Advanced Robotics, Vol. 21, No. 1–2, pp. 209–232 (2007)
© VSP and Robotics Society of Japan 2007.
Also available online - www.brill.nl/ar

Full paper

Meaningful conversation with mobile robots

JOHAN BOS [*] and TETSUSHI OKA
ICCS, School of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, UK

Received 11 January 2006; accepted 14 February 2006

Abstract—We describe an implementation integrating a complete spoken dialogue system with a mobile robot, which a human can direct to specific locations, ask for information about its status and supply information about its environment. The robot uses an internal map for navigation, and communicates its current orientation and accessible locations to the dialogue system using a topological map as interface. We focus on linguistic and inferential aspects of the human–robot communication process. The result is a novel approach using a principled semantic theory combined with techniques from automated deduction applied to a mobile robot platform. Due to the abstract level of the dialogue system, it is easily portable to other environments or applications.

Keywords: Human–robot conversation; dialogue; discourse representation theory; mobile robot; inference; first-order logic.

1. INTRODUCTION

1.1. Motivation and aims

Physically embodied agents show more and more similarities with humans, not only in science fiction films, but also those developed in research laboratories. Indeed, robots that can speak, hear and reason have caused a growing interest in research on human–robot communication. Various aspects of communication with robots have attracted interest from roboticists over the years, due to the newly emerging application of robots in entertainment, domestic work, health care and many others areas [1]. However, it is probably fair to say that the linguistic capabilities of robots have been underdeveloped, which we believe is partly due to the so far weakly established scientific links between the fields of robotics and natural language processing.

The use of spoken language is arguably the most natural way of establishing a communication channel between a human and a robot. A successful implementation

[*]To whom correspondence should be addressed. E-mail: jbos@inf.ed.ac.uk

of a talking robot requires a good understanding of many different aspects of conversation, ranging from acoustic signals, the syntactic structure of the language, the meaning associated it and the underlying goals derived from it. We need to have the functionality to share information to be able to talk about the physical environment that we share with the robot and the robot needs knowledge of a natural language (e.g., English) to understand directions of human beings, in order to break down a complex instruction such as *Clean all the rooms on the second floor except Tim's bedroom* into action primitives that the robot is able to execute in its environment.

In short, spoken dialogue research with robots is more than just combining methods and techniques from the fields of robotics and natural language processing. New challenges arise in human–computer interaction design requiring a precise understanding of the meaning of utterances performed by the user, while integrating the knowledge of the robot within its current situation.

In this article, we present an approach to human–robot dialogue understanding in both its theoretical form and its practical implementation. In the framework that we will introduce, the meaning of utterances of the dialogue participants and other information of the situation are represented as logical forms of compositional semantics and logical inferences are drawn to manage the direction of the dialogue from the robot's point of view: when to perform which actions, how to answer a question and whether to show agreement or disagreement towards the user's contribution. The architecture and technologies for spoken dialogue with mobile robots that we introduce advance the state-of-the-art in various ways. In particular, we contribute to the following three areas:

- Combining off-the-shelf speech recognition technology and domain-independent linguistic grammars to produce logical forms for spoken utterances.
- Using logical inference to interpret instructions, questions and assertions—in particular, first-order theorem proving and model building.
- Establishing interfaces to situational knowledge of a mobile robot with that of the dialogue system in a systematic and generic way.

In addition, we employ an information-state approach to dialogue modeling, allowing us to code dialogue update rules in a declarative way. We also provide a framework to evaluate the dialogue system on real mobile robot platforms, thereby achieving a proof-of-concept of our linguistically motivated approach. In fact, our talking robot has been successfully demonstrated at museum exhibitions and in-house situations.

1.2. Related work

We mainly focus on the purely linguistic aspects of human–robot conversation. We have very little to say about non-linguistic elements of conversation such as gestures (body actions and facial expressions), and the problems of microphone placement and speech recognition in noisy environments. However, these aspects of conversation are complementary and compatible with our approach of a linguistically motivated natural language interface for a robot.

The first robot with a voice interface was SHAKEY-II, as early as 1960 [1]. At that time human–robot communication was at a pioneering and primitive stage, but both the fields of natural language processing and robotics have seen tremendous developments over the last decades. With various off-the-shelf tools at the researcher's disposal, including speaker-independent speech recognition software and high-quality speech synthesizers, a number of interesting robots with natural language interfaces have been developed over the last years, such as [2–7], and many others. For an excellent overview of these approaches the reader is referred to an article in *Advanced Robotics* [1].

Some of these are close in functionality to the system that we present in this article, but it is striking that almost all of them seem to focus on a different aspect of human–robot communication and incorporated few linguistically motivated procedures, but instead use *ad hoc* techniques such as keyword spotting for and predefined slot filling to interpret natural language utterances. The system that we introduce is the first robot that has a linguistically principled natural language interface, has a logical approach to semantic interpretation and inference and, therefore, reaches a high level of abstraction. This distinguishes our human–robot dialogue system from others; because its backbone is domain-independent, it is easy to adapt to new applications, scenarios or other robots.

1.3. Outline

This article is organized as follows. First, in Section 2, we will introduce the semantic formalism and show how we employ logical inference to interpret natural language in the context of spoken interaction with mobile robots. We will demonstrate how to use off-the-shelf automatic speech recognition to construct logical forms and describe the function of the dialogue manager. Section 3 covers the problems of controlling mobile robots, introducing path planning, position correction and navigation behavior. The framework is fully implemented on a Magellan robot. This is described in Section 4.

2. NATURAL LANGUAGE UNDERSTANDING FOR MOBILE ROBOTS

2.1. Phenomena and requirements on formalism

Ideally, communication with robots should be possible in natural language (e.g., English), using spoken instead of typed input and responding to utterances in real-time. The choice of words should be as close as possible to 'everyday usage of language'. For instance, rather than informing a robot that it is currently located at *grid cell 44-89*, we would like the robot to understand natural descriptions such as *the kitchen* and *a corridor*, or even more complex but perfectly natural expressions such as *Tim's office* and *the corridor leading to the emergency exit*, including synonyms or phrases that do not necessarily have a unique designator. Furthermore, we want a robot to be capable of dealing with the following linguistic phenomena common in dialogue.

Pronouns A linguistic device to refer back to objects mentioned previously in the dialogue (anaphoric pronouns) or expressions referring to specific objects in the current situation (deictic pronouns). Example: *I left my mug in the kitchen. Go there and take it*. Here, *there* and *it* are anaphoric pronouns, and *I* is a deictic pronoun. A correct interpretation of these kinds of expressions requires a history of all objects mentioned in the dialogue and a score of prominent objects in the current situation.

Quantifying expressions Cardinals or universal quantifiers are used to denote a specific number of objects or actions. For instance, the instruction *Clean every room!* should be mapped in a set of primitive actions corresponding to the number of rooms available in the context. Understanding quantified expressions requires a formal basis of interpreting quantification and access to the domain of quantification, i.e., the robot's environment.

Presuppositional expressions Many natural language expressions evoke *presuppositions* and a robot should be able to verify them in context. For instance, *Tim's office* presupposes that Tim has an office, *the other kitchen* presupposes that there are two different kitchens and the transitive verb *to clean* presupposes that its object argument is dirty. Calculating presuppositions requires rich lexical and ontological knowledge and tools for inference.

What kind of semantic representations are suitable for modeling these phenomena and how can we make the inferences in a practical implementation? Obviously, these two questions depend on each other. The more expressive semantic representations we will choose, the more problematic the inference task will be. On the one hand, we want to have a rich representation language; on the other hand, we would like to use inference tools that perform their tasks in reasonable time. Given the current advances in automated reasoning, we believe that first-order logic fulfils this task as the best of these worlds. First-order logic can also be used to capture a wide range of natural language phenomena [8], including those mentioned above.

The semantic formalism that we are adopting, Discourse Representation Theory (DRT), is introduced in the next section. DRT is compatible with first-order logic, and we show how to use first-order inference engines such as theorem provers and model builders to implement reasoning in a spoken dialogue system.

2.2. Semantic interpretation

A linguistic formalism that fulfils the requirements posed in the previous section is DRT [9]. DRT is a formal theory of discourse interpretation, covering a wide variety of natural language phenomena including referring anaphoric and deictic expressions, quantified expressions, plural noun phrases, and a wide spectrum of presuppositional expressions [10]. It is probably fair to say that there is no other semantic formalism that comes close to the empirical linguistic coverage of DRT.

We will use Discourse Representation Structures (DRSs, the semantic representations formalized by DRT) to represent the meaning of the dialogue between user and system. There are computational implementations that provide means to extend existing linguistic grammars with DRS construction tools that we will use to design the semantic interpretation component. The link between DRT and first-order logic is rather straightforward and elegant—there is a standard translation from DRSs to formulas of first-order logic that behaves linear on the size of the input. This link gives us the means to implement logical inference.

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. The obvious extension required for modeling dialogue is that of representing and interpreting the semantics of imperatives and interrogatives. In this article we will introduce a modified DRS language covering these extensions. We will do this by giving the syntax of DRSs and some example translations. For the model-theoretic interpretation of DRSs and the translation to first-order logic we refer to other work.

There are two kinds of DRSs: basic and complex DRSs. Basic DRSs have two components: a set of discourse referents and a set of conditions upon those referents. Discourse referents represent objects mentioned in the course of the dialogue. Conditions constrain the interpretation of these discourse referents. Complex DRSs are recursively built out of (basic or complex) DRSs. More precisely, we can define the syntax of DRSs as follows:

Syntax of DRSs

(1) If $\{\mathrm{x}_1, \ldots, \mathrm{x}_n\}$ is a set of discourse referents and $\{\gamma_1, \ldots, \gamma_m\}$ is a set of DRS conditions, then the ordered pair $\langle\{\mathrm{x}_1, \ldots, \mathrm{x}_n\}, \{\gamma_1, \ldots, \gamma_m\}\rangle$ is a DRS.

(2) If B_1 and B_2 are DRSs, then so is $(\mathrm{B}_1; \mathrm{B}_2)$.

Here clause 1 defines basic DRSs. Think of discourse referents as playing the role of first-order variables. Clause 2 defines the merge of DRSs. DRS merging is, for instance, used to represent a sequence of assertions or instructions.

Before we define DRS conditions, it is useful to introduce DRS action terms. We use DRS action terms to deal with imperatives in dialogue, following Ref. [11]. We distinguish between atomic and composite DRS action terms:

Syntax of DRS action terms

(1) If B is a DRS, then $\delta\mathrm{B}$ is a DRS action term.

(2) If A_1 and A_2 are DRS action terms, then so are $(\mathrm{A}_1; \mathrm{A}_2)$ and $(\mathrm{A}_1|\mathrm{A}_2)$.

Clause 1 defines atomic DRS action terms using the δ operator. Clause 2 covers the complex action terms, which are composed using ";" (sequence) or "|" (free choice).

Now consider the DRS conditions:

Syntax of DRS conditions

(1) If R is a relation symbol for an n-place predicate and $\mathrm{x}_1, \ldots, \mathrm{x}_n$ are discourse referents then $R(\mathrm{x}_1, \ldots, \mathrm{x}_n)$ is a DRS condition.

(2) If x_1 and x_2 are discourse referents, then $\mathrm{x}_1 = \mathrm{x}_2$ 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 $\mathrm{B}_1 \vee \mathrm{B}_2$, $\mathrm{B}_1 \Rightarrow \mathrm{B}_2$, $\mathrm{B}_1?\mathrm{B}_2$ 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 $\langle\mathrm{A}\rangle$B are DRS conditions.

(7) If A is a DRS action term then !A is a DRS condition.

Clauses 1 and 2 deal with basic DRS conditions as in standard DRT [9]. Clause 3 introduces negation, the modal operators and yes–no questions. Clause 4 covers disjunction, conditionals and wh-questions. Modal DRS conditions, formed by combining a discourse referent denoting a possible world and a DRS, are introduced in clause 5. Clause 6 describes necessary and possible effects of actions, and clause 7 introduces DRS conditions that describe actions that are commanded.

Now that we know how DRSs are syntactically formed, let us consider an example. Figure 1 shows an interpretation sequence of two utterances: a declarative sentence *I left my mug in the kitchen* and an imperative *Go there and take it!* It shows the DRS after interpreting the first utterance and the updated DRS after interpreting the second utterance in the context of the first one.

DRSs are interpreted in standard first-order models [9]. A model is usually a structure $\langle D, F\rangle$, where D is the domain of interpretation and F an interpretation function, mapping the denotations of relations to entities in D. For instance, the DRS on the left-hand side in Fig. 1 is satisfied in a model with a domain comprising four entities, where one denotes the user, one a leaving event, one a mug and one a kitchen, and where the leaving event holds in the past, the mug belongs to the user and the event is situated in the kitchen. The DRS on the right-hand side in Fig. 1 contains the information of the previous utterance. Note that the discourse referent of the pronoun *there* is resolved to *kitchen* and the pronoun *it* is resolve to *mug*. This DRS is true in a model where the robot is commanded to undertake the composite action consisting of first going to the kitchen and then taking the mug.

2.3. Inference

The model-theoretic notion of meaning that we gave in the previous section is of little value in a practical setting. Here we will show how we implement reasoning

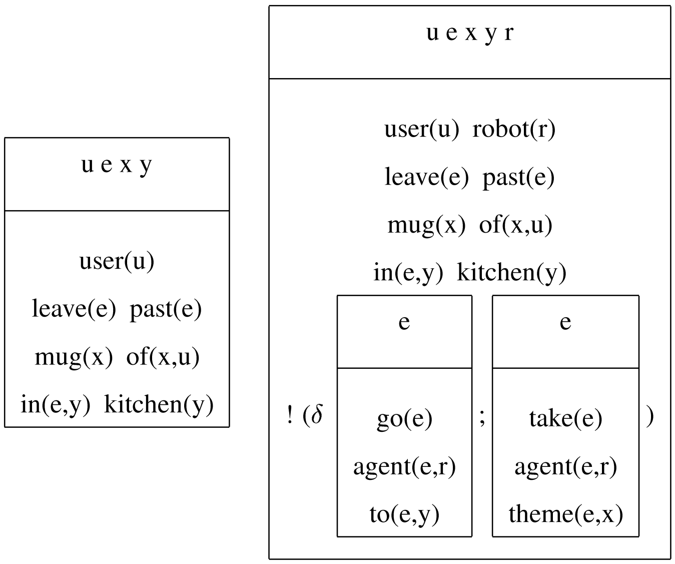

Figure 1. Example DRS paraphrasing the utterance *I left my mug in the kitchen* (left) and the DRS after updating *Go there and take it!* (right).

in a spoken dialogue system for use with an embodied agent. We will do so by first giving a concrete example illustrating inference and then outlining a more general algorithm for inference in a dialogue situation.

The DRS language used for representing the semantic content is useful for dealing with various linguistic phenomena, but how to perform any reasoning with them is a different matter. There are no (efficient) inference engines available that work directly on DRSs, so one way to overcome this is to implement a theorem prover for DRSs from scratch. However, this is a difficult and time-consuming undertaking, and there is no need for it really if one considers the other option: translating the DRSs to first-order formulas and then use theorem provers tailored to first-order logic for inference tasks. This is the option we explore in this article, because nowadays there are a couple of efficient theorem provers (and model builders) for first-order logic available.

The core of the translation to first-order covering our extended DRS language is based on the 'standard translation' from DRT to first-order logic [9]. Basically, it introduces existential quantifiers for discourse referents, unless discourse referents are declared in the universe of the antecedent DRSs of an implication, in which case they will undergo universal quantification. Sets of DRS conditions are translated into a conjunction of first-order formulas. To deal with the modal operators in our extended DRS language, we use the ideas of the relational translation for modal logic to first-order formulas [12]. This means that all basic DRS conditions with n arguments are translated into first-order relations of $n + 1$ arguments, where the additional argument plays the role of a possible world (a 'state'). DRS action terms are translated into three-place relations where the first argument denotes the current

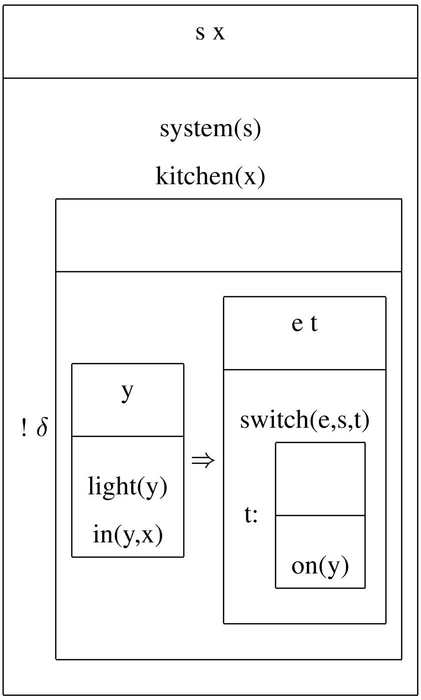

Figure 2. Example DRS paraphrasing the utterance *Switch every light in the kitchen on*.

state, the second argument describes the actions and the third argument denotes the resulting state. The full translation is given in Ref. [13]. For reasons of space we will not repeat it here, but instead illustrate the translation with an example.

Consider the DRS for *Switch every light in the kitchen on!* in Fig. 2. This example will get the following first-order translation:

$$\begin{aligned}&\exists w\exists s\exists x\big(\mathrm{possible\_world}(w) \wedge \mathrm{system}(w, s) \wedge \mathrm{kitchen}(w, x)\\&\quad \wedge \exists v\exists a\big(\mathrm{action}(w, a, v) \wedge \forall y\big(\mathrm{light}(a, y) \wedge \mathrm{in}(a, y, x)\\&\quad \rightarrow \exists e\exists t(\mathrm{switch}(w, e, s, t) \wedge \mathrm{on}(t, y))\big)\big)\big).\end{aligned}$$

Now we are ready to say something about which inference engines we will be using and which inference tasks we can perform (and how). Primarily, we will make use of two kinds of inference engines: theorem provers and model builders. A theorem prover is a tool that when given a first-order formula, tries to decide whether it is true in all possible models. A model builder, on the other hand, is a tool that tries to construct a possible model for the input formula.

Why do we need both a theorem prover and a model builder? The way we will perform semantic interpretation in a dialogue setting is by checking for satisfiability (consistency) of the ongoing dialogue (which is captured by a DRS and then translated to first-order logic). Model builders offer a positive handle on the satisfiability problem, whereas theorem provers offer a negative handle, by trying to

```
1. Construct a (finite) set of DRSs $\mathcal{B}$ for a new utterance with respect to the previous DRS $B_{\mathrm{old}}$;
2. Construct a DRS $C$ representing situational knowledge;
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);
   (a) 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 non-understanding 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 $B_{\mathrm{old}}$ by $B_p$.
```

Figure 3. Update algorithm for inference-based dialogue management.

find a counter-proof for the negation of the same input. In other words, if the model builder finds a model, we know that the information gathered from the dialogue is consistent; if the theorem prover finds a counter-proof, we know we have an interpretation problem.

Figure 3 describes this idea in the form of an algorithm. Step 1 illustrates the fact that due to potential ambiguities appearing in the utterances conveyed by the user, semantic interpretation will normally result in a set of DRSs. In Step 2, we make interpretation of the dialogue sensitive to context (see Section 2.4). We start with the empty set of interpretations (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 (Steps 5 and 6).

As the algorithm implies, the models generated by the model builder play a crucial role in interpretation. Fig. 4 shows the model generated for the DRS of Fig. 2. There are two interesting things to say about models. First, the flat representation of these models (they contain no recursion, in contrast to DRSs) ensures that they are extremely easy to process—all quantification and boolean structures are explicit in models. Second, the models produced by the model builder are minimal models and contain no irrelevant information.

We use the model to deduce what actions need to be performed by the robot or to answer questions posed by the user. For instance, in the model in Fig. 4, a desired action for the robot is to supply power to the lights `d6` and `d7`.

The algorithm itself is implemented on top of the information-state approach to dialogue modeling [14], in which an agent's information state is updated on the

```
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)}
```

Figure 4. Example model. D is the domain (a set of entities) and F the interpretation function mapping relations to (tuples of) elements of D.

basis of observed dialogue moves (a dialogue move is an abstraction of the intention of the speaker when making an utterance, such as asserting, acknowledging or questioning information), leading to the selection of a new dialogue move to be performed by the agent. This is realized by a set of update rules, linking preconditions to effects. The dialogue manager repeatedly computes the effects of those update rules whose preconditions are satisfied by the current information state. Preconditions are expressed in terms of current values in the information state, while the effects will change these values. The 26 update rules in our current system deal with establishing contact with the user, initiating clarification dialogues (when the recognition confidence score is below a certain threshold—for an example of a clarification dialogue see Section 4.4), answering questions, acknowledging requests and confirming or denying statements.

2.4. Background knowledge

Theorem provers and model builders can only make useful inferences when they have sufficient background knowledge at their disposal. The interpretation algorithm in Fig. 3 integrates background knowledge in two ways: in Step 2 situational knowledge is combined with the DRS and in Step 4b other background knowledge is computed. These other sources of background knowledge can be divided in:

- Ontological knowledge.
- World knowledge.
- Knowledge linking language to robot primitives.

Ontological knowledge comprises the various relationships between concepts that appear in the application domain. The two main relations that are expressed are subsumption and disjointness. For instance, we need to express that (in all possible worlds) a kitchen is a region and that (in all possible worlds) a kitchen is not a corridor. Both can easily be coded in first-order logic:

$$\forall w \forall x \big(\mathrm{kitchen}(w, x) \rightarrow \mathrm{region}(w, x)\big),$$
$$\forall w \forall x \big(\mathrm{kitchen}(w, x) \rightarrow \neg\mathrm{corridor}(w, x)\big).$$

World knowledge subsumes all generalisations relevant to the application domain. The so-called 'frame axioms' belong to this class and rules expressing physical laws. The frame axioms state properties of objects when certain actions are performed. For instance, if a robot moves from one region to another, its position will be different, but the positions of all other objects normally remain unaltered. As a second example illustrating a simple law of physics, when a robot wants to move from region X to region Y, it needs to be in X in order to perform that action and it will be in Y after performing the action. Again, this can be relatively easily coded in first-order logic.

Knowledge linking natural language to robot primitives can be seen as a set of meaning postulates, rules that map non-logical symbols expressing the meaning from actions to symbols understood by the robot. Put differently, these rules establish the interface between the instruction language (i.e., English) and the hardware of the robot. An example of such a rule is 'to switch on a device' and its translation into a DRS (Fig. 2), which is mapped to the action `poweron`, as can be seen in the model generated for it (Fig. 4).

Situational knowledge, finally, comprises the information of a specific situation in time. In the application domain of mobile robots, this typically consists of the regions and objects that are accessible to the robot, and the current positions of movable objects (including the robot itself). In Section 3.4 we will explain how to compute this knowledge for a specific sample scenario. We will not typically code this knowledge directly into first-order knowledge, but rather use DRSs. This allows us to combine the situational knowledge with the information from the dialogue, enabling us to interpret actions directly in the current situation.

2.5. Speech recognition

The robot that we developed is able to communicate with humans in spoken language, rather than typed. In this section we describe how we realized speech recognition and how we link the output of the speech recognizer to the semantic representations required for dialogue modeling. It was outside the scope of our project to deal with problems related to the acoustic environment such as microphone placement and background noise. Our technical contribution is to use an existing domain-independent grammar for English and compile this into a form suitable for the speech recognizer, tuned to a specific application.

In order to use a speech recognition system in practical applications, one needs to supply a language model for the recognizer, tuned to a specific domain. There are two established methods for doing this for speaker-independent applications. The first, more traditional approach, is to build a statistical language model, mostly based on n-grams. The disadvantage of this approach is that it requires a large corpus (around 20K utterances at least) of transcribed data. The second approach is to use context-free grammars for constructing a language model. This approach does not reach the same level of accuracy, but it does not require a large corpus of domain data. A disadvantage of this approach is that writing grammar rules can be laborsome and the rules can be hard to maintain.

As we do not have large quantities of data available, we favor the grammar-based approach. We developed a new method to get away with the difficulties involved in writing grammar rules. In our system, we use Nuance's speaker-independent speech recognition system (www.nuance.com), which allows language models to be specified in Grammar Specification Language (GSL), a form of context-free grammars. Rather than coding the grammar-based language model in GSL directly, we use a generic domain-independent, but linguistically motivated grammar as a starting point. The grammar in question is a unification grammar for English, which gets translated into GSL, using the UNIANCE compiler [15].

The resulting GSL is further tuned to a particular application or scenario by using a reference corpus of prototypical utterances given to the robot. The final GSL is able to recognize all the words and grammatical constructions found in the reference corpus. This proved to be an effective way of producing language models for the speech recognizer. Moreover, it is easy to extend the coverage of the GSL: just add examples to the reference corpus and compile a new GSL grammar. Some example utterances that can be recognized are:

Go to the other corridor	(command with presupposition)
Turn left, look up and go forward	(sequence of commands)
Set your location to the living room	(position correction)
You are in the kitchen	(statement)
Where are you?	(Wh-question)
Are you in the kitchen?	(yes–no question)

Most speech recognition grammars allow for semantic interpretation by means of slot-filling: previously hard-coded slots that get instantiated by values during the process of speech recognition (e.g., 'Go to the kitchen' could result in assigning the value `kitchen` to the slot `destination`). We did not adopt the standard slot-filling paradigm for semantic interpretation, because this method, although easy to implement, is inherently tuned to a specific domain. Instead, we used UNIANCE's feature [15] to employ a sophisticated compositional semantics involving λ terms which are passed as the value of a single slot for the recognized sentence. As a result the output of the speech recognition component is a domain-independent semantic representation, i.e., a DRS as introduced in Section 2.2.

3. CONTROLLING MOBILE ROBOTS

In this section we present our control system for mobile robots. The system is designed for a mobile robot with wheels and a video camera with a pan–tilt unit, sending translational and rotational velocity commands to the motors to operate the robot's motion. It also controls the pan–tilt angle and zoom of the camera. The specific problems that we address are the way actions are managed, path planning and obstacle avoidance, position correction, and calculating situational knowledge for inference in dialogue modeling.

3.1. The mobile robot control system

The control system is a looping algorithm reading sensory input and writing motor output at regular intervals. The sensory input comprises the readings of sonars, infrared sensors, bumpers and odometry. The sonars and infrared sensors are used for detecting occupied space and obstacles, and the bumpers notify if the robot hits an obstacle or wall. The odometry component measures the position and orientation of the robot. This information is translated into motor output, generating the translational and rotational velocity of the robot, and pan/tilt/zoom information for the camera unit.

We use units called 'behaviors' to implement robot control. A behavior has its own internal memory, and uses the sensory input and motor output of the control system. While executing, it computes the motor output based on the sensory input and internal memory, and updates the memory triggered by the control system at regular intervals. It terminates when certain conditions are fulfilled. For example, a behavior to move forward repeatedly sends velocity commands in a forward direction and terminates when an obstacle is detected or the robot has reached a specified location. Therefore, it periodically checks the latest readings of the odometry, sonars and infrared sensors.

Behaviors are described as object-oriented classes and a part of the class hierarchy is shown in Fig. 5. An instance of a behavior class can be created using different parameters such as speed, direction, duration or goal.

The robot needs to be able to deal with multiple goals. For instance, the robot may receive a new verbal command from the user when it is already undertaking

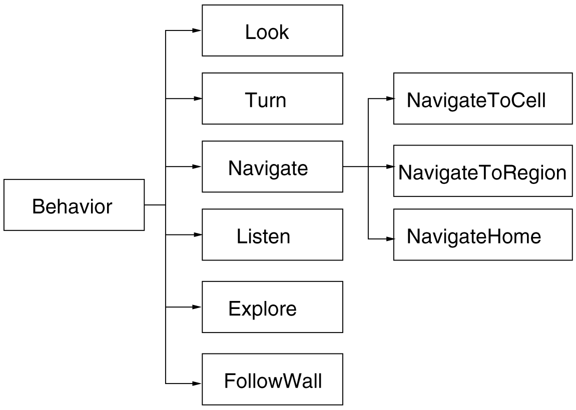

Figure 5. Classes of behaviors.

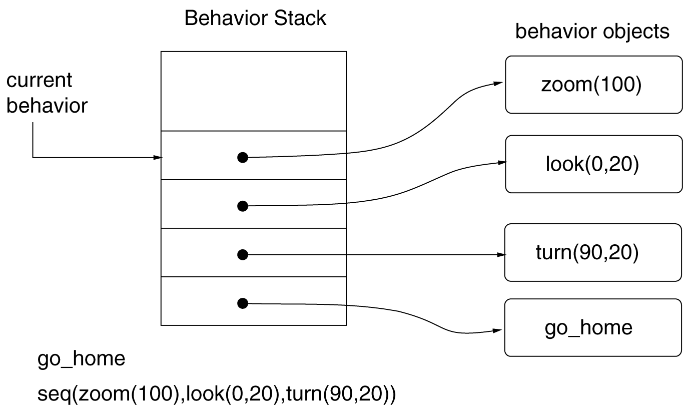

Figure 6. Behavior stack.

an action. In that case, the robot must suspend the current behavior, and create and execute another. In our control system, we use a stack of behavior instances to pursue multiple goals given by the user. The control system creates instances of behavior classes, pushes them into the stack and deletes them when they terminate.

The behavior that is executed is the one on the top of the stack. Any new command suspends the current executing behavior and starts a new behavior. When the new behavior terminates, it is popped off the stack and the previous one resumes execution. The use of a stack simplifies the way the dialogue system sends commands to the control system. It does not need to send commands to explicitly suspend, resume, terminate or remove behaviors to achieve multiple goals.

There are commands (robot primitives) creating simple behaviors such as *go(Distance, Speed)*, *turn(Angle, Speed)* and *look(Pan, Tilt)*, as well as commands introducing complex behaviors like *follow_wall(Distance)* and *move_to_region(N)*. A list of implemented behavior is listed in Table 1. Figure 6 shows the content of the stack after receiving commands *go_home* and *seq(zoom(100), look(0,20), turn(90,20))* from the dialogue manager.

The control system also accepts commands to change the activity of the robot. For example, *stop* pops the current executing behavior from the stack. Commands like *set_rotation(Angle)* and *set_location(X,Y)* change the robot's belief about its current orientation and location. The dialogue manager can also request the current location or activity of the robot. Table 2 shows commands, the dialogue manager can send to the control system. Finally, the control system not only responds to commands, but also initiates communication by sending a warning to the dialogue manager when the batteries need recharging or one of the bumpers hits an obstacle.

3.2. Path planning and obstacle avoidance

In order for a robot to reach a location specified by a user, it needs to know its own position, the target location, and how to reach it. Some of the behaviors are designed to deal with this: NavigateToCell NavigateToRegion and NavigateHome

Table 1.
Implemented behaviors

Class	Command	Description
Go	go(Distance, Speed)	move Distance [m] forward at Speed [m/s]
Turn	turn(Angle, Speed)	turn Angle [deg] at Speed [deg/s]
TurnDirection	turn_direction(Direction)	turn to Direction [deg]
Look	look(Pan, Tilt)	control the camera pan–tilt unit
Zoom	zoom(Zoom)	control the zoom of the camera
Explore	explore(T)	explore avoiding obstacles for T cycles
FollowWall	follow_wall(T)	follow the nearest wall for T cycles
NavigateToCell	move_to_cell(X, Y)	move to cell (X, Y)
NavigateHome	go_home	move to the home position
NavigateToRegion	move_to_region(N)	move to region N
DetectHuman	detect_human(T)	try to detect a human for T cycles
MoveToHuman	move_to_human(T)	detect and approach a human
FollowHuman	follow_human(T)	follow a person for T cycles
Listen	make_contact	stop, look up and wait

Table 2.
Commands to the robot control system

Command	Description
get_pose(X, Y, R)	receive current location and orientation
get_current_region(N)	receive current region number
get_activities(List)	receive the content of behavior stack
set_location(X, Y)	set belief of location in the internal map
set_rotation(R)	set belief of rotation
set_region(N)	set belief of current region
correct_position	correct belief of location based on sensor readings
stop	stop and remove the current behavior
stop_all	stop and remove all the behaviors in the stack
stop(C)	stop and remove behaviors which match with command C
seq(C1, C2, ...)	execute command $C1$, $C2$, ...

plan a path from the position of the robot to the target location (see Table 1). We use an internal map of the environment, and navigate the robot to the target location following the path and avoiding obstacles.

The internal map has two levels of representation: a geometrical and a topological layer. The geometrical layer uses an occupancy grid to represent occupied and free space in the environment. The topological layer is automatically constructed from the occupancy grid-dividing the space into natural regions. The centers of regions and the region number of each grid cell are also calculated automatically from the geometrical map.

The topological layer is used to compute the shortest path from one region to another. Given the grid cell of the center of the next region, the system plans a

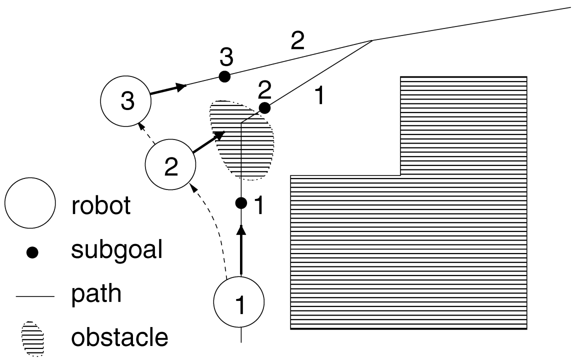

Figure 7. Following paths and subgoals.

path from the current location to the grid cell on the geometrical map by means of a distance transform path planner [16]. After planning a path, one of the cells in the path is selected as the next subgoal. Whenever the robot enters a new region, a new path to the next region is planned.

When the robot is moving to a target location, the velocity commands are computed at regular intervals based on the sensor readings, the current position and orientation of the robot, and the position of the current subgoal in the path. If the robot detects an obstacle in its way using the sonars and infrared sensors, it tries to avoid it by selecting a new grid cell in the path as a subgoal. If the robot becomes too distant from the planned path after avoiding an obstacle, a new path from the current position to the goal is planned. This technique is illustrated in Fig. 7, where the robot follows path 1 and subgoal 1 at location 1, path 1 and subgoal 2 at location 2, and path 2 and subgoal 3 at location 3, respectively, thereby successfully avoiding the obstacle.

3.3. Position correction

In practice, odometry is not very reliable and its errors are unpredictable. Hence, we need a robust control system that corrects the location and orientation. We do this by detecting walls using the sonars and infrared sensors, under the assumption that walls are straight and in parallel with the X or Y axis of the robot's internal map.

Position correction can be described as follows. When a wall is detected, the location and orientation of the robot are corrected based on the angle and distance of the wall relative to it. Three parameters r_{err}, x_{err} and y_{err} can be obtained using the estimation of the correct orientation and location, r_{est}, x_{est} and y_{est} and the odometry readings X, Y and R. Now, r_{err} can be updated whenever a wall is detected, and x_{est} and y_{est} are determined based on r_{est}, the distances to walls and the current hypothesis of the position:

$$x_{\mathrm{err}} = x_{\mathrm{est}} - X, \quad y_{\mathrm{err}} = y_{\mathrm{est}} - Y, \quad r_{\mathrm{err}} = r_{\mathrm{est}} - R. \tag{1}$$

Once x_{err}, y_{err} and r_{err} are obtained, the current location and rotation are recalculated with the help of the odometry readings at every time step t:

$$x(t) = X(t)\cos(r_{\mathrm{err}}) - Y(t)\sin(r_{\mathrm{err}}) + x_{\mathrm{err}}, \tag{2}$$

$$y(t) = X(t)\sin(r_{\mathrm{err}}) + Y(t)\cos(r_{\mathrm{err}}) + y_{\mathrm{err}}, \tag{3}$$

$$r(t) = R(t) + r_{\mathrm{err}}. \tag{4}$$

3.4. Computing situational knowledge

Recall that the geometrical layer uses an occupancy grid to represent occupied and free space in the environment. The topological layer is automatically constructed from the occupancy grid by subdividing the free space into distinct topological regions corresponding to rooms or parts of the corridor (Fig. 8). This is possible by creating a generalized Voronoi diagram [16–18].

The numbers in the geometrical map shown in Fig. 8 are identifiers of topological regions which can be seen as nodes of an undirected graph. A further layer of representation connects the map of the navigation system with a vocabulary of semantic symbols used by the dialogue system. This is done by associating semantic descriptions to regions (Table 3). These descriptions can be arbitrarily complex. For instance, the DRS $\lambda p.(\langle[x, y], [\mathit{office}(x), \mathit{of}(x, y), \mathit{tim}(y)]\rangle; p(x))$ could be used to denote Tim's office.

3.5. Detection of conversational partners

There are two simple ways to communicate with the robot: one where you can call it from a distance, using a microphone placed on the user, and one where you need to be near the robot, because the microphone is situated at the robot. Both ways have

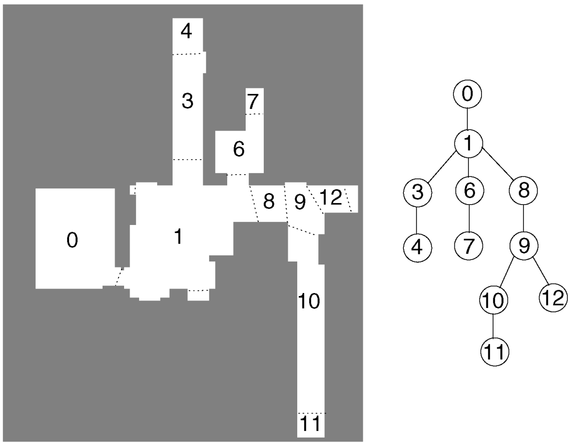

Figure 8. Representation of the environment.

Table 3.
Semantic labels

Region	Description	Semantic label
0	the office	$\lambda p.(\boxed{\begin{array}{c} \text{x} \\ \hline \text{OFFICE(x)} \end{array}}; p(x))$
1	the hallway	$\lambda p.(\boxed{\begin{array}{c} \text{x} \\ \hline \text{HALLWAY(x)} \end{array}}; p(x))$
3, 4, 8, 9, 10, 11, 12	the corridor	$\lambda p.(\boxed{\begin{array}{c} \text{x} \\ \hline \text{CORRIDOR(x)} \end{array}}; p(x))$
6	the rest room	$\lambda p.(\boxed{\begin{array}{c} \text{x} \\ \hline \text{REST\_ROOM(x)} \end{array}}; p(x))$

their advantages and disadvantages. However, the second approach is particularly challenging from the point of view of modeling dialogue, as the robot needs to be aware when a potential dialogue participant is near enough to start or continue a conversation.

We implemented several behavior classes for this kind of interaction mode that allow the robot to detect humans moving around the robot. It monitors changes in sensor readings and images from the camera, and computes the direction of a moving object. It also detects human faces in camera images using calor information and a face detection technique presented in Ref. [19]. We also implemented behaviors to turn to, approach and follow a human using information from both distance and visual sensors.

With the ability of human detection, the robot can initiate conversation without waiting for the first speech input from a user and need not wait for an utterance when nobody is around. We are also planning on implementing sound source localization to improve the currently implemented behaviors.

4. IMPLEMENTATION

4.1. The robot platform

Our robot, Godot, is based on an RWI Magellan Pro mobile robot platform with an on-board PC (Pentium 3 450 MHz) and has a laptop computer (Pentium 4 2 GHz) on the top (Fig. 9). It is about 150 cm high and its cylindrical base is 41 cm in diameter. Godot's sensor equipment consists of 16 sonars, infrared sensors and bumpers, and an odometry component. The camera unit is on a slider so that the height and angle could be adjusted. The two on-board computers (one is used to run the navigation software, the other to run the spoken dialogue system) are connected to each other via a crossover ether cable. Godot also has a wireless LAN interface to communicate with computers on our local network.

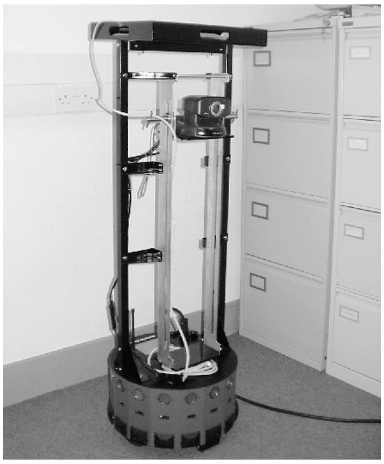

Figure 9. Godot, the robot.

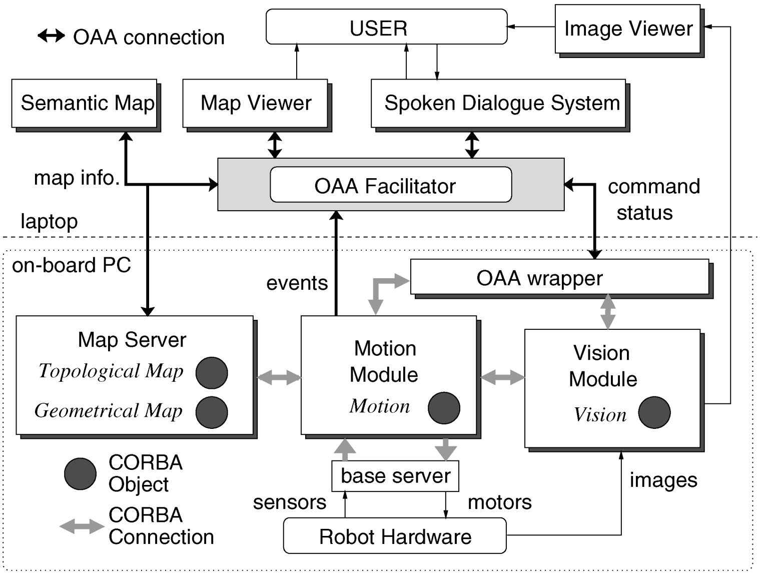

Figure 10. System architecture.

The dialogue system is implemented on top of the Open Agent Architecture (OAA). Godot, however, uses CORBA for inter-process communication. Figure 10 shows how the two systems are implemented and combined into one working system.

4.2. The spoken dialogue system

The dialogue system is implemented on top of OAA 2.1.0 [20], a hub architecture for software agents. OAA agents can run on different machines and even on different platforms and they communicate with each other by posing solvables via a facilitator, supporting various programming languages including C++, Java, Lisp and Prolog.

The system is put together out of OAA agents for speech recognition (Nuance 8.0), speech synthesis (rVoice), semantic interpretation (resolving ambiguities), inference (building models or finding counter-proofs) and dialogue management. There are two further agents for model building (MACE, www-unix.mcs.anl.gov/AR/mace/) and theorem proving (SPASS, spass.mpi-sb.mpg.de/). The system is coordinated by the dialogue manager, triggering OAA solvables for speech recognition, speech synthesis and inference. It can also request information concerning the robot's position and its current environment.

4.3. The robot control system

The robot control system consists of four components which run concurrently on the on-board PC of Godot (Fig. 10). They are Linux processes which communicate with each other *via* CORBA objects. The map server stores Godot's internal representation of the environment as CORBA objects. Information stored in these objects can be retrieved by the motion and dialogue modules *via* CORBA or OAA connections.

The motion module has a CORBA object to store information that it shares with the dialogue system. The OAA wrapper has access to this object and OAA agents in the dialogue system can communicate with the motion module *via* the wrapper. The dialogue system can monitor the current state of the robot control system, e.g., the current location of the robot in the map, and can send commands to it. The motion module can send events directly to the dialogue system *via* the OAA facilitator.

In addition, we have modules for the camera view, and for monitoring the location and orientation of the robot in the map (Fig. 11). The map viewer shows the position the robot believes it in and is therefore useful for debugging. The image viewer is handy for users that remotely control the robot.

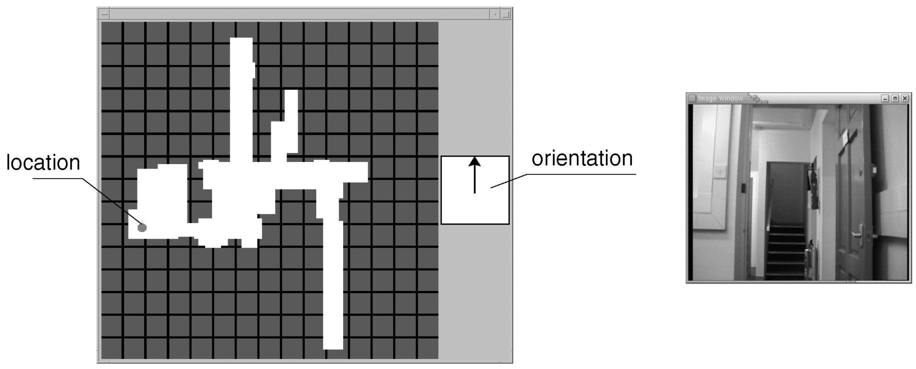

Figure 11. Map viewer (left) and image viewer (right).

4.4. Practical results

Godot is a complete end-to-end system for human–robot communication, where people can start a new dialogue simply by addressing the robot in spoken language, with the robot responding in real-time (using our Pentium 4 laptop, the Nuance speech recognizer parses utterances within their duration and the inference engines usually finds an answer in a few seconds). Although we have not yet reached the stage of carrying out formal usability studies, we have tested Godot with people unfamiliar with the system in two different settings:

- In the basement of our lab, where visitors have controlled Godot's movements and its camera with success.
- At the Scottish museum in Edinburgh, where museum visitors could address Godot in a simulated house over a 2-day period.

On the second occasion the environment was different from Godot's usual one (the basement of our lab), but our modular approach allowed us to quickly adapt it—in fact we only needed to change the internal map. The museum also turned out to be a very noisy environment. Although this resulted in a high number of clarification dialogues between the robot and user, visitors still managed to control the robot by voice. In these interactions with new users we collected the spoken data and used that to improve the system, by extending the vocabulary and grammar constructions using the method described in Section 2.5.

We finally show an example dialogue to illustrate Godot's conversational skills in practice. Recall that the behavior of the robot is controlled by dialogue moves that are triggered by update rules within the dialogue manager, as discussed at the end of Section 2.3. The example below includes answering questions, posing clarification questions, reasoning and performing actions (H indicates the utterances of the human and R the utterances of Godot):

(1)	H: Robot?	R: Yes?
(2)	H: Where are you?	R: I am in the hallway.
(3)	H: You are in the corridor.	R: No, that is not true.
(4)	H: Go to the rest room.	R: Did you mean to say "go to the rest room"?
(5)	H: Yes.	R: OK.

In (1) the user draws attention to the robot by calling it. The robot reacts to this by a short affirmative reply, indicating that it is ready to engage in conversation. The user continues in (2) by asking the robot where it is. The robot, combining its situational knowledge with the semantic representation of the question, creates a model and then generates a suitable answer to the question. In (3) the user is 'teasing' the robot and claiming that it is in the corridor. The robot attempts to build a model, but because the information in the utterance of the user conflicts with that of its background knowledge, it does not succeed (it finds a counter-proof instead). As a result, it prompts with a denial. In (4) the user commands the robot to go to the rest room. Here the speech recognizer assigned a low confidence value to the recognized utterance and therefore the robot follows up with a clarification question. Finally, in (5) the user confirms that it should go to the rest room, and the robot acknowledges this and performs the action.

5. CONCLUSION AND FUTURE WORK

Although it is tempting to work with simulated 'embodied' agents in robot–human communication, we believe that the real test of adequacy involves confronting the familiar problems of noisy sensor data and real hardware. In fact, we successfully implemented a robot with advanced communication capabilities, focusing on linguistic aspects and logical reasoning. We have developed an effective interface between natural language semantics and the robot control layer, thus enabling users to refer to locations in a natural way, rather than resorting to expressions like *go to grid cell 45, 66* or *you are in region 12*. The framework is practically domain-independent: a change of environment would only require a change of the internal map and possibly a new lexicon. We have proved the feasibility of this approach by implementing the robot Godot (we have published videos of Godot in action at http://www.ltg.ed.ac.uk/godot/). In the past we have also applied our inference-based framework to home automation [21], supporting our claim of domain-independence.

Our robot employs an inference-based approach to dialogue understanding, with the aim 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. Inference also contributes to ambiguity resolution (e.g., 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. The resulting dialogue system is formulated on an abstract, logical level (including the access to situational knowledge) and is therefore easily portable to robots in new domains or with different applications. This distinguishes our work from other human–robot platforms, which are mostly tuned to specific applications.

Future work will address the interpretation of vague expressions (*the end of the corridor*), metonymic expressions (*go to the door*, where an artifact is interpreted as a location), together with commands which require Godot to reason and talk about its current activities (*continue going to the kitchen*). We are further planning to extend the system with a sound-location detection system to improve low-level communication signals of the robot.

Acknowledgements

This work was supported by IBL for mobile robots (EPSRC GR/M90160) and the EU Project Magicster (IST 1999-29078), and was carried out at the School of

Informatics, University of Edinburgh, UK. We thank Nuance for permission to use their software and tools. We also are very grateful to two anonymous referees whose comments considerably improved the final presentation of this paper.

REFERENCES

1. R. Prasad, H. Saruwatari and K. Shikano, Robots that can hear, understand and talk, *Adv. Robotics* **18**, 533–564 (2004).
2. G. Bugmann, S. Lauria, T. Kyriacou, E. Klein, J. Bos and K. Coventry, Using verbal instruction for route learning: instruction analysis, in: *TIMR 01—Towards Intelligent Mobile Robots*, (U. Nehmzow and C. Melhuish, Eds). Department of Computer Science, Manchester University (2001).
3. J. Fry, H. Asoh and T. Matsui, Natural dialogue with the Jijo-2 office robot, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Victoria, pp. 1278–1283 (1998).
4. A. Green and K. Severinson-Eklundh, Task-oriented dialogue for cero: a user-centered approach, in: *Proc. 10th IEEE Int. Workshop on Robot and Human Communication*, Bordeaux-Paris, pp. 146–151 (2001).
5. T. Längle, T. C. Lüth, E. Stopp, G. Herzog and G. Kamstrup, KANTRA—a natural language interface for intelligent robots, *Intell. Auton. Syst.* **4**, 357–364 (1995).
6. O. Lemon, A. Bracy, A. Gruenstein and S. Peters, A multi-modal dialogue system for human–robot conversation, in: *Proc. North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA (2001).
7. M. C. Torrance, Natural communication with robots, *Master's thesis*, Department of Electrical Engineering and Computer Science, MIT (1994).
8. P. Blackburn and J. Bos, Computational semantics, *Theoria* **18** (46), 27–45 (2003).
9. H. Kamp and U. Reyle, *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht (1993).
10. R. A. Van der Sandt, Presupposition projection as anaphora resolution, *J. Semant.* **9**, 333–377 (1992).
11. A. Lascarides, Imperatives in dialogue, in: *Proc. 5th Int. Workshop on Formal Semantics and Pragmatics of Dialogue*, Bielefeld, pp. 1–16, (2001).
12. R. C. Moore, Reasoning about knowledge and action, *Technical Report 181*, SRI International, Menlo Park, CA (1980).
13. J. Bos, Implementing the binding and accommodation theory for anaphora resolution and presupposition projection, *Computat. Linguistics* **00**, 00–00 (2003).
14. D. Traum, J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson and M. Poesio, A model of dialogue moves and information state revision, *Technical Report D2.1, Trindi (Task Oriented Instructional Dialogue)* **29**, 179–210 (1999).
15. J. Bos, Compilation of unification grammars with compositional semantics to speech recognition packages, in: *Proc. 19th Int. Conf. on Computational Linguistics*, Taipei, pp. 106–112 (2002).
16. C. Theobalt, Navigation on a mobile robot, *Master's thesis*, University of Edinburgh (2000).
17. J. C. Latombe, *Robot Motion Planning*. Kluwer, Dordrecht (1991).
18. S. Thrun, Learning maps for indoor mobile robots, *Artif. Intell.* **99**, 21–71 (1998).
19. P. Viola and M. Jones, Robust real-time object detection, in: *Proc. 2nd Int. Workshop of Statistical and Computational Theories of Vision—Modeling, Learning, Computing and Sampling*, Vancouver (2001).
20. A. Cheyer and D. Martin, The open agent architecture, *J. Auton. Agents Multi-Agent Syst.* **4**, 143–148 (2001).

21. J. Bos and T. Oka, An inference-based approach to dialogue system design, in: *Proc. Int. Conf. on Computational Linguistics*, Taipei, pp. 113–119 (2002).

ABOUT THE AUTHORS

Johan Bos received his PhD in Computational Linguistics at the University of Saarland, Germany and joined the University of Edinburgh, UK for a Postdoctoral position from 2000 to 2005. Since September 2005, he is an Associate Professor at the Department of Computer Science at the University of Roma ‘La Sapienza’, Italy. His research focuses on natural language processing, in particular spoken dialogue systems, automatic question answering and computational semantics.

Tetsushi Oka is an Associate Professor at Fukuoka Institute of Technology, Japan. He received his PhD in Information Engineering from the University of Tokyo in 1996 and joined the University of Electro-Communications as a Research Associate. From September 2001 to March 2005, he was a Research Fellow at the University of Edinburgh, UK. His current research interests include autonomous robots and agents, human–robot communication, and spoken dialogue systems.