio and David R. Traum. Conversational Actions and Discourese Structure. *Computational Intelligence*, 13(3), 1997.

[Waibel, 1996] Alex Waibel. Interactive Translation of Conversational Speech. *Computer*, 29(7), July 1996.

[Ward, 1994] Wayne H. Ward. Extracting Information in Spontaneous Speech. *Proceedings of the International Conference on Speech and Language Processing*, 1994, Yokohama, Japan.