Partial Information in Multimodal Dialogue

Matthias Denecke and Jie Yang*

Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
{denecke,yang+}@cs.cmu.edu

Abstract. Much research has been directed towards developing multimodal interfaces in the past twenty years. Many current multimodal systems, however, can only handle multimodal inputs at the sentence level. The move towards multimodal dialogue significantly increases the complexity of the system as the representations of the input now range over time and input modality. We developed a framework to address this problem consisting of three parts. First, we propose to use *multidimensional feature structures*, a straightforward extension of typed feature structures, as a uniform representational formalism in which the semantic content stemming from all input modalities can be expressed. Second, we extend the feature structure formalism by an *object-oriented framework* that allows the back-end application to keep track of the state of the representation under discussion. And third, we propose an *informational characterization of dialogue states* through a constraint logic program whose constraint system consists of the multidimensional feature structures. The multimodal dialogue manager uses the characterization of dialogue states to decide on an appropriate strategy.

1 Introduction

In the past, research on multimodal input processing systems has focused on how complementary information in different modalities can be combined to arrive at a more informative representation [4, 16]). For example, representations of deictic anaphora are combined with representations of the appropriate gestures. The fusion algorithms employed use symbolic [10], statistical and neuronal [15] techniques to achieve a reduction of error rate [3]. However, the results presented so far are limited to systems that process one sentence in isolation.

In another strain of research, multimodal processing systems have been applied to interactive error correction of speech recognizer hypothesis, where interactions extend over several turns during which a user may select one of multiple

* We are indebted to Laura Mayfield Tomokiyo for helpful comments on an earlier draft of this paper. Furthermore, we would like to thank our colleagues at the Interactive Systems Laboratories for helpful discussions and support. 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 authors and do not necessarily reflect the views of the DARPA or any other party.

input modalities for each turn [14]. No concurrent multimodal input takes place. In contrast to the multimodal systems described above, no deep understanding of the input is necessary since the interactions between the user and the computer take place on the surface level of the spoken words.

At the same time, researchers in the area of spoken dialogue systems address the question of how human-computer interaction through spoken language can be extended to natural dialogues with a machine. Dialogue managers hold a partial representation of the task to be performed and integrate complementary information over time. How this is done is prescribed in a *dialogue strategy*. In order to increase flexibility, many dialogue managers allow for scripted dialogue strategies that can easily be updated ([11, 12]).

In this paper, we propose a framework for multimodal dialogue systems that ties together different aspects of the three approaches described above. In the following, we refer to a *multimodal dialogue systems* as a system fulfilling the following two conditions. First, it should allow a user to perform one of a set of predefined tasks, and second, it should allow the user to communicate his or her intentions over a sequence of turns through possibly different communication channels.

The implementation of a multimodal dialogue system faces a set of challenges. Since the system is supposed to execute actions during and at the end of the interaction with the user, deep understanding of the input is necessary. The need for interactions that potentially range over a number of turns and modalities is not addressed in the work cited above, as interactions with the cited systems can range either over different turns or different input modalities but not both.

Furthermore, the input from different modalities may directly affect the information represented in the dialogue, which, in turn, may affect any sort of display presenting this type of information. Since the form of presentation may vary from object to object, we advocate an object-oriented methodology.

As the input channels are not entirely reliable, robust dialogue strategies are required. Since the switch of modalities has been shown to be effective, it is of interest to investigate dialogue strategies which actively suggest the switch of input channels when communication breakdown occurs. In order to implement this strategy, the system needs to keep track of the input channel associated with each piece of information in the discourse. Furthermore, the system needs to be capable to reason about these channels and their reliability.

The framework we develop in this paper to address these problems consists of three parts. We propose *multidimensional feature structures* as a representational vehicle to capture different aspects of the input provided by different channels. Multidimensional feature structures are a straightforward generalization of typed feature structures. Multidimensional feature structures allow to introduce as many different partial orders as is necessary for the task at hand. Moreover, we propose an object-oriented extension to the formalism of typed feature structures that allows the back-end application to be notified of changes in the representations in an object-oriented manner. Finally, in order to handle the growth of the state space, we propose to characterize the multidimensional representation through a constraint-logic program [8] where the constraints are formed by the multidimensional feature structures and where the logic program makes assertions as to which state the system is in.

2 Multidimensional Feature Structures

In many multimodal and spoken dialogue systems, variants of slot/filler representations are employed to represent the partial information provided by the different input sources. Abella and Gorin [1] provide an algebraic framework for partial representations. In multimodal systems, the multimodal integration then uses some scheme of combination to arrive at an augmented representation that integrates information either across modalities or across sentences.

Typed feature structures [2] have been proposed in the past as a representational vehicle to integrate information across modalities [10] and across time [6]. Typed feature structures can be considered as acyclic graphs whose nodes are annotated with type information and whose arcs are labeled with features.

In order to be able to represent different aspects of multimodal input, we propose to enrich the structure from which the type information is drawn. More specifically, we propose to annotate the nodes of feature structures with n-dimensional vectors $\mathbf{v} \in \mathbf{V}$ rather than with types. The vectors are drawn from the cross product of n possibly distinct sets $\mathbf{V} = P_1 \times \ldots \times P_n$, where each of the P_i is endowed with a partial order $\sqsubseteq_i$. This allows us to represent multimodal aspects of information in the atoms of the representations. Figure 1 shows an example of a multidimensional feature structure.

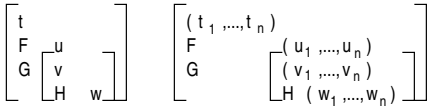

Fig. 1. A typed feature structure and a multidimensional feature structure. Symbols in capital letters denote features, symbols in small letters denote types and indexed symbols denote elements drawn from a partial order.

Note that both the definitions of unification and subsumption of typed feature structures [2] require only that the information associated with the nodes be drawn from a finite meet semilattice. This is a partial order in which all subsets of the elements have a unique greatest lower bound, and if for any subset of elements an upper bound exists, the least upper bound is equally unique. In multidimensional feature structures, the partial orders $\sqsubseteq_i$ of the elements v_i impose a natural partial order on the vectors in V according to

$$\mathbf{v} \sqsubseteq \mathbf{w} :\Leftrightarrow v_i \sqsubseteq_i w_i \qquad \forall 1 \leq i \leq n$$

Thus, standard unification and subsumption generalize in a straightforward manner to multidimensional feature structures.

The Ontology of Classes. The ontology of classes figures most importantly among the knowledge sources. It is used to represent inheritance relations between descriptions of objects, actions, states and properties of objects, actions and states in the domain at hand. The class hierarchy is the equivalent to the type hierarchy for typed feature structures, extended by a simple methodology

to attach methods to the types (see section 3). We also use the class hierarchy to express linguistic information such as speech act types and the like. Figure 2 details two extracts of a domain model.

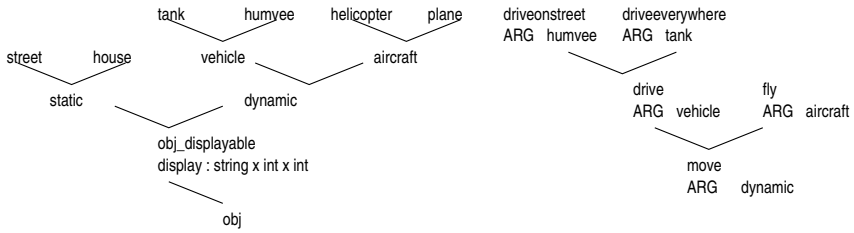

Fig. 2. An extract of an object-oriented domain model for a military application.

Spatial Partial Orders. A crucial feature in multimodal systems is the integration of pointing and moving gestures such as drawing arrows, circles and points. The informational content of a gesture is twofold. First, it communicates a certain semantic content. For example, in some contexts, the gesture of an arrow can be interpreted as a movement while the gesture of a circle can be interpreted as a state. Thus, the informational content conveyed by a gesture can partly be expressed through the concepts introduced in the class hierarchy.

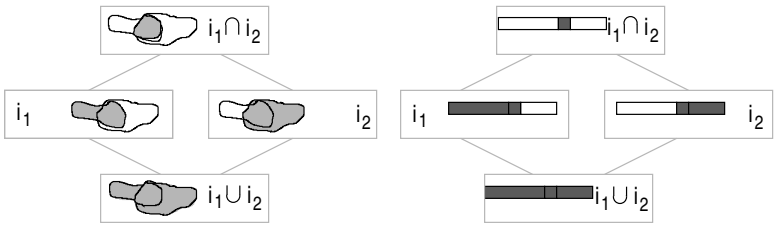

Fig. 3. Partial orders consisting of regions and temporal intervals.

In addition to the taxonomical information, a gesture conveys spatial information as well. This information is represented by sets of two- or three-dimensional points. As the power set of $\mathbb{R}^n$ forms a lattice under union and intersection, spatial information can be represented in multidimensional feature structures as well. An example for a spatial partial order is shown in figure 3.

Interestingly, it is the combination of semantic and spatial information that provides a substantial gain in the representations of multimodal inputs. For example, the system can infer from the domain model shown in figure 2 that streets and houses cannot be the argument of a movement action. If there are movable and unmovable objects in spatial proximity to a movement gesture, the system can infer based on the domain knowledge which objects to move.

Temporal Partial Orders. Temporal information can be used in two separate ways. First, temporal annotations of the hypotheses from different recognizers constrain the multimodal integration ([4, 16]). Second, temporal expressions in the utterance can equally be used be exploited to constrain the database access and to coordinate actions in the domain ([7]). In both cases, temporal information can be represented in intervals.

Figure 4 shows how temporal information constrains the integration across modalities. As the representations to be integrated might not have been created exactly at the same time, the intervals are expanded by a certain factor at the time of creation. This ensures the integration process to be monotonic.

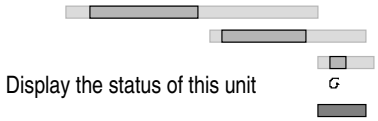

Fig. 4. Overlapping intervals of representations in different modalities. The dark intervals show the creation time of the information while the light intervals show the time during which combination with other modalities is acceptable.

Query Counting. In order to adequately handle communicative breakdowns, the dialogue manager needs to keep track of the number of times a value for a feature has been queried. If this value surpassed certain threshold, alternative strategies can be pursued.

Record of Modalities. In addition to the number of times a value has been queried, we record the modality through which the information has been conveyed. These two information sources interact nicely as they convey information as to how certain an input channel is regarding this particular information.

In order to keep the reasoning mechanism monotonic, we do not represent the fact that a given input modality has not been used yet, as this information may be required to be retracted later in the dialogue. Figure 5 (a) presents the partial order used for the input modalities.

Confidence levels In a similar manner, confidence level annotations can also be represented in multidimensional feature structures. Their partial order is shown in figure 5 (b).

3 Object-Oriented Descriptions and Changes

Multimodal dialogue systems need to address the fact that visualizations of objects may be subject to change at any time in the dialogue, caused either by external events or by user commands. As has been shown in many graphical user interfaces in the last decade or so, object-oriented frameworks greatly reduce complexity of this task. In multimodal dialogue processing, additional complexity is due to the fact that there is not a one-to-one relationship between descriptions

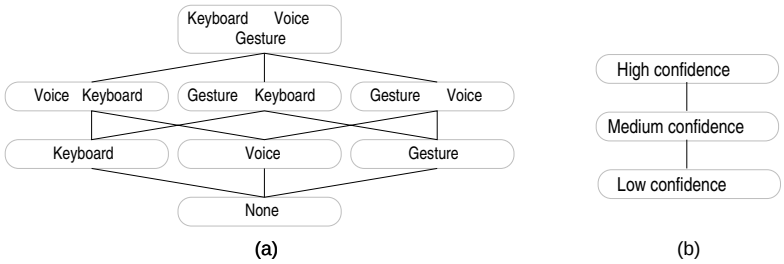

Fig. 5. (a) The partial order used to represent different input modalities. (b) Confidence levels of the input modalities.

(see, e.g. the input *"The units in here +* <circle>*"*) and the objects themselves. For this reason, we develop an extension to the type hierarchy that allows a specification of methods as state constraints. Every time the status of the state constraint changes, the back-end application is notified about the constraint and all necessary parameters and can execute the relevant action. This is comparable in function to the handlers proposed by Rudnicky and Wu [13].

3.1 Method Specification

The domain model employed in the dialogue system uses a simple class hierarchy (see section 2). Class specifications may contain variables (whose type is a class from the ontology) and methods (whose arguments are classes from the ontology). In addition, class specifications may be related through multiple inheritance. While in conventional object-oriented design, objects in the domain correspond to classes, actions of the objects correspond to methods, and properties correspond to variables, we chose to model each of objects, actions and properties of objects and actions by classes. First, this allows us to uniformly express mappings from noun phrases, verbal phrases and adjuncts to classes. Second, any constituent of a spoken utterance may be underspecified.

3.2 Method Invocation

The type inference procedure for feature structures [2] can be generalized in a straightforward manner to include method specifications. More specifically, the method specifications form a partial order (defined by the subsumption ordering of the argument constraints) over which the type inference procedure can be defined. If the type of a feature structure is at least as specific as the type for which the method is defined and all constraints on the arguments are satisfied, the method specification is added to the feature structure. It can be seen easily that this extension is monotonic, increasing and idempotent as is required for inference procedures. Moreover, as the number of method specifications is finite, the type inference procedure halts.

A method specification does not implement any particular behavior of the class it belongs to. Rather, it detects that the informational content of a representation is specific enough for a procedure to be invoked. For this reason,

the addition of a method specification to a feature structure through type inference generates an event to the back-end application. It is then the task of the back-end application to carry out the functionality associated with the method. As an example, consider a class obj_displayable with an associated method display() and the constraint string < obj_displayable.name,int < obj_displayable.x,int < obj_displayable.y (read: the variable obj_displayable.name contains more information than the fact that it is a string, i.e. it is instantiated). We thus have

$$typeinf\left(\begin{bmatrix} obj\_tank & \\ \text{NAME} & \text{“}Bravo\text{-}1\text{”} \\ \text{X} & 153 \\ \text{Y} & 529 \end{bmatrix}\right) = \begin{bmatrix} obj\_tank & \\ \text{NAME} & \text{“}Bravo\text{-}1\text{”} \\ \text{X} & 153 \\ \text{Y} & 529 \\ display(n,x,y) & \\ n = \text{“}Bravo\text{-}1\text{”}, x = 153, y = 529 & \end{bmatrix}$$

As soon as the position and the name of the object become known to the dialogue system, the type inference adds the instantiated method signature to the feature structure and sends and event to the back-end application. The event contains a unique identifier of the representation, along with its name and coordinates as declared in the method specification. Should a description of an object refer ambiguously, an event is generated for each retrieved object that verifies the constraint. Not only does this approach provide a declarative way of specifying behavior and abstract over the form of the dialogue, it also decouples the natural language understanding component from the application itself in a natural way.

In this way, the method invocation interacts nicely with another characteristic of our approach to object-oriented design. While traditionally an instance of a class is an object, in dialogue processing an instance of a class can only be a (possibly incomplete) description of an object. Necessary information for object instantiation may be missing and can only be acquired through dialogue. Since descriptions of objects do not need to refer uniquely to objects, procedural method invocations become more complicated. For this reason, we chose the declarative approach to method invocation over a procedural one.

4 Informational Characterization of Dialogue States

Traditionally, one approach to describing dialogue is to explicitly model dialogue states and transitions. Here, all possible states all well as transitions through the state space need to be anticipated and specified during system design. The difficulties of the specification are aggravated as soon as behavioral patterns need to be replicated for each of the states. For example, when a misunderstanding occurs, it is a common dialogue strategy to repeat confirmation questions for any given filler only a few times. In finite state based dialogue managers, the uncertainty of the information is thus modeled by the state the dialogue manager is in.

Recent dialogue managers allow more flexible interaction through the specification of *dialogue goals* [6] or *forms* [12] which, when filled out entirely, adequately represent the users' intention. It is the duty of the dialogue manager to determine through interaction with the user which form to choose and how to

fill its slots with values. This approach of information-based dialogue management gives up on the notion of an explicit state of the dialogue system. At the same time, it is very useful to make an assertion pertaining to the state of the dialogue manager, e.g., *the dialogue manager is in a state where conversational breakdown has occurred.* The information contained in these abstract states is then used to select appropriate dialogue strategies.

4.1 Constraint Logic Programming

In order to abstract away the concrete information available in the discourse and to arrive at a characterization of the dialogue state, we use a constraint logic program [8] to determine abstract states. The constraints are of the form

$$i : c \sqsubseteq_i x \qquad i : c \text{ is compatible to } x \qquad i : c \text{ unify } x$$

where i identifies the partial order P_i, $c \in P_i$ is an element from the partial order, and x is a variable instantiated with a multidimensional typed feature structure.

4.2 Characterization of States

The dialogue state is characterized by a set of five variables $s_1, \ldots, s_5$. These variables express the confidence in the representation of the current turn, the confidence of the overall dialogue, the speech act of the current utterance, whether or not the intention of the user could be determined uniquely, and whether or not referring expressions in the current utterance have unique or ambiguous referents, respectively. The values of the s_i range over one of the partial orders and are determined by the constraint logic program. Figure 6 lists the possible values for the five state variables. Details on a unimodal veriant of this approach as well as the determination of the users' intention are described in more detail in [5].

Variable	Meaning	Values taken from
s_1	modality confidence	confidence order $\times$ input modality order
s_2	dialogue confidence	confidence order
s_3	speech act type	class hierarchy
s_4	users intention	{none,unique,ambiguous}
s_5	reference of referring expressions	{none,unique,ambiguous}

Fig. 6. Values of the Dialogue State Variables

The integration of the multimodal information is achieved through additional clauses where the parameters are constrained so as to ensure the combination of appropriate representations.

4.3 Specification of Strategies

Additional clauses rely on the characterization of the informational state to decide the next action of the dialogue system. For example, if the confidence in the current utterance is medium, but the confidence in the overall dialogue is high, the system decides to ask for confirmation. Then, appropriate clauses determine the semantic content of the confirmation question, select the template, generate the clarification question and pass it on to the output module. The dialogue state variables decouple thus a concrete application specific dialogue state from a dialogue strategy that can be formulated in an application independent fashion.

5 Conclusion and Future Work

We have argued that a framework for multimodal dialogue systems not only needs to address the integration of information from different input streams, but also needs to be capable of representing and reasoning about input sources, input reliability and dialogue states. We have presented a framework for multimodal dialogue systems consisting of three central aspects addressing these requirements. First, we showed how multidimensional feature structures, a generalization of typed feature structures, can be used as a unified representational formalism for representing information stemming from different input sources. Second, we introduced an object-oriented extension to the feature structures that allows applications to receive notifications of state changes in the representations, to be employed for example for decentralized updates of displays. Finally, we demonstrated how the clauses of a constraint logic program over the multidimensional feature structures can be used to informationally characterize the informational content. These more abstract dialogue states are then used to determine appropriate dialogue strategies.

The work closest to ours is probably the the work by Johnston *et al* [9, 10]. In this work, multimodal input is represented in the standard types of typed feature structures and combined on a sentence level. The difference between this work and ours consists in the fact that the former encodes spatial information in the feature structures directly and relies on procedures external to the logic to perform the integration of multimodal input (e.g., intersection algorithms taking lists of types representing the coordinates of the points). In our work, however, this property is built in through the different dimensions in the feature structures and the combination with constraint logic programming. In addition, we propose object-oriented extension enabling the back-end application to track the state of the multimodal discourse and mechanisms to integrate information beyond the sentence level. Finally, the multimodal structures also allow to represent information that is necessary for guiding a multimodal dialogue; thus, the proposed representations enable the interaction to extend over the sentence level.

Future work includes the addition of logic to allow the dialogue system to determine an appropriate modality for the information being queried.

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. B. Carpenter. *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1992.
3. P.R. Cohen, M. Johnston, D. McGee, S.L. Oviatt, J. Clow, and I. Smith. The Efficiency of Multimodal Interaction: A Case Study. In *Proceedings of the International Conference on Spoken Language Processing,Sydney*, pages 249–252, 1998. Available at http://www.cse.ogi.edu.
4. P.R. Cohen, M. Johnston, D. McGee, S.L. Oviatt, J. Pittman, I. Smith, L. Chen, and J. Clow. Quickset: Multimodal Interaction for Distributed Applications. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 1997.
5. M. Denecke. Informational Characterization of Dialogue States. In *Proceedings of the International Conference on Speech and Language Processing, Beijing, China*, 2000.
6. 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.
7. G. Ferguson and J.F. Allen. TRIPS: An Integrated Intelligent Problem-Solving Assistant. In *Proceedings of AAAI/IAAI*, pages 567–572, 1998.
8. J. Jaffar and J. L. Lassez. Constraint Logic Programming. In *Proceedings 14th ACM Symposium on Principles of Programming Languages, Munich*, pages 111–119, 1987.
9. M. Johnston. Unification-Based Multimodal Parsing. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 98), Montreal, Canada*. Association for Computational Linguistics Press, 1998. Available at http://www.cse.ogi.edu.
10. M. Johnston, P.R. Cohen, D. McGee, J.A. Pittman S.L. Oviatt, and I. Smith. Unification-Based Multimodal Integration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics Press, 1997. Available at http://www.cse.ogi.edu.
11. E. Levin and R. Pieraccini. Spoken Language Dialogue: From Theory to Practice. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, 1999.
12. K.A. Papineni, S. Roukos, and R.T. Ward. Free-Flow Dialogue Management Using Forms. In *Proceedings of EUROSPEECH 99, Budapest, Ungarn*, 1999.
13. A. Rudnicky and X. Wu. An agenda-based Dialog Management Architecture for Spoken Language Systems. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, 1999.
14. B. Suhm, B. Myers, and A.H. Waibel. Model-Based and Empirical Evaluation of Multimodal Interactive Error Correction. In *Proceedings of the CHI 99, Pittsburgh, PA*, 1999. Available at http://www.is.cs.cmu.edu.
15. M.T. Vo. *A Framework and Toolkit for the Construction of Multimodal Learning Interfaces*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1998. Available at http://www.is.cs.cmu.edu.
16. M.T. Vo and C. Wood. Building an Application Framework for Speech and Pen Input Integration in Multimodal Learning Interfaces. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1996. Available at http://www.is.cs.cmu.edu.