INFORMATIONAL CHARACTERIZATION OF DIALOGUE STATES

Matthias Denecke

Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
denecke@cs.cmu.edu

ABSTRACT

We introduce multidimensional feature structures as a generalization of standard slot/filler representations commonly employed in spoken language dialogue systems. Nodes in multidimensional feature structures contain an n dimensional vector of values instead of one single filler element. The additional elements serve to represent, among other information, confidence measures of speech recognizers or the number of times a filler has been queried. We demonstrate the application of multidimensional feature structures to spoken dialogue systems. We show that unification based dialogue processing can be retained as long as the elements of the fillers are drawn from partially ordered sets. The dialogue manager employs a variant of constraint logic programming for representing dialogue strategies and update rules. The constraint logic program partitions the space of possible dialogue states in sets of states that are equivalent for the dialogue strategy.

1. INTRODUCTION

The notion of dialogue states plays an important role in dialogue systems. The functionality of a dialogue state is twofold. First, it gives an appreciation of the amount of information acquired in the dialogue thus far, and second, it prescribes the action the system should take in that state. In finite state automata based dialogue systems, states are represented explicitly. The amount of acquired information in the dialogue up to a given time is equated with the states the dialogue manager is in. The accepting states of the automaton are assigned the tasks the user intends the system to execute. Explicit confirmation questions are represented by circular state transitions, while grounding is represented by forward edges in the automaton.

While the assumption that the amount of acquired information equals a state simplifies the implementation of dialogue managers, the restriction the FSA imposes on the dialogue structure is not negligeable. For this reason, other approaches favor an implicit representation of dialogue states. In [4] the dialogue state is implicitly described by a vector of slots that can be filled incrementally. In [3, 5] the dialogue states is described implicitly by typed feature structures and partially filled forms.

I would like to thank my colleagues at the Interactive Systems Laboratories for helpful discussions and encouragement. This research is supported by the Defense Advanced Research Projects Agency under contract number DAAD17-99-C-0061. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the DARPA or any other party.

In this paper, we introduce *multidimensional feature structures*, a generalization of typed feature structures that store in addition to semantic information information on the quality on the input, the number of times a value has been queried, and so on. Furthermore, we develop an approach to characterize the informational state of the representations in the discourse by an attribute vector which we equate with the dialogue state. The attribute vector "summarizes" the information in the multidimensional feature structures in a more abstract form. It is determined by a constraint logic program that traverses the representations recursively. The difference between the attribute vector and the representations presented in the work above is that the attribute vector abstracts entirely over the application domain. The information in the attribute vector is specific enough to answer questions like *"Is there enough information available in the discourse to determine the users' intent uniquely?"* or *"Were there sufficiently many conversational breakdowns to warrant a transfer to a human operator?"* without relying on domain specific information. This approach has the following advantages. First, the fact that standard unification generalizes to multidimensional structures allows each component to contribute different aspects to the representations. Second, the characterization of the dialogue states yields a set of attributes that serves as a basis for the dialogue manager to decide on an appropriate strategy. And third, the characterization of the dialogue states abstracts away peculiarities of the domain under consideration.

2. MULTIDIMENSIONAL REPRESENTATIONS

Frame-based [5, 6] or feature structure based [3] representations in dialogue managers are appealing because of the partiality of information (see [1] for a theoretical analysis). We generalized typed feature structures to representations where not only partial semantic information, but also information on the quality of the input or confidence measures among others are represented. More specifically, we employ *multidimensional feature structures* where type information in the nodes is replaced with a vector of elements each of which is drawn from a partial order. The motivation behind multidimensional representations is to represent multiple facets of the input while being able to retain unification based processing.

2.1. Contents

The following information is currently represented in the multidimensional representations.

Counting of Prompts. With each node is associated an integer indicating the number of times the value of the node has been

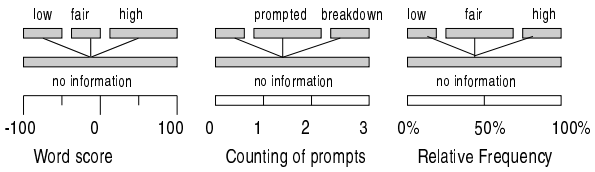

Figure 1: Different partial orders.

actively queried by the dialogue manager. High counts indicate some sort of conversational breakdown.

Confidence Score. Confidence score or word score information is separated in intervals which form a partial order.

Relative Frequency. The relative frequency is used to represent the number of the occurrences of a given word in the hypotheses lists, normalized by the length of the list.

Parse Quality. The percentage of words that are covered by the skipping parser.

Consistency with the domain model. The ontology represents IS-A and HAS-A relations between concepts.The parse tree is converted into feature structures by conversion rules. When partial structures of the converted parse trees provide inconsistent type information, the degree of the inconsistency is stored in the node. The degree of inconsistency is the lengths of the paths from the incompatible types to their greatest lower bound in the ontology.

Figure 1 shows how the different aspects are cast into partial orders. Each partial order contains the elements used in one particular dimension of the multidimensional feature structures.

2.2. Operations

The unification and subsumption operations on typed feature structures require only the node information to be partially ordered. Since the node information in multidimensional feature structures is drawn from a set $\mathbf{V} = P_1 \times \ldots \times P_n$ where each of the P_i is endowed with a partial order $\sqsubseteq_i$, we can define a partial order on V, where $\mathbf{v} \sqsubseteq \mathbf{w} :\Leftrightarrow v_i \sqsubseteq_i w_i \ \ \forall \mathbf{p}, \mathbf{q} \in \mathbf{P}$. Consequently, unification and subsumption generalize in a straightforward manner to multidimensional feature structures.

2.3. Object-Oriented Approach

The fact that the attributes are represented together with the semantic representations rather than in detached state representations in the dialogue manager enables an object-oriented combination of information in the discourse. From an object-oriented point of view, forming the multidimensional representations can be considered as an instance of multiple inheritance, where the finalized representations inherit the domain specific semantics and the domain independent attributes. This, together with the fact that the standard unification procedure carries over to multidimensional representations, also implies that each processing module may contribute to any aspect of the representation, as the contributing information is simply unified with the present representation.

3. COMBINING INFORMATION SOURCES

The knowledge sources provide the interface of the dialogue manager with the back-end application.

3.1. Input from the Speech Recognizer

The input is provided by a speech recognizer through the MS Speech API. The API provides a list of hypotheses each of which is parsed and converted to typed feature structure representing the semantics of the request. Only then is the relative frequency of the nodes determined. This has the advantage that slight variations that do not influence the semantic representations (such as `'a'` vs. `'the'` or `'flight'` vs. `'flights'`) are ignored and only variations in the semantic representations are considered.

Consistency with the domain model is not of much use with grammar based recognizer we are currently using. This is true because all possible utterances from the recognizer are entirely determined by the grammar, and consistency can be checked off-line. However, once a language model based recognizer is employed, this feature could be used to determine the quality of the current utterance.

As an example, the hypothesis list for the utterance *" a flight to Pittsburgh on March first"*

a	flights	to	Pittsburgh	on	March	first
the	flights	to	Pittsburgh	on	the	first
the	flight	to	Pittsburgh	on	the	fifth
the	flights	to	Pittsburgh	on	the	fifth

will cause the following two feature structures to be generated (shown is only the relative frequency for clarity):

$$\begin{bmatrix} \textit{obj\_flight,high} \\ \text{DST} \quad \textit{"Pittsburgh",high} \\ \text{DATE} \begin{bmatrix} \textit{obj\_date,high} \\ \text{MONTH} \quad \textit{march,high} \\ \text{DAY} \quad \textit{1st,fair} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textit{obj\_flight,high} \\ \text{DST} \quad \textit{"Pittsburgh",high} \\ \text{DATE} \begin{bmatrix} \textit{obj\_date,high} \\ \text{MONTH} \quad \textit{march,high} \\ \text{DAY} \quad \textit{5th,fair} \end{bmatrix} \end{bmatrix}$$

3.2. Dialogue Goals

A dialogue goal can be seen as the description of a form that is filled out through the spoken dialogue with the system [3, 5]. The goal description consists of a typed feature structure whose types are drawn from the domain model. It serves as an informational lower bound describing the amount of information to be acquired before the action associated with this goal can be executed. Note that the dialogue goal specification does not make any assumptions as to how this information is acquired, nor as to how the acquired information is to be processed. Thus, the dialogue goals form the specification of a task model that is orthogonal to any dialogue strategy specification and independent from the implementation of the back-end system.

As long as a user is engaged with the system in a dialogue, it is then the task of the dialogue system

1. to determine if the user intends to have the system perform one of the tasks known to the system, and if so,
2. to interactively acquire all the information that is needed for the system to uniquely determine the task to be executed and all its parameters, and

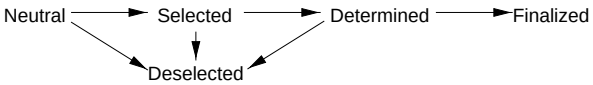

Figure 2: State transitions of the dialogue goals

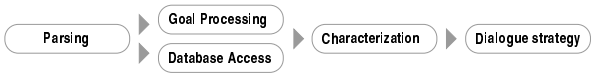

Figure 3: Processing Chain

3. finally to notify and pass control to the subsystem responsible for the task execution once this state has been reached.

For that purpose, each task description has an internal state that can take one of the following values: NEUTRAL, SELECTED, DESELECTED, DETERMINED and FINALIZED. The state transitions are shown in figure 2. Each state transition is passed on to the implementation of the dialogue package in the back-end application which may or may not choose to make use of this information.

The state of a goal incompatible with the current representation becomes DESELECTED. A goal in the state SELECTED becomes DETERMINED as soon as it is the only goal in the SELECTED state. A DETERMINED goal becomes FINALIZED as soon as the information acquired in the dialogue is at least as specific as it is required by the goal.

To continue the above example, as both feature structures are compatible to the dialogue goals *inquire_price* and *book_flight*, the state of these goals transits to *selected* while all other dialogue goals move to a deselected state.

3.3. Databases

Referring expressions are resolved by accessing the databases associated with the dialogue manager. A successful database access returns an underspecified feature structure [3]. The underspecified structures are used to determine the content of disambiguation questions.

For each of the representations created by the parser, a database request is issued. Should the result set for one of the requests be empty, the multidimensional feature structures represent this fact. Otherwise, a feature structure rerpesenting a description of the retrieved objects is generated.

Assuming that in the above example the origin of the flight becomes known or can be inferred, a database request is triggered for the flights to Pittsburgh on March 1st and on March 5th.

4. INFORMATIONAL CHARACTERIZATION

The introduction of the multidimensional feature structures leads to an increasing size of the state space. This is aggravated by the fact that for each input from the speech recognizer a set of representations is generated. At the same time, the information provided by a multidimensional feature structure is too fine grained to determine the dialogue strategy directly. For example, in order for the dialogue manager to determine if an explicit confirmation question needs to be generated, it is necessary to determine if a filler has low confidence score, but not which filler it is.

For this reason, we use a constraint logic program to determine a five dimensional dialogue state vector $\mathbf{s} = \langle s_1, \ldots, s_5 \rangle$. The logic program traverses the representations in the discourse and the representations in the goal manager recursively and returns the characterization of the current dialogue situation. The motivation behind this approach is to arrive at a compact, domain independent representation based on which the dialogue manager can derive its decision for the dialogue strategy. As such, the functionality of the constraint logic program is to partition the high dimensional space of all possible dialogue states into sets of states whose elements are equivalent w.r.t. the dialogue strategy to be pursued.

The five elements of the dialogue state vector describe the following abstract attributes of the ongoing dialogue: (1) quality of the current input, (2) quality of the overall dialogue, (3) speech act of the current input, (4) intention and (5) manner of reference.

4.1. Dialogue State Vectors

The goal of the characterization is to abstract away the form of the particular representation of the utterances and to arrive at a more abstract classification of the ongoing dialogue. The dialogue strategy is then determined based on the values of the features rather than on the representations directly (see figure 3). The characterization of the dialogue state takes place after the semantic content of the utterance has been determined and before its semantic representation is inserted in the discourse. The dialogue manager decides its strategy based on the information in the dialogue state vector.

It should be noted that the attributes do not contain information relevant to a specific domain, but can be applied to any domain.

In the following, we describe the attributes that partition the state space.

4.1.1. Overall Quality

The motivation behind this feature is for the dialogue manager to be aware of the quality of the ongoing dialogue. Depending on the value of this feature, the dialogue manager can switch to a strategy providing explicit prompts or, in case of entire breakdown, transfer to a human operator. The value of this attribute is one of *good, fair* and *bad*. The value is determined in a fashion similar to the quality of the current input, except that the all accumulated representations are traversed, not only the current representations.

4.1.2. Quality of current Input

The motivation for this attribute is for the dialogue manager to determine if and how the current utterance should be entered into the common ground. As the dialogue manager needs to decide if parts of the utterance need to be integrated in the discourse and if parts of the utterance need to be confirmed explicitly, the three admissible values for the certainty feature are *certain, partly uncertain* and *entirely uncertain*. The certainty of the representation is determined by three factors, namely the confidence measure or acoustic scores of the speech recognizer, the consistency of the semantic representation with the domain model and the parse tree coverage.

4.1.3. Type of Speech Act

The type of the speech act determines the structure of the dialogue and the way the dialogue history is updated. In some cases, the speech act type could be already determined during parsing and

State vector	Systems' reaction
$\langle bad, \_, \_, \_ \rangle$	Transfer to human operator
$\langle fair, bad, \_, \_ \rangle$	Ask to repeat utterance
$\langle good, fair, \_, \_ \rangle$	Confirm parts of the utterance
$\langle good, good, \_, ambiguous \rangle$	Ask disambiguation question

Figure 4: Examples of relationship between states and a possible dialogue strategy

no additional processing is necessary. More often, however, the speech act type is determined by the context.

The speech act type is determined by a set of clauses that check for compatibility with the dialogue goals and the representations in the discourse. Simply put, if the representation of the current utterance is compatible with at least one of the dialogue goal representations that is not deselected, and the speech act before the current utterance was a question by the dialogue manager, the current speech act is considered an answer. Note that this algorithm allows the user to "over-answer" questions, that is, to provide more information than the dialogue manager has asked for.

4.1.4. Intention

This attribute characterizes the amount of accumulated information relative to the activated dialogue goals at this time. The attribute determines if the dialogue manager could determine the intention of the user, and, if so, the available information is specific enough to execute the action. The value of the attributes is one of *neutral, selected, determined, finalized* and *inconsistent*. The first four values correspond to those in the dialogue state transition diagram (see figure 2) while the value *inconsistent* indicates that no enabled dialogue goal is consistent with the information available in the discourse.

4.1.5. Manner of Reference

This attribute characterizes the way referring expressions refer to objects in the database. The possible values for this attribute are *none, close, unique* and *ambiguous*. The first value indicates that no object fulfilling the constraints given by the user could be found. The second value *close* describes the state where only closely matching objects could be retrieved that do not entirely satisfy the request of the user. The third value describes the state where matching objects could be retrieved and the number of retrieved objects fulfils the numeric constraint of the determined dialogue goal. The last value describes the state where the number of retrieved objects exceeds the number of required objects in the determined dialogue goal.

4.2. Link to Dialogue Strategy

The values of the dialogue state vector do not contain domain specific information. The dialogue manager decides its next action based solely on the information in the dialogue state vector. For example, in the case of good input quality, good overall quality and ambiguous reference, the dialogue manager decides to generate a disambiguation question. As another example, parts of the relationship between the values of the quality attribute and a possible dialogue strategy is shown in figure 4.

5. CONCLUSION

The advantages of the rules in combination with the multidimensional feature structures are twofold. First, they allow it to combine dialogue states that are equivalent with respect to some particular feature such as "conversational breakdown occurred". In addition to the dialogue state given above, any state in which a slot has two incompatible fillers with high confidence could be perceived as a conversational breakdown, causing the dialogue manager to behave in the same way as above. The rules allow to determine the equivalence of both dialogue states regarding the feature "conversational breakdown". Thus, the informational characterization of dialogue states introduces a layer of abstraction hiding details of the dialogue irrelevant for determining the appropriate dialogue strategy. Second, since in multidimensional feature structures information relating to the semantics of the utterance is represented independently from, say, confidence information, dialogue strategies relying on confidence information may be specified independently from the domain at hand. In the above example, both conditions for detecting conversational breakdowns are virtually independent from the underlying domain. Since particularly the strategies for implicit and explicit confirmation may become complex, reuse of parts of the dialogue strategy specifications is of interest. We show how the independent parts of the dialogue strategy can be reused in other application domains, since the orthogonal specifications can simply be combined.

The work presented here contributes to our goal of building a decomposable, modularized dialogue manager. In [3] we demonstrate how the semantic content of clarification questions in the presence of dialogue goals can be selected while relying only on an ontology as domain specific knowledge source. In [2] we show how techniques borrowed from object-oriented programming enable reusable grammar and ontology modules. Here, we describe how a dialogue manager can arrive at a domain independent characterization of the dialogue state. Remaining work includes translating the characterization of the dialogue state into a reusable dialogue strategy.

6. REFERENCES

[1] A. Abella and A.L. Gorin. Construct Algebra: Analytical Dialog Management. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[2] M. Denecke. Modularity in Grammar and Ontology Specification. In *Proceedings of the MSC 2000 Workshop, Kyoto*, 2000. Available at http://www.is.cs.cmu.edu.

[3] M. Denecke and A.H. Waibel. Dialogue Strategies Guiding Users to their Communicative Goals. In *Proceedings of Eurospeech, Rhodos, Greece*, 1997. Available at http://www.is.cs.cmu.edu.

[4] A. Ferrieux and M.D.Sadek. An Efficient Data-Driven Model for Cooperative Spoken Dialogue. In *Proceedings of the International Conference on Spoken Language Processing, Yokohama, Japan*, pages 979 – 982, 1994.

[5] K.A. Papineni, S. Roukos, and R.T. Ward. Free-Flow Dialogue Management Using Forms. In *Proceedings of EUROSPEECH 99, Budapest, Ungarn*, 1999.

[6] W. Ward. Extracting Information in Spontaneous Speech. In *Proceedings of the International Conference on Spoken Language Processing, Yokohama, Japan*, pages 83–87, 1994.