To appear in edited form in IEEE –Intelligent System & their Applications.
Special Issue on: Semi-Sentient Robot: Routes to Integrated Intelligence.

Personal Robot Training via Natural-Language Instructions.

Stanislao Lauria, Guido Bugmann[1], Theocharis Kyriacou, Johan Bos*, Ewan Klein*

Centre for Neural and Adaptive Systems, School of Computing, University of Plymouth
Drake Circus, Plymouth PL4 8AA, United Kingdom.

*Institute for Communicating and Collaborative Systems, Division of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, Scotland, United Kingdom.

http://www.tech.plym.ac.uk/soc/staff/guidbugm/ibl/index.html

19/3/2001

Abstract

Future domestic robots will need to adapt to the special needs of their users and to their environment. Programming by natural language will be a key method enabling computer language-naïve users to instruct their robots. Its main advantages over other learning methods are speed of acquisition and ability to build high level symbolic rules into the robot. This paper describes the design of a practical system that uses unconstrained speech to teach a vision-based robot how to navigate in a miniature town. The robot knows a set of primitive navigation procedures that the user can refer to when giving route instructions. A particularity of this project is that the primitive procedures are determined by analysing a corpus of route instructions. It is found that primitives natural to the user, such as "turn left after the church" are very complex procedures for the robot, involving visual scene analysis and local route planning. Thus, to enable natural user-robot interaction, a high-level of intelligence needs to be built into "primitive" robot procedures. Another finding is that the set of primitive procedures is likely not to be closed. Thus, on time to time, a user is likely to refer to a procedure that is not pre-programmed in the robot. How best to handle this is currently investigated. In general, the use of Instruction-Based Learning (IBL) imposes a number of constraints on the design of robotics systems and knowledge representation. These issues and proposed solutions are described in the paper.

1. Introduction

Intelligent robots must be capable of executing reasonably complicated tasks with some degree of autonomy. This requires adaptivity to a dynamic environment and the ability to generate plans. In the case of helper robots, or domestic robots, the ability to adapt to the special needs of their users is crucial. Most users are computer-language-naïve and cannot personalise their robot using standard programming methods. Indirect methods, such as learning by reinforcement or learning by imitation, are also not appropriate for acquiring user-specific knowledge. For instance, learning by reinforcement is a lengthy process that is best used for refining low-level motor controls, but becomes impractical for complex tasks. Learning by imitation is of limited scope. Further, both methods do not readily generate knowledge representations that the user can interrogate. An alternative method, learning from verbal instructions is explored in this paper.

Previous work on verbal communication with robots has mainly focused on issuing *commands*, i.e. activating pre-programmed procedures using a limited vocabulary (e.g. IJCAI'95 office navigation contest). Other work has focused on language learning (e.g. Roy 1999). Only a few research groups have considered learning as the stable and reusable acquisition of new procedural knowledge. An inspiring project was Instructo-SOAR (Huffman & Laird, 1995). This system used

[1] To whom correspondence should be addressed.

textual input into a simulation of a manipulator with a discrete state and action space. Another investigation (Crangle and Suppes, 1994) used voice input to teach displacements within a room and mathematical operations, but with no reusability. In (Torrance, 1995), textual input was used to build a graph representation of spatial knowledge. This system was brittle due to place recognition from odometric data and use of IR sensors for reactive motion control. Knowledge acquisition was concurrent with navigation, not prior to it. Concurrent learning was also used in Jijo-2, an "office conversant mobile robot" (Asoh et al., 1997). The present work aims at using unconstrained language in a real-world robotic application where learning occurs prior to execution. More recent projects with related scope are CARL (Seabra Lopes and Teixeira, 2000) and HERMES (Bischoff and Jain, 1999).

Instruction-Based Learning (IBL) has several potential advantages. Natural language can express rules and sequence of commands in a very concise way. Natural language uses symbols and syntactic rules and is well suited to interact with robots that have knowledge represented at the symbolic level. Accelerated learning by symbolic communication has been shown in (Cangelosi and Harnad, 2001). This was contrasted with the much slower learning at the level of direct sensory-motor associations. This paper describes initial steps and considerations towards a practical realisation of an IBL system. The experimental environment is that of a miniature town in which a robot provided with video camera executes route instructions. The robot will have a set of pre-programmed sensory-motor action primitives, such as "turn left" or "follow the road". The task of the user is to teach the robot new routes by combining action primitives using unconstrained speech.

The IBL concept, briefly discussed in section 2, involves following key tasks. First, in a corpus collection phase, samples of task-specific dialogues are recorded and analysed. This generates a list of action primitives that are natural to users (Section 3). The second task is the design of the Dialogue Manager (DM) comprising Natural Language task-specific routines, for the conversion of user utterances into a symbolic representation and for handling the user-robot dialogue. The development of the DM and its application to a dialogue from the corpus is described in section 4. A concurrent task is the design of the Robot Manager (RM) whose task is to convert the symbolic representation from the DM into executable robot procedures. An example of such conversion is described in section 5. The RM can "understand" the DM only if there is a correspondence between symbols and executable actions or real-world objects to recognize. For that purpose, a number of primitive functions associated with symbols must be pre-programmed. This follows closely the list of primitive procedures found in the corpus (Section 3). The design of these task-specific procedures is not completed yet. Eventually, tests with users will reveal if effective communication and execution can be achieved with the IBL concept. The implications of the results obtained so far are discussed in section 6, along with the question of how the proposed system compares to other approaches.

2. The IBL concept

In IBL, verbal instructions given by the user are converted into new internal program code that represents new procedures. Such procedures become part of a pool of procedures that can then be reused to learn more and more complex procedures. Hence, the robot should become able to execute increasingly complex tasks.

This process starts with a predefined initial knowledge. This “innate” knowledge consist of primitive sensory-motor procedures with names, such as “turn_left” or “follow_the_road” (the choice of primitives is explained in section 3). These names are the “symbols”, and the pieces of computer program that controls the execution of the procedures are the “actions” (Figure 1A). Each symbol is associated with an action and it is therefore “grounded” (Steels, 1998)

When a user explains a new procedure to the robot, say a route from A to B that involves a number of primitive actions, the IBL system assigns a new name to the new procedure, and writes a new piece of program code that executes that procedure (see section 4 for details). Within that code, primitive actions are referred to by name. The low-level code defining theses primitives is not duplicated therein. For that reason, the new program can be seen as a combination of symbols rather

than a combination of actions (figure 1B). As all new procedures are constructed from grounded symbols, they become grounded by inheritance and are thus "understandable" by the system when referred to in natural language. When explaining a new procedure, the user can also refer to old procedures previously defined by himself. In that way the complexity of the robot's symbolic knowledge increases (fig. 1C). Thus, a key part of IBL is the generation of program code. This is made possible by the use of a scripting language (section 4).

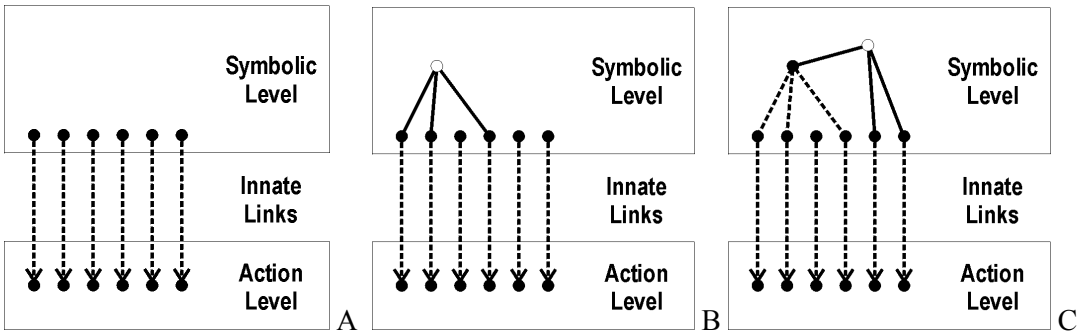
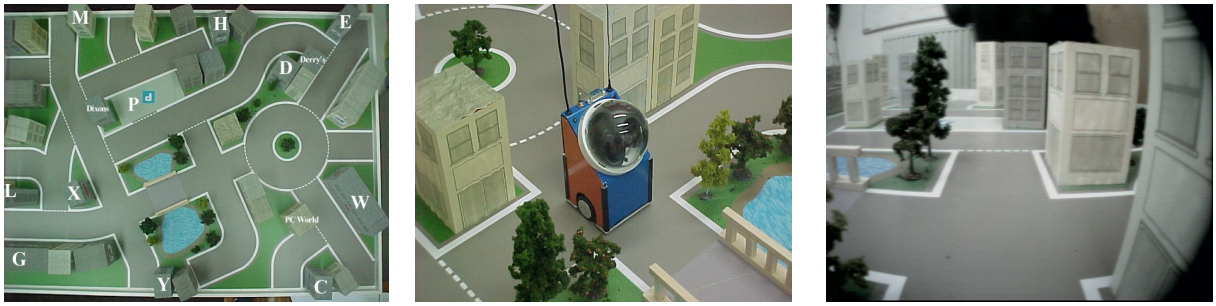


Figure 1. Symbolic learning. *(A) is a schematic representation of the initial system, comprising symbols associated with pre-programmed (innate) primitive action procedures. In (B) the user has defined a new procedure (open circle) as a combination of symbols. The new symbol is grounded because it is a construct of grounded symbols. In (C), the user has defined a new procedure that combines a procedure previously defined by himself with primitive action procedures*.

Man-machine communication is known to be error-prone. Error types range from word mis-recognition to incomplete instructions issued by the user. One of the planned error-detection mechanism is to verify if the learned sequence of action is executable. For that purpose, each procedure will be represented as a triplet $S_iA_{ij}S_j$ with properties similar to productions in SOAR (Laird et al, 1987). The state S_i is the *pre-condition* for action A_{ij}. The state S_j is the final state, resulting from the action A_{ij} applied to the initial state S_i. For a sequence of actions to be realisable, the final state of one action must be compatible with the pre-condition of the next one. If necessary, the system will attempt to satisfy this condition by dialoguing with the user to obtain additional information.

3. Corpus Collection and Data Analysis

To evaluate the potential and limitations of IBL, a real-world instructions task is used, that is simple enough to be realisable, and generic enough to warrant conclusions that also hold for other task domains. The environment is a miniature town covering an area of size 170cm x 120cm (see figure 2). A simple route scenario has been selected, using real speech input and a robot using vision to navigate the instructed route (see (Bugmann et al. 2001) for more details).



A **B** **C**

Figure 2. A- *Miniature town in which a robot will navigate according to route instructions given by users. Letters indicate the destinations and origins of various routes used in the experiment.* **B-** *Miniature robot (base 8cm x 8cm). The robot carries a CCD colour TV camera*[2] *(628 (H) x 582 (V) pixels) a TV VHF transmitter and an FM radio.* **C-** *View from the on-board color camera.Images are processed by a PC that acquires them via a TV capture card.*). *The PC sends motion commands by FM radio to the robot.*

To collect linguistic and functional data specific to route learning, 24 subjects were recorded as they give route instructions to the robot in the environment. Subjects were divided into three groups of 8. The first two groups (A and B) used totally unconstrained speech, to provide a performance baseline. It is assumed that a robot that can understand these instructions as well as a human operator would represent the ideal standard. Subjects from group C dialogued with a remote operator who induced shorter utterances by brief interactions with the subject. Subjects in groups A and B were told that the robot was to be tele-operated at a later date by a human operator following their recorded instructions. It was specified that the human operator would be located in another room, seeing only the image from the wireless on-board video camera (see figure 2C). This induced subjects to use a camera-centred point of view relevant for future robot autonomous navigation. Each subject described 6 routes having the same starting point and six different destinations. Starting points were changed after every two subjects. A total of 144 route descriptions were collected. For more details on the collection and analysis of the corpus see (Bugmann et al. 2001).

A dialogue example from the corpus collection is shown in table 1. Here the user requests the robot to go from the museum to the library. The robot responds with a request for explanation. The user replies with three utterances to explain the route (note that the route to the post-office was explained earlier in the session). We will use this example in the rest of the paper to explain the concept of IBL and its implementation. We will first show how human instructions are analysed and mapped onto meaning representations (Section 4). Then we show how the robot generates an executable plan out of that (Section 5).

```
User : Go to the library.
Robot: How do I go to the library?
User : go to the post office go straight ahead the library is on your left
```

Table 1. *Example of dialogue between a user and the remote operator. The initial position for the robot is the museum. In figure 2A, the museum, the post office and the library are indicated respectively with the letters* M, X *and* Y.

[2] Provided by Allthings Sales and Services (http://www.allthings.com.au/)

3.1 Corpus Analysis: The functional vocabulary

The aim of the corpus analysis is twofold. First, to define the vocabulary used by the users in this application in order to tune the speech recognition system for an optimal performance in the task. Secondly, to establish a list of primitive procedures that users refer to in their instructions, the "functional vocabulary". These are then to be pre-programmed so that a direct translation from the natural language to grounded symbols can take place. Hereafter, we report on the functional analysis of the corpus (Groups A and C merged. Group B not included at this point in time). The reader interested in the task vocabulary can refer to (Bugmann et al., 2001).
The annotation of instructions in terms or procedures, as reported here, is somehow subjective, and influenced by two considerations. (i) The defined primitives are to be realised as computer programs. Therefore, the corpus has been transcribed into rather few general procedures characterised by several parameters (table 2). (ii) An important issue is knowledge representation. According to the SAS representation discussed in section 2, primitive procedures always have a starting and termination condition. Subjects however rarely specified explicitly the starting point of an action and sometimes did not define the final state in the same utterance. It was assumed that the IBL system would be able to retrieve missing information from the context. Therefore, all actions referred to by subjects were assumed to be of the SAS type. For instance, when a subject specified a non-terminated action, such as "keep going", it was classified as "MOVE FORWARD UNTIL", assuming that a termination point would be inferred from the next specified action.

This analysis methodology differs slightly from the one in (Denis, 1997). In our analysis, there are no statements describing landmarks, as these are integrated into the procedure specifications, and consequently there are no actions without reference to landmarks either.

The annotation is done manually and there is no off the shelf tool for doing it automatically. The list of procedures found in the route descriptions of groups A and C is given in table 1.

	Count	**Primitive Procedures**
1	178	MOVE FORWARD UNTIL [(past |over |across) <landmark>] | [(half_way_of | end_of) street] | [after <number><landmark> [left | right]] | [road_bend]
2	118	TAKE THE [<number>] turn [(left | right)] | [(before | after | at) <landmark>]
3	94	<landmark> IS LOCATED [left | right |ahead] | [(at | next_to | left_of | right_of | in_front_of | past | behind | on | opposite | near) < landmark >] | [(half_way_of | end_of | beginning_of | across) street] | [between <landmark> and <landmark>] | [on <number> turning (left | right)]
4	49	GO [(from) <landmark> (before | after | to) <landmark>]
5	32	GO ROUND ROUNDABOUT [left | right] | [(after | before | at) <landmark>]
6	27	TAKE THE <number> EXIT [(before | after | at) <landmark>]
7	9	FOLLOW KNOWN ROUTE TO <landmark> UNTIL (before | after | at) <landmark>
8	3	STATIONARY TURN [left | right | around] | [at | from <landmark>]
9	1	TAKE THE ROAD in_front
10	1	PARK AT <location>
11	1	CROSS ROAD
12	1	EXIT [car_park | park]

Table 2. *Primitive navigation procedures found in the 96 route descriptions collected from groups A and C. Procedure 3 is used by most subjects to indicate the last leg of a route, when the goal is in sight.*

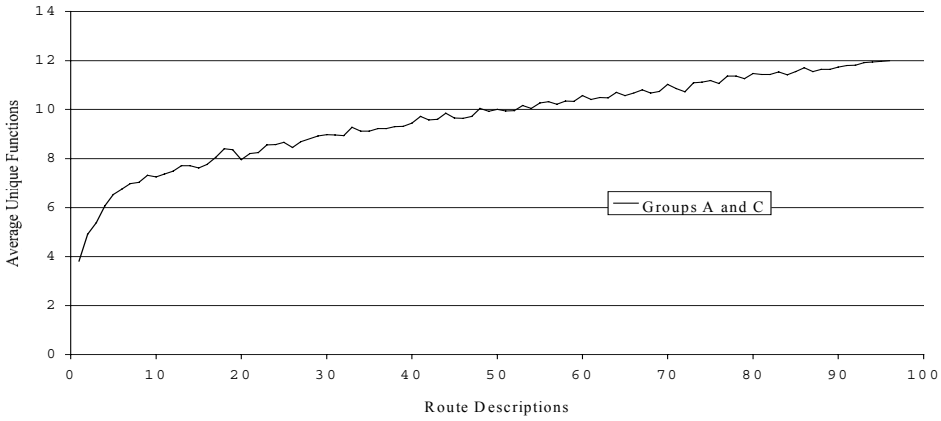


Figure 3. *Average number of unique procedures as a function of the number of collected route instructions. The curve is obtained by averaging over 96 sets comprising a random selection of n route descriptions. The number n is shown on the x-axis of the graph. The slope of the curve indicates that, on average, one new function will be added to the functional lexicon for every 25 additional route instructions collected.*

Fig. 3 shows that the number of distinct procedures is increasing with the number of sampled instructions. Here we discover on average one new procedure for every 25 route instructions. This rate is much smaller than rate with which subjects introduce new words. This was about one per route instruction in (Bugmann et al., 2001). New procedures typically are the least frequent in table 1. The implications of this finding are discussed in section 6.

4. Understanding Instructions in Natural Language

The process of understanding instructions in spoken English in the domain of personal robot training is divided into four sub-tasks: (1) speech recognition, (2) linguistic analysis, (3) ambiguity resolution, and (4) dialogue updating. These are realized by the Dialogue Manager which communicates with the user and the Robot Manager (figure 4).

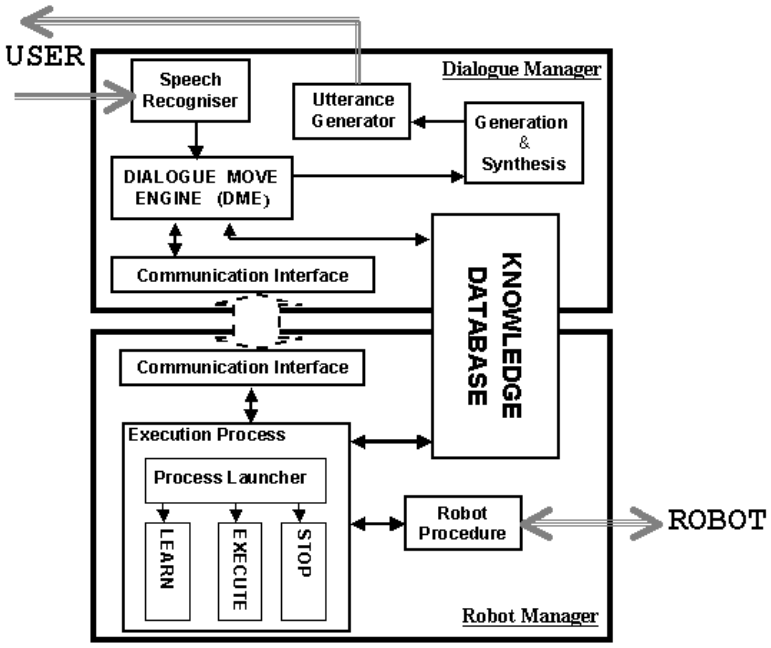


Figure 4. *IBL system's architecture. The Dialogue Manager (DM) is a bi-directional interface between the Robot Manager (RM) and the user, either converting speech input into a semantic representation, or converting requests from the Robot Manager into dialogues with the user. Its components are run as different processes communicating with each other via a blackboard architecture. The RM listens to messages from to the DM while processing previous ones using a multi-threads approach. In the RM module the Communication Interface is the process that launches a message-evaluation thread Execution Process and then resumes listening to the DM. The aim of the Execution Process thread is to understand the message received from the DM and act accordingly (see section 5).*

4.1 Speech Recognition

The task of speech recognition is to map acoustic signals into symbolic representations. In general, the output of the speech recognizer is a word lattice (covering all possible recognized patterns), but in its simplest form, it is a list of words associated with a confidence score. Given the limited domain we work in (and a lexicon size well below 1000 words), we believe that off-the-shelf speech recognition tools offer us enough power to perform this task, even when speaker independent requirements are taken into account. The prototype currently being developed uses the Nuance speech tools for speech recognition. Nuance's speech technology allows one to specify a speech recognition package on the basis of their grammar specification language (GSL). Within Nuance's technology of speech recognition, the GSL grammar is both used for language modeling and parsing. This contrasts with the traditional approaches, where the output of the speech recognizer is typically a word lattice, which is fed to the parser to filter out non-grammatical interpretations and to produce a meaning representation of the instruction uttered by the user.

There are various difficult issues involved in successfully integrating a speech recognition component into a working prototype. Some of them have practical importance, ranging from speaking in noisy environments, speaking in situations with different acoustic qualities, varying distances between the speaker and the robot's microphone, or knowing when the user is talking to the robot and when not. Others are more theoretical of character, and involve the process of designing a GSL grammar, a design we believe should be primarily driven by linguistic knowledge.

4.2 Linguistic Analysis

GSL grammars are mostly hand-built by the user to work for specific applications, although we use a rather different approach by compiling the GSL grammars from linguistically motivated unification grammars. The unification grammar used in the prototype encodes linguistic knowledge based on the corpus analysis and is tuned for the domain of talking to mobile robots. The grammar rules support both features for syntactic constraints as well as features for producing logical forms.

Compiling the unification grammar to GSL format involves eliminating left-recursive rules within the grammar, as well as replacing features and their possible values for syntactic category symbols, as GSL neither support left-recursive rules nor a feature-value system. As a consequence, the language models generated for speech recognition are enormous in size (compared to the original unification grammar), but still feasible for small lexicons (a few hundred words in the case of IBL), and, more importantly, linguistically motivated. Each word in the lexicon is given a semantic representation, and the semantic operations associated with the syntactic structure are compiled-out into GSL as well. As a pleasant side-effect, speech recognition and semantic construction are integrated into one component. In other words, we short-cut the parsing and compositional semantics into a single component. Hence, the output of the speech recognizer will directly be a logical form, rather than a list of words.

4.3 Ambiguity Resolution

Used in isolation, natural language can be highly ambiguous in its meaning. To deal with ambiguities occurring in the user's utterances, we encode the logical forms as Underspecified Discourse Representations Structures, motivated by Discourse Representation Theory (Kamp & Reyle 1993) to cover a range of context-sensitive expressions such as pronouns and presupposition triggers. Typically, the entire dialogue between user and robot is captured by a Discourse Representation Structure (DRS henceforth). So we need to embed the meaning representation of a newly uttered utterance into the DRS of the context so far and resolve any ambiguities with respect to that context. Phenomena that come into question are lexical ambiguities, structural ambiguities, and referential ambiguities (for instance pronouns).

The DRS for the dialogue of Table 1 is shown in Figure 5 (where imperative expressions such as "go to the post office" are marked by the delta-operator). DRSs can be viewed as small models of the world, where a set of discourse referents (standing for objects in the model of the world) are asserted and properties (or relations) are assigned to these referents. Graphically, this is done by placing the discourse referents in the upper part of the DRS, and the properties and relations between them in the lower part of the DRS. Moreover, as Figure 5 makes clear, DRSs are recursive structures, so DRSs can appear as subcomponents of other DRSs. Discourse referents signify objects for possible later reference, for instance by use of a pronoun. However, the internal structure of DRSs constrains pronoun resolution, which is realized by a relation of accessibility. Accessibility is governed by the way DRSs are nested into each other, and hence narrows down the choice of an antecedent in the process of pronoun resolution. Figure 6 demonstrates this internal structure.

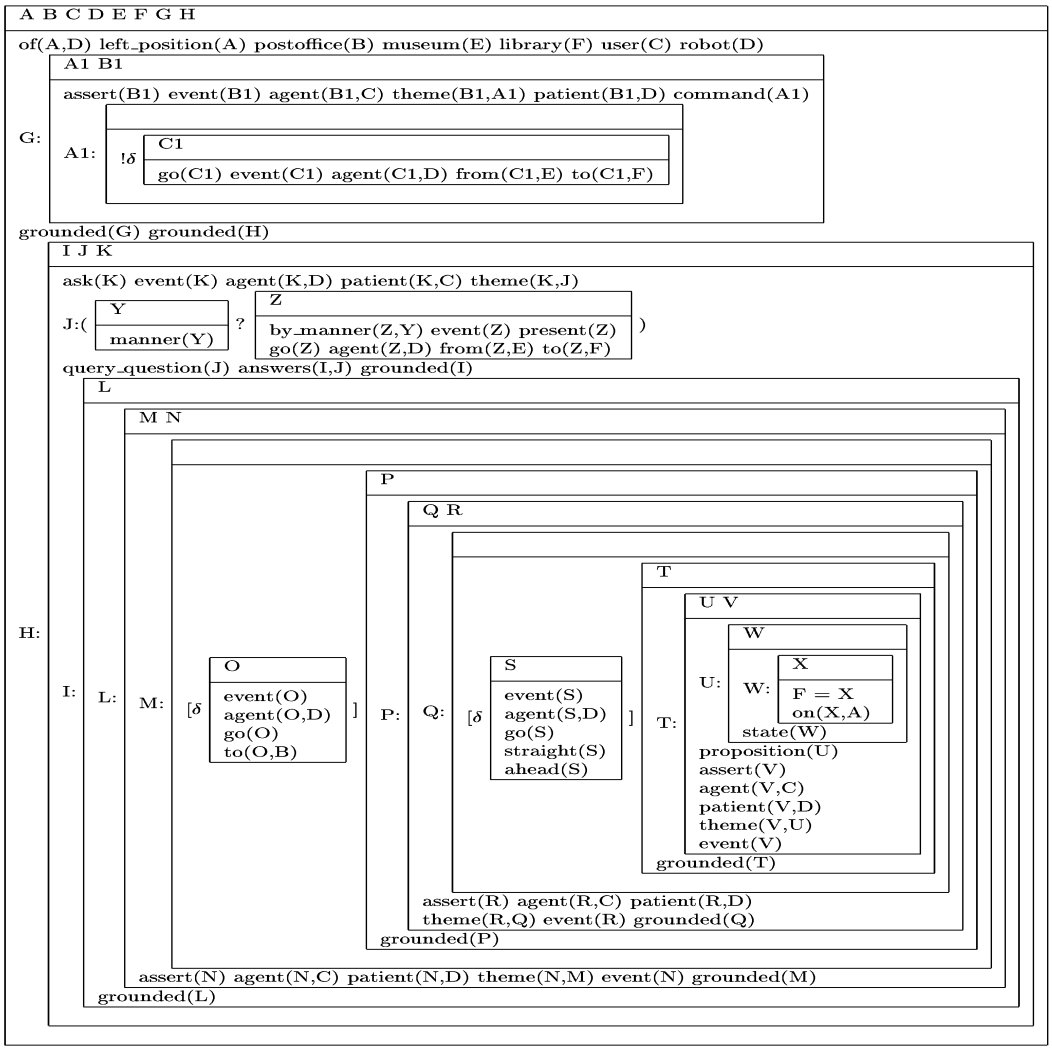


Figure 5. *The Discourse Representation Structure (DRS) produced by the DM for the dialogue example in table 2.*

4.4 Dialogue Updates

A dialogue is viewed as a series of alternative "moves" between the user and robot. Common moves are assertions, questions, or instructions, but moves also include acknowledgements, acceptances or denials. Moves are explicitly represented as part of the DRS (see Figure 5). The dialogue move engine coordinates the integration of new moves (from the user as well as the robot) and decides the next move (for the robot), following the recent information state approach to dialogue processing (Traum et al. 1999). Utterances of the robot are realized by generating prosodically annotated strings from the DRS and feeding these to a speech synthesizer.

Using DRSs for modeling the dialogue has an obvious advantage: they allow us to make inferences, as there is a standard translation available from DRSs to first-order logic (and for first-order logic, there are various theorem provers available (cf. Blackburn et al. 1999)). Hence, this

feature provides us with the means to let the robot make rather "intelligent" responses. Inferences are not only required to resolve ambiguities present in the user's input (being of scopal, referential, or lexical nature), but also to detect the move associated to a new utterance (for example, did the user answer a question or has a new issue been raised?), to plan the next utterance or action, and to generate natural sounding utterances (by distinguishing old from new information within an utterance).

5. The Design of the Robot Manager

The Robot Manager (RM) analyses the DRSs communicated by the DM and controls the robot. As indicated in figure 4, the Execution Process thread interprets the message received from the DM and initiates the relevant sub-processes. In particular, if the user utterance is an executable command (i.e. there is a corresponding procedure) the "Execute" process is started. Otherwise, if it is an unknown execution command the "Learn" process is started and an interaction with the DM starts to resolve this impasse. It is planned that the user will also be able to issue "Stop" command to suspend temporarily or interrupt permanently any other process. The Robot Manager is written using the scripting language Python[3] and C. An important feature of scripting languages is their ability to write their own code. For instance, a route instruction given by the user will be saved by the Robot Manager as a Python script, which then becomes part of the procedure set available to the robot for execution or for future learning.

During learning, the updated DRS received from the DM is searched by the Process Launcher for user commands (indicated by !δ) or explanations. In figure 5, information associated with an explanation is represented either as a box included in square brackets and tagged with a δ symbol (i.e. [δ DRS]) or as a pair made by the predicate *state*(X) and the DRS preceded by the symbol X (i.e. *state*(X) and X: DRS). For example, the box containing the predicates *event(O), agent(O,D), go(O), to(O,B), tense(O)*, is an instruction which is part of the explanation of how to go to the library. Once such a box has been found, the action (i.e. 'go') and the necessary attributes (i.e. 'to' for the action 'go') are detected, using action formats stored in the Knowledge Database. For instance, the primitive action GO, as defined using the corpus and showed in table 1, matches the user instruction and requires the attribute *to*, which is also found in the DRS received from the DM. In this way a first list is generated that contains the actions and their attributes given by the user. Then, the Process Launcher checks whether the necessary parameters for the requested attributes of an action have been given by the user (i.e. 'post office' for the attribute 'to' associated with the action 'go'). Moreover, the system can infer the starting point (*from(C1,E)* and *museum(E)*) from the context. Finally the procedure name for the action is generated (`go_museum_postoffice`). It then checks if a procedure matching that name is found in the database. If that is the case, the procedure is added to the program that defines the new procedure being learnt. If the procedure is not in the database, or the list of attributes is incomplete (for example, the user action is 'turn' but the direction has not been specified), the system will start an interaction with the DM to solve the impasse while the "Learning" thread is put on hold. Table 3 shows the procedure `go_museum_library` after the system has terminated the learning process. According to the user instructions, the first procedure to be called is `go_museum_postoffice.action,` which allows the robot to reach the post office. When the next procedure, (`is_located.action('library','left_position')`) is called, the system follows the remaining user instructions to reach the library. Note that the instructions 'the library is located on your left' and the 'go straight ahead' are represented as the boxed T and S in figure 5. The RM has combined them extracting the correct procedure primitive that performs that part of the instruction.

[3] http://www.python.org

```
Import go_museum_postoffice,is_located
 .
 .
 def action():
        go_museum_postoffice.action ()                  # user defined preocedure
        is_located.action('library','left_position')    # procedure primitive
        return()
```

Table 3. *Python procedure created from the DRS in figure 5 for the route from the museum to the library explained in table 1.*

The next time when the user asks the robot to execute a command (for example ‘go to the library') that can be associated with an existing procedure (i.e. `go_museum_library` now present in the database), the Process Launcher successfully calls the Python procedure through the "Execute" thread.

So far, the full conversion from NL utterances into procedures has been tested with primitives executing pre-programmed robot displacements instead of the eventual vision-based navigation procedures. The later are currently under development.

6. Practical Implications

Teaching a route to a robot using natural language is an application of a more general instruction-based learning methodology. The corpus-based approach described here aims at providing users with the possibility of using unconstrained speech, whilst creating an efficient natural language processing system using a restricted lexicon. It is found that the functional vocabulary is small, containing only 12 primitives, but is not closed. Hence, at some point in the robot's life, the user will have to teach it new primitives (e.g. "cross the road") or reformulate its instructions. To enable the learning of new primitives, the robot should posses an additional set of primitives. These would enable a user to refer to lower level robot actions, e.g. a number of wheel turns, in its instructions. Learning of primitives has been explored in (Seabra Lopes, 2000). With our approach, a new corpus collection process would be required to determine the necessary additional primitive procedures. Another solution may lie in an appropriate dialogue management to suggest a reformulation of the instruction. It is expected that with the corpus-based method used here, the frequency of such "repair dialogues" will be minimised. An open question is the detection of new functions in the user's utterance, as the lexicon may not contain the required vocabulary.

The overall approach to robot control described here may be seen as an attempt to integrate the good properties of Behaviour-based control (Brooks, 1991) and classical AI. Behaviour-based control is an effective method for designing low-level primitives that can cope with real-world uncertainties, and AI has developed effective tools for symbol manipulation and reasoning. However, the system differs in several ways from both methods. Here, the corpus defines what symbols and primitives to use. Consequently, some of the primitives are rather complex functions, involving representations of the environment and planning. These are not always compatible with the representation-less philosophy of behaviour-based systems. On the AI side, the system does not use the full range of reasoning capabilities offered by systems such as SOAR. Here there are no other aims in symbolic processing than verifying the consistency of instructions, and the construction of new procedure specifications. In particular, planning at the symbolic level is not needed at this stage of the project. Instead, planning is performed by the user, and the resulting plan is communicated to the robot using natural language. This limits the autonomy of the robot, but also improves the safety of its use, as unpredictable behaviour is limited.

Other hybrid architectures integrating Behaviour-based systems and AI have been investigated as possible solutions to the symbol-grounding problem (Malcom 1995, MacDorman 1999, Tani 1996, Asoh et al., 1997). This problem is one of maintaining the coherence between representations used to reflect upon actions and events, and the stream of sensory information

produced by a dynamic environment (for a more detailed discussion about symbol grounding problem see for example Harnad, 1990). It is generally accepted that this problem can be avoided if the reasoning process itself depends in some way on its relation to the world, or, in other words, if the development of the internal categories and their transformations depends on external interactions. Accordingly, truly sentient robots should have learning abilities such that dynamically changing external events and results of own actions are allowed to constrain abstract reasoning.

Several works have addressed this issue. The system developed by (Malcom, 1995) operated in a relatively well-ordered and predictable world in which complex tasks had to be achieved. But since the symbol system could only operate under internal syntactic constraints the grounding problem was not really addressed. In the system suggested by (MacDorman, 1996; Tani, 1996) the development of the internal categories and their transformations depended on external interactions. However, there was no human interaction and the grounding could not be modified by a non-experienced user. Whereas in (Asoh et al. 1997), the system could learn new actions through natural language dialogues but only while the robot was performing them (i.e. it could only learn a new route from A to B while it was actually moving from A to B and dialoguing with the user).

In the IBL system described here, learning operates purely at the symbolic level, hence it can be done prior to performance. The ability to predict future states, using SAS representations, enables to engage in a verification dialogue before execution errors occur. If environmental conditions change such that an instruction is not valid anymore, this can be detected from the mismatch between the expected result and the actual one. Learning however is not autonomous. The system requires interaction with a human user to learn new symbols and their meaning. This simplifies the design of the robot due to the transfer of part of the cognitive load to the user. Future experiments will reveal whether this approach results in effective and socially acceptable helper robots, and if the proposed method has a potential for generalization to other instruction contexts.

Acknowledgement: This work is supported by EPSRC grants GR/M90023 and GR/M90160. The authors are grateful to Angelo Cangelosi and Kenny Coventry for enlightening discussions.

References:

H.Asoh, S.Hayamizu, H.Isao, Y.Motomura, S.Akaho, and T.Matsui (1997) Socially embedded learning of the office-conversant mobile robot, Jijo-2, In Proceedings of 15th International Joint Conference on Artificial Intelligence (IJCAI'97), pp.880-885, 1997.

Blackburn P., Bos J., Kohlhase M. and de Nivelle H. (1999). Inference and Computational Semantics. In: Third International Workshop on Computational Semantics (IWCS-3), Tilburg, The Netherlands.

Bischoff, R. Jain T. (1999). Natural Communication and Interaction with Humanoid Robots. Second International Symposium on Humanoid Robots. Tokyo, Japan, October 1999, pp. 121-128.

Brooks, R. A. (1991) Intelligence without representation, Artificial Intelligence, 47, 139-159

Bugmann G., Lauria S., Kyriacou T., Klein E., Bos J. and Coventry K. (2001) "Using Verbal Instruction for Route Learning", Proc. of 3rd British Conference on Autonomous Mobile Robots ande Autonomous Systems: Towards Intelligent Mobile Robots (TIMR'2001), Manchester, 5 April.

Cangelosi A., Harnad S. (2001) The adaptive advantage of symbolic theft over sensorimotor toil: Grounding language in perceptual categories. Evolution Communication. (in press)

Crangle C. and Suppes P. (1994) Language and Learning for Robots, CSLI Lecture notes No. 41, Centre for the Study of Language and Communication, Stanford, CA.

Denis M. (1997) "The description of routes: A cognitive approach to the production of spatial discourse", Current Psychology of Cognition, 16:4, pp.409-458.

Harnad, S. (1990) The Symbol Grounding Problem. Physica D 42: 335-346.

Huffman S.B. and Laird J.E. (1995) "Flexibly Instructable Agents", Journal of Artificial Intelligence Research, 3, pp. 271-324.

Kamp H. and Reyle U.(1993): From Discourse to Logic. Kluwer.

Laird J.E., Newell A. and Rosenbloom P.S. (1987) "Soar: An architecture for general Intelligence" Artificial Intelligence, 33:1, pp.1-64.

Malcom C. M. (1995), The SOMASS system: a hybrid symbolic and behavioured-based system to plan and execute assemblies by robot. In J. Hallam, et al. (Eds), Hybrid problems and Hybrid solutions pp 157-168. Oxford: ISO-press.

MacDorman, K. F. (1999). Grounding symbols through sensorimotor integration. Journal of the Robotics Society of Japan, 17(1), 20-24.

Roy D., 'Learning words from signs and sounds: a computational model', Ph.D. Thesis, MIT Media Laboratory, 1999.

Seabra Lopes, L. and A.J.S. Teixeira (2000) Human-Robot Interaction through Spoken Language Dialogue, Proceedings IEEE/RSJ International Conf. on Intelligent Robots and Systems, Japan.

Steels, L. The origins of syntax in visually grounded robotic agents. (1998). Artificial Intelligence,103:1-24

Tani J. (1996) Model based learning for mobile robot navigation from the dynamical system perspective 1996 IEEE Trans. Sys. Man Cybernetics, Part B, 26:3, pp 421- 436

Torrance M.C. (1994) Natural communication with robots. MSc Thesis submitted to MIT Dept of Electrical Engineering and Comp Science, January 28, 1994.

Traum, D., J. Bos, R. Cooper, S. Larsson, I.Lewin, C. Matheson and M. Poesio (1999): A model of dialogue moves and information state revision. Trindi Report D2.1. Available from http://www.ling.gu.se/research/projects/trindi