Converting Multi-Modal Task Instructions to Rule-Based Robot Instructions

Joerg C. Wolf and Guido Bugmann

Abstract—**While frame-based representation of knowledge is a well known concept [1,2], its application to verbal rule instructions poses a number of problems. This paper describes how verbal instructions can be converted (mapped) into a frame-based representation, to form a base for reasoning and carrying out robot actions. Furthermore the paper introduces idea of corpus-based robotics, which supports the design of natural human-robot communication systems. An application and experimental results to the scenario of teaching a card-game are shown.**

I. Introduction

NATURAL communication between people using speech and gesture is part of our daily live and we are oblivious to how complicated the mechanisms are that are involved. If a person would like to communicate in the same way to a robot, with the same speed and level of language natural communication to robots becomes a hard problem to solve. In this paper we hope to present some solutions which can bring us closer to solving this problem.

A particularly useful scenario for future human-robot communication is teaching a task that the robot should be able to perform afterwards. The main advantage of *natural* task instructions is that it removes the need for the users training and the need for thick manuals, making service robots programmable by everyone with normal human communication ability.

Related work in the human-robot teaching scenario has been carried out by [3,4,5]. In these works the emphasis is on creating natural or near to natural human-robot teaching interfaces for service robots in a realistic scenario. The user of the interface treats the robot as a competent learner that can understand the task after a single demonstration.

A. Corpus-Based Robotics

The idea of Corpus-Based Robotics is borrowed from Corpus Linguistics. In Corpus Linguistics text is collected into a database called a "corpus". These texts can also consist of transcribed spoken dialogues. The strength of Corpus Linguistics is that the actual use of language can be investigated as opposed to the traditional study of language structure [18]. Similarly Corpus-Based Robotics also uses a corpus to determine the language and gestures used when interaction between a human and a robot takes place.

The point of departure for creating such natural communication with robots is corpus-based robotics (CBR). In corpus-based robotics, human-to-human instructions are recorded and used as an information source and guideline for the design of the robot. The recorded corpus can be multi-modal and contain speech and gestures in form of sensor data. This speech and data is then mapped to *primitive functions* that the robot can carry out. This paper describes in section IV the mapping of speech and gesture to primitive functions and from there into a frame-based reasoning system.

These primitive functions are high-level functions referred to in the human language, not action primitives devised by robot designers. We have found that a clause in an utterance usually contain one primitive, derived from the main verb. The noun-phrases contain parameters of the primitive. These primitives are often complex procedures in the robots lower level structure. In order to successfully perform the primitive, the implementation requires appropriate algorithms and hardware. Ultimately, the corpus tells the robot designer what is required. This is therefore a corpus-based design process.

1) Robot Grammars, a logical consequence?

Linguistics is divided into corpus linguistics and structural linguistics (in "structural linguistics", sentences are built from a grammar). The same division could be hypothesized in robotics, where corpus based robots is opposed to structural robotics. In structural robotics, the robot is build from components. For example, a part of a robot-grammar of structural robotics could be:

```
robot ->  sensors processing_unit actuators
actuators ->  drive_electronics drive_hardware
drive_hardware ->  wheel gearbox shaft-encoder
```

Whereas in corpus-based robotics, the utterance "drive forward" would create the need for a drive hardware design

Reviewed Manuscript received May 19, 2008.

Joerg. C. Wolf (IEEE Member) is with the Centre for Robotics and Intelligent Systems at the School of Computing, Communications and Electronics, University of Plymouth, Drake Circus, Plymouth, PL4 8AA, United Kingdom (e-mail: joerg.wolf-xaxt-plymouth.ac.uk)

Guido Bugmann is with the Centre for Robotics and Intelligent Systems at the School of Computing, Communications and Electronics, University of Plymouth, Drake Circus, Plymouth, PL4 8AA, United Kingdom (e-mail: gbugmann-xaxt-plymouth.ac.uk)

TABLE I
LINGUISTICS CONCEPTS APPLIED TO ROBOT DESIGN

Symbol	Corpus	Structural
Linguistics	Corpus Based Linguistics (has Corpus of words)	Structural Linguistics (has linguistic Grammar)
Robotics	Corpus Based Robotics (has Corpus of functions)	Structural Robotics (has robot-function grammar)

2) Previous work on Corpus-Based Robotics

In previous work, our research group applied corpus-based robotics to a route instruction scenario (the Instruction Based Learning project (IBL) [6]). In the IBL project the robot was able to navigate its way through a model town, after a human instructor explained the path. The set up for the IBL project did not consider gestures. The primitive combinations found were purely sequential. Neither conditionals nor loops where found. Therefore a new scenario was needed to include these other components of instructions. Teaching a card game includes actions (and gestures), and conditional primitives (game rules), and is a flexible test bed, since the robot could learn different games. The results are described here.

II. CORPUS COLLECTION

A. Procedure

Corpus collection has been described in detail in [7]. We summarize the main points here. We have collected a corpus of 21 instruction dialogues between a human teacher and a human student. The teacher explained the card game Scopa. An initial instruction session was followed by one or two games during which the instructions were refined. We report here on the initial instruction phase. Subjects did not know the selected Italian card game, but had prior knowledge of card games. The setup is shown in Figure 1. The subjects explained the game to each other in a long chain, whereby the last student becomes the teacher for the next student.

Fig. 1: Corpus collection setup. The instructor on the right moves a card on the touch screen. The learner sees a copy of the move on her

B. Example Conversation

TABLE II
EXAMPLE DIALOGUE (SESSION 11 FROM MIBL CORPUS)

Type	Time in 10th sec.	utterance text or gesture semantics
TU	1588-1619	"and er this is how the game goes um"
TU	1622-1686	"what you have to do is er if you can you take one card from your er three cards"
TU	1688-1742	"and you have to either like Im doing here you either match it with a card on here"
TG	1739-1756	move(D/07,+table+table)
TG	1715-1735	move(C/07,+hand1,+temp1)
SU	1740-1747	"ok"
TU	1914-1975	"so in my case what I have done there is I have put the seven in and therefore I have won that seven"
TG	1931-2000	move(C/07,+hand1,+temp1) move(C/07,+temp1,+temp1) move(D/07,+table,+emp1)
TU	1976-2024	"which means that I have won that er"
TG	2007-2007	turn (C/07",up)
TU	2027-2060	"I have won those cards there and now you will down to the three and then its your go"
TG	2018-2045	move(C/07",temp1",side1") move(D/07",temp1",side1")

The table shows a typical example of instructions found in the corpus. T= human teacher, S= human student, U=verbal utterance, G=gesture / action

C. Language Primitives

Analyzing such instructions leads to the definition of following semantic classes that will need to be identified by the speech recognizer (Table III). We are using Nuance(TM) which supports semantic grammars mapping directly from speech to any desired output using slot filling.

Each utterance of the corpus is divided into linguistic clauses. For each clause a grammar rule was created, mapping it to a primitive. For more information on this procedure see [8]. The primitives are extracted manually by carefully looking through the corpus and taking notes until a final format of the primitives has been found.

We may note several features of the instructions:

- They contain procedural instructions "first you deal"
- They contain declarative factual instructions "a king is worth X"
- They contain declarative rule instructions "that is how you can capture a card". These are often in the form of sometime hypothetical examples that the student needs to generalize.
- They contain imperative commands "you do this until there are no cards left" These can also be interpreted as a statement of the goal of the game.

TABLE III
IDENTIFIED LANGUAGE PRIMITIVES

Type	D/ P	Language Primitve
fact	D	value(cardname, value)
fact	D	exist(cardname) / not_exist (cardname)
conditional	D	ifcond(how, what, lhs, rhs, rhs,…)
conditional	D	ifloc(cardname, location)
conditional	P	until(...)
context	D	new_case()
context	D	type(imaginary / real)
action	P	move(cardname, amount, from, to)
action	P	turn(cardname)

Table listing the primitive functions found in the MIBL corpus. D=Declarative primitive, P=Procedural primitive

Human instructions are constructed for a listener with human reasoning capabilities. Indeed, the instructions found in the corpus reflect the assumption that human learners have reasoning capabilities and prior knowledge. For instance, the instructions contain no instruction on how to use the declarative information (marked with D in table III). The teacher assumes that the card-game experienced student has the capability to reason using the acquired knowledge in order to decide the next move during the game.

This means a useful future service robot should have an adult-like (experienced) prior knowledge of the domain, for maximum efficiency during instruction receiving. The key is that adult-like prior knowledge is achievable in a specific domain, but so far not in "general" terms. In addition, a robot able to use such instructions would also needs human reasoning capabilities. However, in a particular task domain, represented by a given corpus of instructions, a robot does not necessarily need to emulate all forms of human reasoning. Current models of human reasoning also show such task specificity. An investigation into the required computational approach is required to determine the right framework.

The problem in designing a learning robot is the selection of a suitable representation of knowledge and of operations that can be performed on that representation.

III. COMPUTATIONAL APPROACH

A. Issues

In order to organise knowledge from natural language, Minsky [1] and Schank and Abelson have presented pioneering work in their book "Scripts Plans Goals and Understanding" [2]. In this case very domain specific frames and scripts where holding the information about a story.

Models designed to reproduce human problem solving in the domain of logic games (chess, mathematical puzzles) all use some form of production rules, that specify the consequence of an action performed on a given initial state. These are defined in discrete state spaces suited for the domain. Through the use of an inference engine, the required transformation steps required to achieve a goal state can be determined. These models include the General Problem Solver (GPS from Newel and Simon [9]), SOAR [10] and ACT-R [11].

For modelling time-constrained decision-making processes (e.g. of nurses, rescue workers or military commanders) computational models of recognition-primed decision making were developed [12, 13]. These proceed by selecting initial actions from a library of solutions applied in similar situations. Then, by forward-chaining they evaluate if the initial step can lead to the desired goal.

When modelling how students are able to solve a new problem by analogy with a known example, models of analogical problem solving are used [14]. Such models use some form of spreading activation between problem representations.

The declarative form of some of the game instructions (marked with D in table III) requires the use of a problem solver of some form. As game instructions essentially define actions in a discrete state space, problem solvers developed for mathematical puzzles are good candidates. Early work on robot instruction was based on SOAR. The Instructo-SOAR system [15] was able to handle the types of instructions listed in table III. It was however restricted to typed text input and could not handle rules explained over multiple utterances.

In order to store knowledge that can be generalised easily, we also investigated ontological reasoning. [16]. By using `is-a` and `has-a` and instantiation relations, a hierarchical representation of the robots world can be created. See [8].

For storing and reasoning with the card-game instructions from the teacher, we decided to store all knowledge gained from the instructions into frames, which are part of ontology. (see rule-frames in the next chapter) This well structured knowledge then provides the base for creating production rules for a problem solver attempting to plan the task that the teacher explained.

The presented implementation is done in logic programming (Prolog). The main advantage is that programming consists of stating the problem in a declarative form (like many primitives) and Prolog will seek the solution.

IV. MAPPING

A. Rule frames and States

The content of rule frames is flexible, determined by the content of instructions. It must include at least a conditional to make a rule only apply in a given situation. This is usually followed by sequential instructions that have to be carried out in the situation. Since rule are context-dependent, a new rule frame is created every time the context changes. This is achieved by appropriately using mapping rules in the speech recognition grammar. For instance:

```
((and ?er this is how the game goes ?um)
        {return(":context=new_case:")}
```

This new value of the variable context validates a Prolog rule (not described in detail here) that creates a pointer to a new frame. After a new rule frame is created in that way, every consecutive utterance will be added to it. Initially users usually start explaining in which situation the rule applies. For instance:

```
(?(what you have to do is ?er) if you can you
take one card from your ?er three cards)
{return(":ifloc=ns-cardname-ns,01,+hand2:")}
```

The *ifloc* condition for example means that in a situation a card must be in a specific location initially. We found that *ifloc* conditions are sometimes implicit and can only be recovered from demonstrations rather than from language. The *ifcond* condition is a more general if-statement that compares properties of these cards found in specific locations. *ifcond* takes a minimum of four parameter, namely how to compare , what to compare and the cards involved. The reference card is mentioned first, the other cards follow. An example of a rule frame is shown in figure 2.

```
(and you have to either like i am doing here you
either match it with a card on here )
{return("ifcond=match,?,deictic_determinative,de
ictic_determinative:")}
```

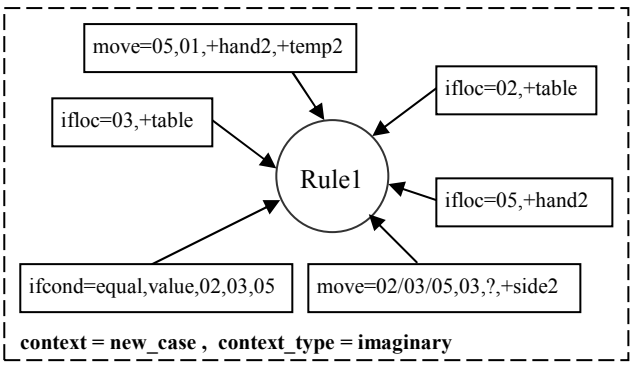

Fig 2: Rule Frame, a collection of semantics that are connected to Rule1 since the utterances were in the same context. This rule frame is used to construct a function that the robot can carry out to play the game/ perform a task. If there are question-marks left, or ambiguities, the robot can clarify information with the teacher before creating the new robot function.

B. *Mapping issues*

Semantic analysis of the corpus reveals the previously described language primitives (table III). These language primitives are connected to the knowledge base by a mapping process. Different types of language primitives affect different areas of the knowledge base. Simple facts directly change object properties in the robots world model.

We found that most rule instructions are a combination of several utterances; hence a framework is required to show the relationship between them. For example, a rule that describes how to capture a card in the game consists of conditionals describing the situation when it is allowed to capture a card and then continue to describe how to move the cards by using sequential action primitives. These combinations are stored in a framework called rule frames.

A context change will open up a new rule frame. And subsequent instructions are stored into this new rule frame until another context change occurs. This mechanism allows mapping instructions that belong together into the same rule.

Underspecification in the parameters of language primitives result in question marks placed in rule frame slots. For example the utterance `"and then we put four cards in the middle"` generates the following language primitive: `move=ns-cardname-ns,04,?,+table:` which has a question mark in the source location, since it is unknown where the card comes from. It the question mark is not resolved at the end of the explanation the robot will ask the user for clarification. Normally, resolving question marks is done by unifying information from gestures and other utterances of the same rule explanation. The unification process takes two or more language and gesture primitives of the same kind and tries to combine their parameters. Our multi-modal fusion system performs this unification in real time. The unification algorithm is described in [19]. A further unification system is applied at the end of a rule explanation.

The rule frames are wrapped into state-transition rules (STR) to allow the robot to predict the outcome of its actions. A state transition rule consists of the entry state, a rule frame and the exit state.

C. *Mapping Process overview*

The mapping of corpus utterances to robot functions is divided into several steps. A summary of the process:

1. Utterances are mapped to primitives and parameters using grammar
2. Primitives are mapped to rule-frame instructions taking into account unification with gestures and reference resolution.
3. Production rules are created that can create plans from rule-frame instructions. This enables planning in the robots brain. The robot plans consequences of its actions.
4. Each production rule may have a implementation of an actual robot action (i.e. move gripper) attached.

D. *Anaphora Resolution*

Anaphora are references to explicitly mentioned nouns, earlier in the discourse [17]. The determinative demonstrative deictic references (DDD): *this, these, that, those* and *the* in the noun-phrase are indicators for anaphora. Further corpus references are determinative possessive deictic references (DPD): *my, your, our, his, her, its, their, ones*. And finally the word *them* is also treated as a reference to earlier mention. A grammar has been defined to forward the reference category to the unification process. Here an anaphora resolution algorithm tries to identify the reference by looking at the previous utterance. If a previous utterance and its corresponding rule-frame were identified, it is possible to recover missing information for the new rule-frame. For example the utterance “turn *them* over”, does not

say which cards need to be turned over, how many or where they are.

Every noun phrase is stored in the rule frame (knowledge base). If the resolution algorithm is confronted with a DDD, the resolution is achieved by retrieving the previously stored noun phrase.

E. Generalisation

We found from the corpus that complex tasks are often explained using an example, which means that the robot needs a generalization mechanism.

A personal robot has to be able to learn from one or two examples of a task explanation. Asking for further explanations will annoy the user, since an experienced personalised service robot is not seen as a child in the user's eyes. It is an adult servant who should reduce the workload of the user. Furthermore, anyone who has used speech recognition knows that it can test the user's patience. Therefore the robot must go to great length in order to generalise what it learned autonomously. It is possible to rely on so called Hasty Induction of the rule without proves. If a user says "if you have a five in your hand" , Prolog will treat this five as a placeholder in its representation of the instruction. This implicitly implements the concept of generalization. Practically, this is equivalent to storing "if you have card *A* in your hand". This may be how humans learn game rules, and they are often stopped later on by the teacher if the generalization was flawed. However, the limited task domain is an advantage that helps making correct generalizations. While explaining the capture and the pairing rule of the game, all subjects used actual cards as examples or at least gave examples set in an imaginary situation.

F. Problem Solver

In order to use rules that have been explained to the robot, the robot needs to constantly compare the current state of the environment with the precondition in the rule frames in order to derive the next valid step in its actions. This is realised using a problem solver. Once the next valid step has been found the robot can predict the consequences using the state transition rules. The robot simulates the next step by using the state transition rules (production rules) which consist of rule frames.

When trying to apply a rule, i.e. going from state to state, an algorithm is used to put the rule frame into action.

1. check all *ifloc* conditions of the rule-frame
2. considering the cards selected in the *ifloc* conditions, do the *ifcond*, which usually is a comparison function between cards.
3. do all action instructions of the rule-frame

Firstly, the current state is examined if the necessary cards are in place (*ifloc*). This first application of the *ifloc* rules also creates a tuple-list of cards and their properties that are involved in the rule. A Tuple-list is necessary to preserve the reference to objects within a rule throughout the application of the primitives. For example "if you have say a five" "you can bring the five forward" This is a fact and an action, but the "five" is mentioned two times. This link must be preserved in the reasoning when applying the rule. The next step is to apply comparison functions to the tuple-list and the current state. If the comparison function is passed as successful the last step is carried out, which is doing the actions of the rule in sequence. First the actions are only simulated by modifying the current state and replying this outcome to the problem solver. If the problem solver selects this state the actions are actually applied by the robot. Within a rule-frame, the problem solver tries to carry out the functions in order of explanation. If this fails, the problem solver can try actions in a different order, since some human teachers fail to explain the rule in the right order to the robot.

V. Experiments with Corpus

A. Corpus playback of dealing rule and Card-pairing

In order to confirm that the system described in earlier chapters performs correctly (learning a rule as the teacher intended), the data of the corpus can be played back to the robot.

The teacher's voice and touch-screen data is fed into the robot (software agent). In two experiments, the explanations of dealing of cards and the explanation of the pairing-rule is played back. The robot has a chance to ask questions during this learning phase. After a single rule explanation the robot is instructed to play. At this point the robot will start its problem solver to apply the learned rule. A printout of the rule-frames quickly reveals any problems during development. The corpus has been split into half named test-set and evaluation-set.

In the first experiment 10 dialogues (of our test-set) where played back to the robot, and the robot successfully learned the dealing rule in the way the teacher explained it.

In the second experiment we investigated a total of 19 dialogues (test and evaluation set) from the corpus which explains how to capture cards by pairing them together.

In 3 cases the explanation of the pairing rule was by the teacher was so incomplete that the robot did not know what to do. The robot successfully learned and applied pairing rule in all 16 remaining teaching dialogues.

The explanations of the same rule can result in a different set of instructions, since every teacher has his individual understanding of the rule. Some teachers would show the card from the hand first before capturing for example. Others may define the winning pile in a different place.

In most cases the teacher did not explain the pairing rule completely when comparing to the original rules of the game. However the robot was able to learn and execute the rule in the way the teacher explained it.

B. Discussion

These tests with the corpus do not show much on general the framework is. However they confirm that the instructions

found in the corpus have been successfully implemented into the robotic software agent. They also show if a combination of language primitives lead to correct reasoning in the robot.

The nature of spoken language makes the determination of language primitives and a reasoning system particularly difficult. Users often say incomplete statements or change their mind in the middle of an utterance. An extensive practical evaluation of this system should reveal its potential and limitations. Currently we are testing the system "live" with people rather than from the corpus. This will hopefully prove the robustness of the MIBL system in a final evaluation.

Due to the limitation in the anaphora resolution system and the limited understanding of the dialogue state, the robot sometimes fails to resolve all parameters that the human student was able to resolve. Which is not a serious problem, since the robot can resolve these missing parameters by interrogating the teacher. Other limitations, are the detection of a repletion of an instruction. Teachers sometimes repeat an utterance, even if they actually want the robot to do the action only once, not twice.

The use of Prolog to implement a learning system is convenient as it includes the necessary inference engine. However, it is also time consuming to program each primitive manually, and cannot be considered as a general purpose tool for roboticists working on HRI. For this, it will be necessary to develop automatic code generation tools that support primitive mapping and implementation.

We are proposing to take corpus-based robotics to the next level by providing convenient developer tools, such as MuTra [7].

C. *Future experiments*

Future experiments are planned whereby subjects will be invited to instruct the robot verbally. If speech recognition fails, subjects often simplify their sentence. We hope to catch these simplified utterances and add these to our corpus.

In the long term, experiments on other more complex domains (other than card games and route instructions) will push the boundaries of the framework.

VI. CONCLUSION

This paper demonstrated that natural task instructions can be converted to human level primitive functions (semantics). Due to the nature of task instructions, which go over several utterances, we demonstrated that rule-frames are able to keep the relevant information of the primitive functions. Furthermore the robot was able to use the instructions from the rule frame by applying a problem solver to play cards. The design concept for creating the card-game learning robot was starting from a corpus of dialogues via primitive extraction and design to implementation.

We have shown in the IBL project that the corpus-based robotics approach works [6] and in this project (MIBL) the results look promising. We can therefore assume that this approach may work in other human-robot teaching domains. We are convinced that the corpus-based approach brings us a step closer to design service robots that are programmable by end-users.

REFERENCES

[1] Marvin Minsky, (1974) "A Framework for Representing Knowledge", MIT-AI Laboratory Memo 306, June, 1974.

[2] Schank, R. and Abelson, R., (1977), "Scripts Plans Goals and Understanding", Book published by Lawrence Erlbaum Associates Inc, New Jersey, U.S.A., ISBN 0-470-99033-3

[3] Steffen Knoop, Michael Pardowitz, Rüdiger Dillmann. „Automatic robot programming from learned abstract task knowledge." In Proc. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007).

[4] Anders Green. "Characterising dimensions of use for designing adaptive dialogues for human-robot communication." In Proceedings of IEEE RO-MAN 2007 16th International Symposium on Robot and Human Interactive Communication, August 26-29, Jeju, Korea

[5] S. Hüwel, B. Wrede, and G. Sagerer. (2006). "Robust speech understanding for multi-modal human-robot communication", In Proceedings RO-MAN (pp. 45-50). Hertfordshire.

[6] Lauria S., Kyriacou T. Bugmann G., Bos J and Klein E. 2002, "Converting Natural Language Route Instructions into Robot-Executable Procedures" Proceedings of the 2002 IEEE Int.Workshop on Robot and Human Interactive Communication (RO-MAN 2002), Berlin, Germany, pp. 223-228.

[7] Wolf J.C., Bugmann G. (2005) Multimodal Corpus Collection for the Design of User-Programmable Robots. Proc. Taros'05, London, p. 251-255.

[8] Joerg C. Wolf, Guido Bugmann,(2007) "Understanding Rules in Human-Robot Instructions", presented at the RO-MAN 07: , South Korea, ,Paper No. WA2-4, pg. 714-719 , Sept 2007

[9] Newell, A., and H. A. Simon. 1972. Human problem solving. Englewood Cliffs, NJ: Prentice Hall.

[10] Rosenbloom, Laird, and Newell, (1993) "The Soar Papers: Readings on Integrated Intelligence"

[11] Anderson, J. R. (1996). ACT: A simple theory of complex cognition. American Psychologist, 51, 355-365

[12] Klein G.A. and Calderwood R. (1991) "Decision models: Some lessons from the field", IEEE Trans. on Systems, Man and Cybernetics, 21:5, pp.1018-1026.

[13] Warwick, W., S. McIlwaine, R. Hutton, and P. McDermott. 2001. Developing Computational Models of Recognition-Primed Decision Making. In Proceedings of the Tenth Conference on Computer Generated Forces, May 15-17, Norfolk, VA, pp 323--331.

[14] Holyoak, K. J., & Thagard, P. R. (1989). A computational model of analogical problem solving. In S. Vosniadou & A. Ortony (Eds.), Similarity and Analogical Reasoning London: Cambridge University Press (pp. 242-266).

[15] Huffman, S.B. and Laird, J.E. (1995). Flexibly Instructable Agents. JAIR 3, 271-324.

[16] Smith Barry, "Ontology: An Introduction How to Build an Ontology", Online Lecture, University of Buffalo, Department of Philosophy, 2006,http://ontology.buffalo.edu/smith/ (visited 13/02/2007)

[17] Grishman, R. (1986) "Computational Linguistics, An introduction", Series: Studies in Natural Language Processing, Cambridge University Press, Cambridge, U.K.

[18] Biber Douglas, Conrad Susan and Reppen Randi, (1998) "Corpus Linguistics – Investigating Language Structure and Use", Cambridge University Press, Cambridge, U.K. ISBN 0-521-49957-7

[19] Wolf Joerg C., Bugmann, Guido (2006) "Linking Speech and Gesture in Multimodal Instruction Systems" in the Proceedings of RO-MAN 06: Hatfield, U.K., pg. 141-144