An Inference-based Approach to Dialogue System Design

Johan Bos and **Tetsushi Oka**
Institute for Communicating and Collaborative Systems
Division of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW, Scotland, UK
{Johan.Bos,Tetsushi.Oka}@ed.ac.uk

Abstract

We present an architecture for spoken dialogue systems where first-order inference (both theorem proving and model building) plays a crucial role in interpreting utterances of dialogue participants and deciding how the system should respond and carry out instructions. The dialogue itself is represented as a DRS which is translated into first-order logic for inference tasks. The system is implemented as a society of OAA-agents, and evaluated against a specific application (home automation).

1 Introduction

This paper presents an inference-based approach to human-computer dialogue understanding in both its theoretical form and its practical implementation. In this framework, contributions of the dialogue participants are recorded in a semantic representation, and logical inferences are drawn to manage the direction of the dialogue from the system's point of view: when to perform which action, how to answer a question, and whether to show agreement or disagreement towards the user's contribution. This paper specifically addresses the interpretation of instructions commanded by dialogue participants.

The key problem in dialogue management is to make sense of vocal contributions of other dialogue participants and to generate a suitable reaction. The role of inference to play here is in the first place distinguishing consistent from inconsistent states of affairs. In the inference-based approach to dialogue understanding the aim is to find a *consistent* semantic representation capturing the meaning of the dialogue. If a consistent interpretation cannot be found, we have got a signal that something is going wrong in communication. Such a situation might arise from disagreement or misunderstanding between dialogue participants.

But the use of inference goes further than just triggering disagreements. It contributes to ambiguity resolution (for instance the resolution of pronouns and other anaphoric expressions) and helps finding preferred interpretations. Moreover, with the help of *model building*, inference can be used for performing actions and answering questions, too. We argue that, with the help of inference tools, it is possible to design a dialogue system that is more easily portable to new domains than existing dialogue systems, which core is based on a semantically-driven, domain independent dialogue move engine.

What kind of logical representations are suitable for this task and what kind of inference rules should we use? There are two requirements to build a prototype dialogue system. Firstly, the representation language should have a level of expressiveness capable of capturing a substantial amount of natural language's meaning. Secondly, because we aim to implement a genuine inference component, the logic of choice should be supported by an arsenal of inference engines that provide enough capacity to implement interesting dialogue behaviour. These requirements balance each other out: on the one hand we want to have a rich representation language, but on the other hand we want inference tools that perform their tasks in reasonable time. We believe that *first-order logic* fulfills this task as the best of these worlds.

First-order logic is certainly strong enough to capture many interesting natural language phenomena (and is able to give useful approximations of phenomena that require higher-order descriptions). For instance, the formalism that we are adopting, Discourse Representation Theory (DRT, Kamp and Reyle (1993)), deals with context-sensitive phenomena, and offers a translation from Discourse Representation Structures (DRSs) to first-order formulas. Moreover, nowadays there are many out-of-the-box inference tools available for first-order logic (*theorem provers* and *model builders* have seen an enormous increase in performance during the last decades). We will use automated theorem provers to check for potential inconsistencies arising in the dialogue. Simultaneously, we will use automated model builders that generate first-order models and thereby show consistency of the dialogue.

Especially the use of model builders is of importance in our inference-based approach to dialogue analysis. Model builders (for first-order logic) have pleasant properties for semantic processing: they offer a positive handle on the satisfiability problem (theorem provers offer a negative handle by trying to find a counter-proof), and they are able to construct finite minimal *first-order models* for a theory. The representations of these models contain no recursion and therefore are extremely easy to process. Hence, they are much more convenient to use for dialogue management than the logical representations that gave rise to these models.

Model building approaches for text understanding are proposed by Gardent and Konrad (2000) and Ramsay and Seville (2000). The approach in this paper differs from these in the fact that we actually use these models to carry out concrete actions in dialogue. In our view, models are not just formal objects—they are objects with fantastic practical implications.

This paper introduces an inference-based framework for dialogue system design (focusing on instructive dialogues) and is organised as follows. Section 2 formulates an extension of DRT that covers a wide variety of dialogue phenomena and offers a (modal) translation to first-order logic. The interpretation of this DRS language is given in Section 3. We present the inference-based approach for dialogue system design in Section 4, where we show the use of inference (both theorem proving and model building) in dialogue management. Then, in Section 5, we present an implementation of a general dialogue system using state-of-the-art theorem provers and model builders as part of the inference component. Finally, in Sections 6 and 7, we evaluate the use of our framework by using it for a specific application (home automation).

2 Representing Dialogue

We will use Discourse Representation Structures to represent the meaning of the dialogue between user and system. There are three reasons that motivate this choice of formalism. First and foremost, DRT is a well understood framework and covers a wide variety of linguistic phenomena (Kamp and Reyle, 1993; Van der Sandt, 1992). These phenomena include context-sensitive expressions such as pronouns and presuppositions. To our knowledge, there is no other semantic formalism that comes close to the empirical coverage of DRT. Second, there now exist computational implementations that provide means to extend existing linguistic grammars with DRS-construction tools, and there are efficient algorithms available that implement Van der Sandt's presupposition projection algorithm for DRT (Blackburn et al., 2001). Third, there is a direct link between DRT and first-order logic—there is a translation from DRSs to formulas of first-order logic that behaves linear on the size of the input.

DRT was initially designed to deal with texts, so we will use an extension of standard DRT that enables us to cope with certain dialogue phenomena. This extension introduces first of all *actions* into the DRS-language, but it also covers up *modal* expressions and *questions*. This extended DRS-language is interpreted by a translation function from DRSs to first-order logic that covers these extensions as well.

Let us first look at the DRS language itself. Basic DRSs have two components: a set of *discourse referents*, and a set of *conditions* upon those referents. Discourse referents stand for objects mentioned in the course of the dialogue. Conditions constrain the interpretation of these discourse referents. More formally, DRSs and merge of DRSs are defined in the usual way:

Syntax of DRSs

1. If $\{x_1, \ldots, x_n\}$ is a set of discourse referents, and $\{\gamma_1, \ldots, \gamma_m\}$ is a set of DRS-conditions, then the ordered pair $\langle\{x_1, \ldots, x_n\}, \{\gamma_1, \ldots, \gamma_m\}\rangle$ is a DRS;

2. If B_1 and B_2 are DRSs, then so is $(B_1;B_2)$.

In order to deal with imperatives in dialogue, we extend the DRS language with *action terms* following Lascarides (2001). Atomic action terms are identified by the δ-operator. Actions can be of composite nature. Complex action-terms are composed out of other action terms by either ";" (sequence) or "|" (free choice).

Syntax of DRS-action-terms:

1. If B is a DRS, then δB is a DRS-action-term;
2. If A_1 and A_2 are DRS-action-terms, then so are (A_1;A_2) and ($A_1|A_2$).

The DRS-conditions (see below) subsume those of standard DRT. Further we have the modal operators $\Box$ and $\Diamond$ (clauses 3 and 6). Hybrid DRS-conditions formed by discourse referents and DRSs (clause 5). Clause 6 describes necessary and possible effects of actions, and clause 7 introduces DRS-conditions that describe actions that are commanded.

Syntax of DRS-conditions:

1. If R is a relation symbol for an n-place predicate and $x_1 \ldots x_n$ are discourse referents then $R(x_1,\ldots,x_n)$ is a DRS-condition;
2. If x_1 and x_2 are discourse referents, then $x_1 = x_2$ is a DRS-condition;
3. If B is a DRS, then $\neg$B, $\Box$B, $\Diamond$B, ?B are DRS-conditions;
4. If B_1 and B_2 are DRSs, then $B_1 \vee B_2$, $B_1 \Rightarrow B_2$, B_1?B_2 are DRS-conditions;
5. If x is a discourse referent and B a DRS, then x:B is a DRS-condition;
6. If A is a DRS-action-term, and B a DRS, then [A]B and $\langle$A$\rangle$B are DRS-conditions;
7. If A is a DRS-action-term then !A is a DRS-condition.

Because hybrid DRS-conditions (clause 5) are non-standard, they deserve some extra attention. The domain D of interpretation is sorted, consisting of D_i (the individuals) and D_w (the possible worlds). Depending on their sort, discourse referents either denote individuals or possible worlds. So, what a DRS-condition of the form x:B effectively does is explicitly picking out a possible world, stating that the information expressed by the DRS B holds in the world denoted by x. An example DRS is shown in Figure 1.

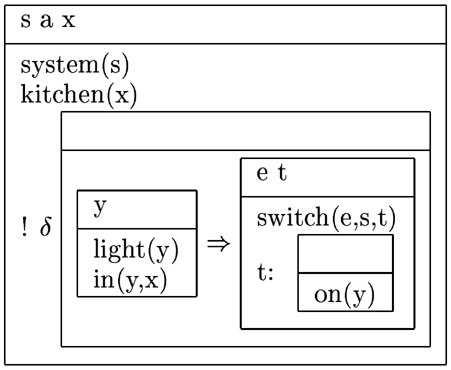

Figure 1: Example DRS paraphrasing the utterance "Switch every light in the kitchen on".

3 Interpreting Instructions

The translation to first-order covering our extended DRS-language is based on the relational translation for modal logic to first-order formulas (Moore, 1980). It is essentially similar to the standard translation from DRT to first-order logic (Kamp and Reyle, 1993), extended with Moore's rules to deal with the modal operators. DRS-action-terms are translated into three-place relations where the first argument denotes the current state, the second argument describes the actions, and the third argument denotes the resulting state. (Due to space restrictions we won't give a formal definition of the translation function here.) The example in Figure 1 would be translated into first-order logic as follows:

$\exists w \exists s \exists a \exists x$ (possible_world(w) $\wedge$ system(w,s) $\wedge$ kitchen(w,x) $\wedge$ $\exists v \exists a$ (action(w,a,v) $\wedge$ $\forall y$ (light(a,y) $\wedge$ in(a,y,x) $\rightarrow$ $\exists e \exists t$ (switch(w,e,s,t) $\wedge$ on(t,y)))))

Note that the translation increases the arity of all predicates symbols with one, where the additional argument position denotes a possible world. In a situation with one kitchen with two lights a (minimal) model satisfying this formula (and further background knowledge in the form of meaning postulates describing the preconditions and effects of switching on devices) could contain the following information:

```
D={d1,d2,d3,d4,d5,d6,d7,d8}
F(possible_world)={d1,d2,d3}
F(system)={(d1,d4),(d2,d4),(d3,d4)}
F(kitchen)={(d1,d5),(d2,d5),(d3,d5)}
F(action)={(d1,d2,d3)}
F(light)={(d1,d6),(d2,d6),(d3,d6),
          (d1,d7),(d2,d7),(d3,d7)}
F(in)={(d1,d6,d5),(d2,d6,d5),(d3,d6,d5),
       (d1,d7,d5),(d2,d7,d5),(d3,d7,d5)}
```

```
F(poweron={(d2,d6),(d2,d7)}
F(off)={(d1,d6),(d1,d7)}
F(on)={(d3,d6),(d3,d7)}
```

The nature of concrete models as produced by model builders makes them preferable to the original description in "formula" syntax for several dialogue management tasks. Since models are essentially flat structures without recursion, they are easy to process. For instance, all quantification and boolean structures are explicit in models. This makes models ideal to function as a database-lookup table to find out whether there are actions to be performed by the system. In the example model above, for instance, a possible action is to supply power to the devices d6 and d7.

4 The Update Algorithm

In the inference-based framework of dialogue processing we present here, we assume that contributions of the user are processed as semantic representations in the form of DRSs (see Section 2). We also assume a translation function from DRSs to first-order logic, and a theorem prover and model builder for first-order logic. Given this, we propose the algorithm shown in Figure 2 for utterance interpretation and move planning.

The contributions of the user are interpreted with respect to the previous context. Due to ambiguities appearing in natural language, this will result in a set of DRSs (Step 1). To make the interpretation of the dialogue sensitive to context, also a DRS of the extra-dialogic context is available (Step 2). The aim of the algorithm is to find a set of interpretations. Therefore, we start with the empty set (Step 3) and attempt to find consistent interpretations for each of the DRSs (Step 4). The theorem prover and model builder work in a complementary way. As soon as a proof is found the model builder does not need to further attempt to find a model because it will never succeed in doing so. However, if a model is found, the theorem prover can stop its attempt to find a proof, because it will never be able to do so. Finally, the system is instructed on how to react on the set of interpretations found (Step 5 and 6).

Note that this algorithm deals directly with declarative and imperative forms of utterances. Both declaratives and imperatives will be either acknowledged (by an *understanding act*) or refused (by means of a *misunderstanding act*).

```
1. Construct a (finite) set of DRSs $\mathcal{B}$ for a new utterance with respect to the previous DRS $B_{old}$;
2. Construct a DRS $C$ representing the extra-linguistic context;
3. Initialise the set of interpretations $\mathcal{I}$ to $\emptyset$;
4. For each $B_i \in \mathcal{B}$:
   (a) Translate the compound DRS $(B_i;C)$ to the first-order representation $\phi_i$;
   (b) Compute appropriate background knowledge stated as the first-order theory $\theta_i$ for $\phi_i$;
   (c) Attempt to build a model for $(\theta_i \wedge \phi_i)$ by simultaneously performing:
       i. Give $(\theta_i \wedge \phi_i)$ to a model builder, possibly resulting in a model $M_i$; (for consistent interpretations).
       ii. Give $\neg(\theta_i \wedge \phi_i)$ to a theorem prover, possible resulting in a proof (for inconsistent interpretations);
   (d) If a proof is found cancel $4(c)i$. If a model is found add $\langle B_i, M_i \rangle$ to $\mathcal{I}$ and cancel $4(c)ii$;
5. If $\mathcal{I} = \emptyset$ perform a misunderstanding act and quit. Else perform an understanding act and select a preferred interpretation $\langle B_p, M_p \rangle$ from $\mathcal{I}$;
6. Use the information in $M_p$ to decide whether to perform any actions. Replace $B_{old}$ by $B_p$.
```

Figure 2: Update Algorithm for inference-based dialogue management.

5 Implementation

The inference-based dialogue system as proposed in this paper is implemented as a collection of agents within version 2.1.0 of the Open Agent Architecture, OAA (Cheyer and Martin, 2001). OAA is a piece of middleware supporting C++, Java, Lisp and Prolog, which enables one to put components together as a working dialogue system in a prototyping environment, where agents can run on different machines. Agents (roughly corresponding to the different components) communicate via *solvables*, specific queries that can be solved by certain agents. Although the agents are connected to one central facilitator, in the inference-based dialogue system there is a separate functional hierarchy governing them. Figure 3 illustrates

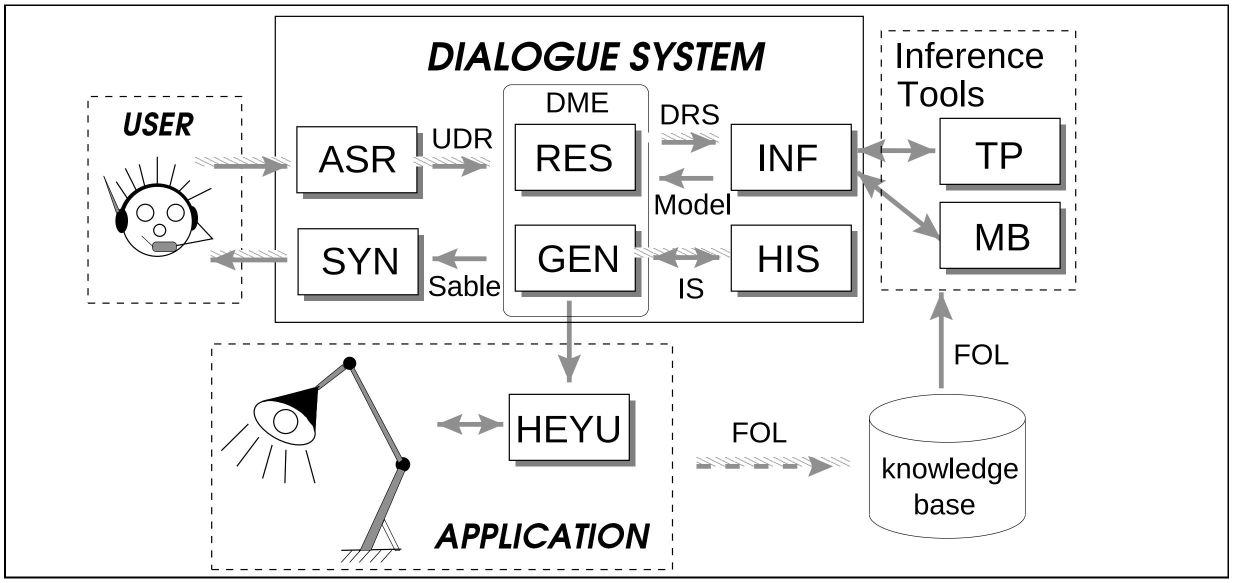

Figure 3: Architectural overview of the inference-based dialogue system. Boxes represent OAA-agents (see Section 5). The application shown here is dialogue in home automation (see Section 6).

these dependencies, and shows what kind of messages are used as interfaces between them and in what order they are sent between components. We will now describe each of these agents in more detail.

The ASR agent (Automated Speech Recognition) is implemented as the off-the-shelf (speaker independent) Nuance 7.0 speech recogniser (see `http://www.nuance.com`). The Nuance recogniser requires an application specific GSL grammar which is compiled from a linguistic unification grammar and includes semantic representations (Bos, 2002). This means that the output of the ASR agent is actually a semantic representation, more specifically it is an *underspecified discourse representation* (UDR). UDRs are proto-DRSs with still unresolved information (anaphora and scope). The SYN agent, implemented as the Festival synthesiser (see `http://www.cstr.ed.ac.uk/projects/festival/`), synthesises utterances coded in SABLE format (an XML standard for speech synthesis markup).

The DME, the dialogue move engine based on the information-state approach to dialogue management (Traum et al., 1999), updates the information state with respect to new utterances (from the user) and decides the next move of the system. The information state (IS), stored in the dialogue history HIS, is fairly straightforwardly structured, and comprising a record structure consisting of a stack of 'last-moves' and an ordered list of pairs of DRSs and models. Dialogue updates are carried out in a rule-based fashion, where the effects of update rules are applied to the information state if their pre-conditions hold. One of the update rules consults the resolution component RES to perform contextual resolution (resolving ambiguities arising from anaphoric expressions or scope bearing operators), mapping the UDR (stored in 'last-moves') into pairs of DRSs and models. Other update rules use the GEN component to generate utterances from the DRS, or generate actions from the model.

Finally, the INF agent is a mediator for inference facilities, using both theorem provers (controlled by the TP agent) and model builders (controlled by the MB agent) to find either models or counter-proofs. It collects incoming calls (DRSs), starts TP and MB agents, waits for the first result, and kills inference jobs that are doomed to fail or not anymore required. The TP agent takes the DRS, translates it into a first-order logic formula, consults the knowledge base for supporting background knowledge, and attempts to prove the negation of the resulting formula by giving it to the theorem prover SPASS (see `http://spass.mpi-sb.mpg.de/`). The MB agent, on the other hand, tries to generate a model for the translated DRS and background knowledge, using the model builder MACE (see `http://www-unix.mcs.anl.gov/AR/mace/`).

6 Example Application

To apply our inference-driven spoken dialogue prototype system to a specific scenario, we set up a home machine environment where people can talk to voiced-controlled domestic devices. The scenario we envisaged comprised a 'smart room' with three lights (referred to as red, blue, and black respectively) and a radio, that can be controlled over the standard power-net using X-10 technology (see `www.x10.com`). Users are able to control these devices by spoken commands such as turn the red light on, turn it off, switch on all devices, turn off all devices except the radio, turn on the red or blue light, turn off another light, and so on, and the system reacts with a response that either expresses understanding (for consistent instructions) or signals contradiction (for inconsistent instructions), followed by a performance of the action(s) that were requested.

To illustrate the system and the use of inference with an example, assume the user commands "turn on a light". The ASR will produce an unresolved discourse representation (UDR) and give that to the DME. The DME will resolve the UDR with respect to the DRS of the previous dialogue. This will give a set of ranked potential DRSs. Further a DRS containing the current situation of the home is created (assume for the sake of the example that the red light is turned on, and all other devices are switched off). The models generated on the basis of this information (the DRS of the dialogue combined with the DRS of the current situation) will contain a 'power-on' action for either the blue or black light with respect to the actual world. So for this example, model building naturally connects the use of the indefinite "a light" with an object that meets that description and its precondition of the command.

7 Evaluation

The issues that were addressed in evaluation were: (1) *adaptability*—how difficult is it to plug a new application in the core dialogue system? (2) *portability*—what kind of additional knowledge is required? (3) *scalability*—how well do general purpose theorem provers and model builders for first-order logic perform in real applications? We will address these issues one by one in the remainder of this section.

Adaptability To adapt the core dialogue system (Section 5) to the devices in the 'smart room' we used HEYU software (see `heyu.tanj.com`. This allowed us to create an OAA agent that was able to control the devices with simple power commands (for instance `turn on a3` switches on the device plugged in socket `a3`) and to return the current situation of all used devices (i.e., whether they are on or off). In order for the dialogue system to know what kind of devices are plugged into the X-10 sockets, the HEYU agent (see Figure 3) reads an XML encoded configuration file that specifies the devices that are used and links a meaning description (in the form of a DRS) to each device (Figure 4). This functionality enables the INF agent to request information on the current situation (the devices in the network and their status), and combine it straightforwardly with the DRS capturing the meaning of the dialogue. Hence, all inferences that are carried out take the current situation into account.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE heyu_config SYSTEM "heyu_config.dtd">
<heyu>
<use device="a1"/> <use device="a2"/>
<use device="a3"/> <use device="a4"/>
<label device="a1">
lambda(P,merge(drs([X],[black(X),light(X)]),apply(P,X)))
</label>
<label device="a2">
lambda(P,merge(drs([X],[red(X),light(X)]),apply(P,X)))
</label>
<label device="a3">
lambda(P,merge(drs([X],[blue(X),light(X)]),apply(P,X)))
</label>
<label device="a4">
lambda(P,merge(drs([X],[radio(X)]),apply(P,X)))
</label>
</heyu>
```

Figure 4: X-10 Configuration file for HEYU agent, labelling devices with DRS descriptions.

Portability Tuning our dialogue system to this new domain required only adaptations in the lexicon and knowledge base. First of all, lexical entries had to be selected to produce a new GSL grammar for the ASR agent. No changes to the grammar were needed because the GSL compiler we use employs a general domain-independent linguistic grammar. Next, the semantic descriptions of the new lexical entries had to be linked up to the core ontological database that is used to compile out background knowledge for the inferences tasks. Finally, and this is the most challenging task, domain specific axioms about actions and state changes had to be specified in first-order logic and supplied to the knowledge base (e.g. that "turning a device on" and "switching a device on" are kinds of "power-on" action, that a power-on action requires the devices to be off in the current state and causes them to be on in the resulting state, plus "frame-problem" related axioms: that power-on relations do not change the colour of devices, and so on).

Scalability The last evaluation issue—scalability with respect to inference—is partly an empirical issue, because there is little known about performances of general purpose theorem provers (originally designed for mathematical problems) applied to linguistic problems. Performances depend of course on the nature and amount of background knowledge and on the formulation of the actual problem. It should come as no surprise that unrestricted theorem proving and model building fails for large dialogues, but for small dialogues (a few utterances) in a small domain there is a reasonable real-time performance—even though using state-of-the-art inference engines (SPASS and MACE) finding proofs and generating models can take up several seconds.

To conclude, there are three main reasons for first-order theorem proving and model building to play a future role in dialogue systems. First of all, automated theorem proving, and in particular the area of model generation, is a promising emerging field. Moreover, most of the first-order inference engines, albeit general purpose machines, are designed to cope with mathematical problems, and cooperation with researchers in the area of automated deduction might improve their performance on linguistic inference problems. Secondly, the current approach is non-incremental. After a new utterance is combined with the previous DRS, the complete newly constructed DRS is translated to first-order logic and checked for consistency—without appealing to previous inference results at all. It is likely that inference-based dialogue understanding would benefit from an incremental approach, in particular with regard to model building. Thirdly, we believe that there is room for improvement on the formulation of the inference problem itself. Future work should address the use of sorted logics, include experimenting with other modal formulations, and consider to use *discourse structure* to limit the size of DRS to be checked for consistency.

Acknowledgements

This work was funded by the EU 5th Framework project D'Homme (Dialogues in the Home Environment).

References

Patrick Blackburn, Johan Bos, Michael Kohlhase, and Hans de Nivelle. 2001. Inference and Computational Semantics. In Harry Bunt, Reinhard Muskens, and Elias Thijsse, editors, *Computing Meaning*, volume 2, pages 11–28. Kluwer.

Johan Bos. 2002. Compilation of unification grammars with compositional semantics to speech recognition packages. In *Proceedings of Coling 2002*, Taipei.

Adam Cheyer and David Martin. 2001. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1/2):143–148, March.

Claire Gardent and Karsten Konrad. 2000. Interpreting definites using model generation. *Journal of Language and Computation*, 1(2):193–209.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.

Alex Lascarides. 2001. Imperatives in Dialogue. In *Proceedings of the 5th International Workshop on Formal Semantics and Pragmatics of Dialogue (BI-DIALOG)*, pages 1–16, Bielefeld, Germany.

Robert C Moore. 1980. Reasoning about knowledge and action. Technical Report 181, SRI International, California.

Allan Ramsay and Helen Seville. 2000. Models and discourse models. *Language and Computation*, 1(2):167–181.

David Traum, Johan Bos, Robin Cooper, Staffan Larsson, Ian Lewin, Colin Matheson, and Massimo Poesio. 1999. A model of dialogue moves and information state revision. Technical Report D2.1, Trindi (Task Oriented Instructional Dialogue), November.

Rob A. Van der Sandt. 1992. Presupposition Projection as Anaphora Resolution. *Journal of Semantics*, 9:333–377.