MULTI-MODAL TASK INSTRUCTIONS TO ROBOTS BY NAIVE USERS

by

JOERG CHRISTIAN WOLF

A thesis submitted to the

UNIVERSITY OF PLYMOUTH, U.K.

in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing Communication and Electronics,
Faculty of Technology

2008

PREFACE

This research was carried out at the Robotics and Intelligent Systems lab (RIS) at the University of Plymouth, U.K. The MIBL project was running from 2004–2008 and was funded by a Faculty of Technology scholarship. The project was a partly a continuation of the IBL project, that was running from 2001-2003. The motivation for this project came from the open questions in the IBL project: How can a robot be build that understands not only sequential instructions but also rules? How can a grammar be created that covers a particular domain from a corpus?

Dr. Guido Bugmann was the director of studies of this thesis and I thank him for his advice and invaluable guidance. Also, I would like to thank Dr. Paul Robinson for supervising the work and the financial support for attending conferences. I would show my appreciation to my supervisors for letting me take part in other robotics research projects during my PhD and their guidance in all areas of live.
Furthermore I would like to thank the Faculty of Technology management for entrusting me with the scholarship.

I would like to thank Louise Entwistle for her help with the transcriptions. Furthermore I would like to thank all members of the research lab, who I have exchanged inspiring ideas with.

I also thank my examiners, Dr. Yiannis Demiris, Dr. Tony Belpaeme for their valuable suggestions on my thesis.

Many thanks to my family who have supported me all the way.

Finally I thank my lovely wife for her patience during the days and nights when I was preparing this thesis.

Plymouth, August 2008

Joerg Christian Wolf

AUTHOR'S DECLARATION

At no time during the registration for the degree of Doctor of Philosophy has the author , Joerg Wolf, been registered for any other University award without prior agreement of the Graduate Committee.
This study was financed with the aid of a Faculty of Technology scholarship.

A programme of advanced study was undertaken, which included the extensive reading of literature relevant to the research project and attendance at international conferences on Robotics and Human-Robot Interaction.

The author has published papers in the following peer-reviewed international journal:

1. "Industrial Robot: An International Journal, Special Issue on Service Robots Volume 32 Number 6, ISSN 0143-991X, page 499-504 , Oct 2005

and has presented papers in the following international conferences:

2. IEEE RO-MAN 08: The 17th IEEE International Symposium on Robot and Human Interactive Communication, Munich, Germany , Aug 2008.
3. the Second Workshop on Humanoid Soccer Robots @ 2007 IEEE-RAS International Conference on Humanoid Robots, Pittsburgh (USA), November 29, 2007
4. TAROS 2007: Towards Autonomous Robotic Systems Aberystwyth,U.K. ,p. 176-181, Sept 2007
5. IEEE RO-MAN 07: The 16th IEEE International Symposium on Robot and Human Interactive Communication, Special Session Paper in Human-Robot Interactive Teaching ,Jeju Island, South Korea,Paper No. WA2-4, pg. 714-719 , Sept 2007
6. IEEE RO-MAN 06: The 15th IEEE International Symposium on Robot and Human Interactive Communication, Hatfield, U.K., pg. 141-144, ISBN 1-4244-0564-5
7. European Robotics Symposium (EUROS-06), Palermo,Italy
8. Third International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore(CIRAS) 2005, (ISSN: 0219–6131)
9. TAROS 2005: Towards Autonomous Robotic Systems, (ISBN 0-905247-03-5)
10. 2004 FIRA Robot World Congress (Paper 151)
11. 2004 FIRA Robot World Congress (Paper 158)
12. UK Championships of FIRA Robot Football 2004

and has presented a public lecture:

13. "Robotics @ Plymouth" IEE Devon and Cornwall Branch 2006

and has engaged in learning and teaching at the University of Plymouth in:
Robotics, Control Systems, Digital Electronics, Advanced GUI Programming, Artificial Intelligence and Natural Language Interfaces.

COPYRIGHT

Word count of main body of thesis: 49560

Signed : ..... J. Wolf .....

Date : ..... 2/03/09 .....

PhD EXAMINERS

External: Dr. Yiannis Demiris (Imperial College)

Internal: Dr. Tony Belpaeme (University of Plymouth)

PhD SUPERVISORY TEAM

Director of Studies: Dr. Guido Bugmann (University of Plymouth)

2nd Supervisor: Dr. Paul Robinson (University of Plymouth)

Abstract

This thesis presents a theoretical framework for the design of user-programmable robots. The objective of the work is to investigate multi-modal unconstrained natural instructions given to robots in order to design a learning robot. A corpus-centred approach is used to design an agent that can reason, learn and interact with a human in a natural unconstrained way. The corpus-centred design approach is formalised and developed in detail. It requires the developer to record a human during interaction and analyse the recordings to find instruction primitives. These are then implemented into a robot. The focus of this work has been on how to combine speech and gesture using rules extracted from the analysis of a corpus. A multi-modal integration algorithm is presented, that can use timing and semantics to group, match and unify gesture and language. The algorithm always achieves correct pairings on a corpus and initiates questions to the user in ambiguous cases or missing information. The domain of card games has been investigated, because of its variety of games which are rich in rules and contain sequences. A further focus of the work is on the translation of rule-based instructions. Most multi-modal interfaces to date have only considered sequential instructions. The combination of frame-based reasoning, a knowledge base organised as an ontology and a problem solver engine is used to store these rules. The understanding of rule instructions, which contain conditional and imaginary situations require an agent with complex reasoning capabilities. A test system of the agent implementation is also described. Tests to confirm the implementation by playing back the corpus are presented. Furthermore, deployment test results with the implemented agent and human subjects are presented and discussed. The tests showed that the rate of errors that are due to the sentences not being defined in the grammar does not decrease by an acceptable rate when new grammar is introduced. This was particularly the case for complex verbal rule instructions which have a large variety of being expressed.

Contents

List of Tables

List of Figures

Glossary

AI	Artificial Intelligence. A branch of computer science concerned with creating intelligence in machines.
acoustic packaging	Psychology. Acoustic information, usually in the form of speech helps infants to structure and separate "package" a stream of actions that is being demonstrated to them.
alignment	In Natural Language: use of a dialogue and sentence structure that both dialogue partners understand.
anaphora	Anaphora are references to explicitly mentioned nouns, earlier in the discourse, see Grishman (1986).
AR	Augmented Reality. Usually 3D computer graphics added to a live video feed.
BFO	Basic Formal Ontology. An ontology is a specification of a conceptualization, for example a specification how to network of semantic classes. Basic Formal Ontology is a special form for defining ontologies, see Smith (2006)
CFG	Context Free Grammar. A grammar that consists of a single non-terminal symbol on the left-hand side and terminals/non-terminals on the right hand side.
CG	see Conceptual Graphs.
Conceptual Graphs	Like semantic networks, Conceptual Graphs represent concepts and their relationships, see Sowa (2005)
context tagging	The process of tagging parts of the corpus with a context marker that describes the situation.
corpus	Latin for "body". In linguistics a corpus is a collection of texts, for example transcriptions or newspaper articles.
corpus-based robotics	*Design method* in robotics that allows the design of an artificial agent with natural communication skills and matching user requirements. The design method is based on collecting a corpus of instructions before implementing the agent. See (Bugmann *et al.*, 2004) or Chapter 1.
corpus-based clause grammars	Method of deriving a grammar for natural language interpretation from a corpus, see chapter 6.

DDD	determinative demonstrative deictic references: *this, these, that, those* and *the.*
DPD	determinative possessive deictic references: *my, your, our, his, her, its, their, ones.*
dependent clauses	In grammar, a clause that cannot stand alone as a sentence.
DRS	Discourse Representation Structure. The structure to represent sentences and discourse used in Discourse Representation Theory. The structure is related to predicate logic and every introduced object and referent is assigned an identifier, allowing accurate representation of anaphora..
DRT	Discourse Representation Theory. A theory to express sentences and language discourse in a formal framework. Invented by Hans Kamp, see Kamp (1993).
DTD	Document Type Definition. A DTD file defines XML tags and their syntax for a particular document type.
EPSRC	Engineering and Physical Science Research Council, a British research council
frames	In Artificial Intelligence: Frames combine domain knowledge into a structure for representing and reasoning with stereotypical concepts or situations.
GPS	General Problem Solver, problem solving A.I. program, see Newell and Simon (1972)
HCI	Human Computer Interaction, a discipline concerned with the study of the interaction between humans and computers and the design of user interfaces.
HRI	Human Robot Interaction, a discipline concerned with the study and improvement of the interaction between humans and robots.
IBL	Instruction Based Learning. IBL is the process of learning a task from a teacher through instructions, usually verbal. The learning process may be supported by, but is not depending on a demonstration.
instruction primitive	An instruction on human level, when given verbally, can be expressed using a single main verb.
instruction primitive parameter	Information further specifying an instruction, like parameters of a function in programming

instruction primitive type	Either a Conditional, Context, Action or Fact
instances	In Object-Oriented Languages / Ontology: Instantiations ("as copies") of classes which serve as the template
KB	In Artificial Intelligence: Knowledge Base, a database that stores knowledge in an organised format. If machine-readable, deductive reasoning can be applied to the knowledge base by applying algorithms.
Lambda Calculus	In Computer Science: The Lambda Calculus is a formal system designed to investigate function definition, function application and recursion
LISP	A high level programming language, popular in artificial intelligence.
GSL	Grammar Specification Language. Language to specify grammar in the Nuance Speech Recognition system
HMM	Hidden Markov Model, a statistical model often used in temporal pattern recognition.
MIBL	Multi-modal Instruction Based Learning
multi-modal	In Human-Computer Interaction: multiple modes of input, usually including modes that go beyond the traditional mouse and keyboard.
multiple inheritance	refers to a feature of object-oriented representation in which a class can inherit behaviours and features from more than one superclass.
NLI	Natural Language Interface, an interface to a robot/computer with natural language expressions, usually by keyboard or with speech recognition.
NLTK	Natural Language Toolkit, a suite of open source Python modules, for research in natural language processing
NLU	Natural Language Understanding, the process of interpretation and making sense of natural language expressions by attaching a meaning to the natural language expression and deductive reasoning.
ontology	

	In computer science: An ontology is a specification of a conceptualization, for example a specification how to network semantic classes. Ontology is also a branch of metaphysics (philosophy) concerned with the nature of being.
OOSG	Out-of-Speech-Grammar (Error). A common error that occurs when the user attempts to express an instruction with grammar that has not been mapped to any instruction and is therefore not in its speech model.
pair-rule	In the card game Scopa: a rule that involves comparing "pairing" two cards by their value.
PCA	Principal Component Analysis, a method of reducing a data set to lower dimensions for analysis.
primitive	see Instruction Primitive
primitive verb	A finite verb in a clause indicating the Instruction Primitive
PROGRAMMAR	A parsing system which interprets the grammars written in terms of programs from Terry Winograd.
Prolog	Programming Language based on first-order logic. It is a declarative language such as SQL or LISP.
PSL	Procedure Specification Language, in IBL.
rule frame	Frame to hold information about a rule. Rules can consist of several Instruction primitives of various types.
rule instruction	In Human-Robot Interaction: an instruction, often verbal that describes a rule. Rules are distinguished from sequential instructions by having conditionals such as "if", "only".
Scopa	An Italian card game.
SFG	Systemic functional grammar, a model of grammar developed by Michael Halliday, see Halliday, (1976).
STR	State Transition Rule (in MIBL).
SLM	Statistical Language Model. A language model that has been generated by the statistical occurrence and word order.
syllogism	Logical argument whereby a conclusion is determined by combining statements
tautology	In logic, a tautology is a formula that is true under any possible valuation. For example ("A or not-A") is a tautology.

taxonomy	The branch of science concerned with classification. from Greek 'taxis' meaning 'arrangement'
tuple	A ordered list of values. In databases often a row in a list of queried results.
unification	in logic, the combination of two terms, if one of them is not instantiated.
uni-modal	In Human-Computer Interaction: single mode of input or communication. Opposite of multi-modal.
word class	A category of words of similar form or function.

1. Introduction

1.1 Need and Aim of the research

PROVIDED the current trend in service robotics continues, service robots will become more common in our households. According to a United Nations study, (UNECE/IFR 2005a), the demand in personal and service robots will rise to 7 million units of personal robots sold within a period from 2005 to 2008. In 2007 another market survey from (World Robotics 2007) suggested that there will be 3.6 million units sold in the period from 2007 to 2010. Current service robots on the market include automated vacuum cleaners, lawn mowers and toy robots.

However, in order to expand to new areas within the household domain, robots must master much more complex tasks, such as identifying and manipulating clothes, manipulating household items and communicate efficiently. A study at the University of Plymouth by (Copleston and Bugmann 2008) showed that the most common tasks that robots should be able to do are preparing dinner, tidying and school work. However very few research groups are working on these problems.

When comparing the user's needs to what robots on the market can actually do, it can be speculated that the market in personal and service robots is driven and limited by what developers can do with robots (at the moment) rather than what users want. In order to meet the user's needs, much research is still to be done.

The motivation of this research project focuses on household robot scenarios. An important issue in the household robot scenarios is that the users of the robots are not trained operators or engineers. Therefore a service robot should be programmable by anybody interacting with them, since there are far too many possible tasks for the robot to be pre-programmed completely. Users want to adapt the robot's behaviour to their individual preference (Wermter 2003, Bugmann 2005). For example the simple task of making tea is a very personal issue, water first or tea leaves first or even milk first? How much sugar, which kind of sugar? How would a computer-illiterate elderly person teach a robot these preferences? Users may not be experts in programming. Therefore "Programming" of service robots should be done in the language of humans. "Programming" between humans is giving instructions from person to person. So there is a clear need for researching human-to-human instructions. Humans instruct (teach) by

speaking and demonstrating actions. Therefore a robot must be able to accept these instructions without the need for the instructor to change significantly his way of communicating. Users in a home environment can not be expected to read a large manual on how to use the robot. Many people would find it difficult or impossible to understand to operate a robot that has many buttons and menus. A report commissioned by the U.K. Government to Sir Claus Moser investigates basic literacy and numeracy (DfEE 1999) stated that one in five adults are functionally illiterate - that is, if given the Yellow Pages they cannot find the page for plumbers. It can be assumed that functionally illiterate people can give a robot instruction by speech and gesture in a similar way that they communicate with others. However they would be unable to deal with an un-natural complex set of instructions from a written manual or even when taught.

A natural way for humans to teach is by actions accompanied by a verbal explanation. Therefore a natural unconstrained human-robot communication interface must be *multi-modal* (spoken natural language + gestures/actions).

The instructions can include rules as well as sequential instructions. Rules, for example are "if it is raining, close the windows." How could these instructions be encoded into a robot? Is it possible such a truly natural human-robot interaction system, where the communication is unconstrained, so the user can communicate freely (free choice of vocabulary, free natural flow of gestures and speech in a limited domain) ? To the best knowledge of the author there is currently no service-robotics project with this emphasis. Other projects that have been devoted to verbal Robot Instruction systems used constrained language, which means that users have to learn specific verbal commands to instruct the robot (Crangle and Suppes, 1994; Torrance, 1994; Huffman and Laird, 1995; Matsui *et al.*, 1999; Perzanowski, 2001; Iba *et al.*, 2002).

The target of this PhD is to contribute to knowledge in the field of human-robot communication. More specifically **how to convert unconstrained multimodal instructions (spoken natural language + gestures/actions) into a knowledge representation usable for robot reasoning and acting**.

The investigation will build on a previous project called Instruction-Based Learning (Kyriacou, 2004; Bugmann et. al. 2004) and the idea of Corpus-Based Robotics which will be elaborated in the next sections.

1.2 Corpus-Based Robotics

A user-programmable robot must use an interface that is natural to the user. User friendliness must be the starting point of the design procedure rather than a later developed feature. Corpus-Based Robotics stands for a design procedure to completely adapt the robot to the user rather than adapting the user to the robot by training. Let's demonstrate this user centred design approach in an example. If the engineering specification is the starting point, the first verbal command an engineer would add to the robot is "go forward one thousand and two hundred millimetres". However no household user will say this command in practice. Furthermore, the engineer might conclude that there is no vision processing required to complete the task. In contrast, a user would say "get me the dirty plates from the dinning table", which may happen to be 1200 mm away from the robot. The designer using the corpus-based approach would conclude that vision is always required for a "move" command, since the user specifies the target in terms of visible objects e.g. "dirty plates, kitchen". To make robots really user-friendly and functional we must first examine how humans give commands and interact, then build a robot according to the interaction model. How to examine human interaction? In general by recording interaction. Interaction between a human teacher and a human or robotic student is recorded and investigated.

These recordings form a so called "corpus", hence the name Corpus-Based Robotics. This approach will ensure a perfect match between requirements of the user, the robots capabilities and the communication level. The design method of Corpus-Based Robotics ensures that the developer is guided towards creating a system that can understand and act upon the end-users needs, because the end-user describes his needs; this information is captured by the corpus.

Previous work carried on the "Instruction Based Learning Project" (IBL) (Kyriacou, 2004; Bugmann et. al. 2004) has shown that it is possible to extract information from a representative sample of the teacher's utterances (the "corpus") in order to:

- Identify primitive procedures that the robot has to be able to carry out i.e. the robot's "prior knowledge"
- Write and tune speech-recognition software to call and combine these primitive procedures.

This approach to the definition of the robot's functionality and natural-language interface (NLI) has been first described as "Corpus-Based Robotics" (Bugmann et. al. 2004) and is outlined in figure 1.

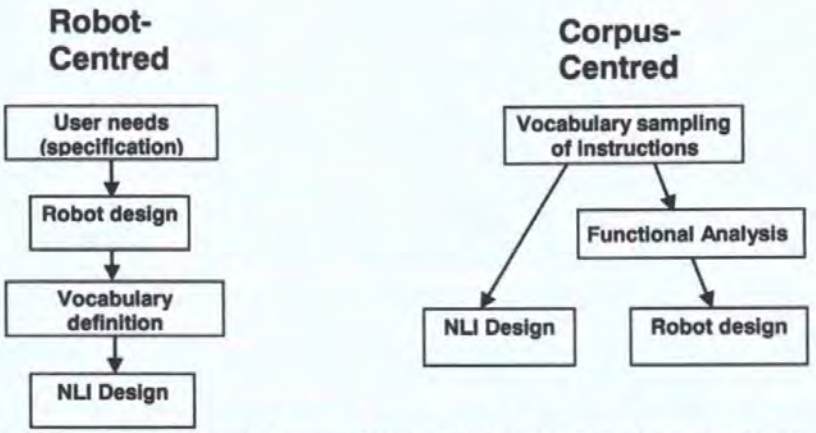


Figure 1-1: Robot vs. Corpus-Centred Natural Language Interface (NLI) design. In the Corpus-centred approach, the content of samples of instructions between humans defines at the same time the vocabulary to be dealt with by the speech interface and the required functionality of the robot. In the robot-centred approach, the functionality is defined first, then the access vocabulary, then the NLI.

By using parts of the corpus as test data while developing the system, the focus is always on the end-users demands. If the corpus is not present the developer will incorporate his own perception rather than what the users want. This can lead to incongruity between the user's need and the robots capabilities, foremost in language capabilities and in functionality and intelligence.

Corpus-Based Robotics goes further than the analysis for language interface designs. A corpus can be used to identify functions that the robot needs to be able to do.

Figure 1-2: Example of a user programmable robot.
Possible Language Primitive: pick_and_place(ddd-socks-filthy,?,+laundry)

Based on the corpus, concepts in the field of understanding task instructions for a robot can be established. These concepts aim at answering questions like:

- How to design a grammar from a multi-modal corpus?
- What knowledge representation and reasoning engine is suitable?
- What semantic structures are used in the language of the teacher?
- How to map language into a semantic representation suitable for a service robot?

These will be discussed in the next chapters.

1.2.1 From Corpus Linguistics to Corpus-Based Robotics

The idea of Corpus-Based Robotics is borrowed from Corpus Linguistics. The initial idea and the term were coined by Guido Bugmann at the University of Plymouth (Bugmann 2004). In Corpus Linguistics text is collected into a database. This collection is called "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 (Biber *et al.*, 1998). 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.

As mentioned earlier, linguistics is divided into corpus linguistics and structural linguistics, whereby in "structural linguistics", sentences are defined from elements, the words and clear structure, the grammar. The same division could be hypothesized in robotics, where Corpus-Based Robotics is opposed to "Structural" Robotics. In Structural Robotics, the robot is build from components.

If there is an analysis of the robot's functionality based on a corpus then the logical consequence is that there is robot-function grammar. See table 1 for clarification.

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)

Table I-1: Corpus vs. Structural

For example, a part of a robot-function grammar of structural robotics could be:

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

Whereas in corpus-based robotics, the utterance "drive forward" would create the need for a drive hardware design. The terminal symbols of this robot-function grammar are the components and software algorithms such as wheel, gearbox and shaft-encoder mentioned above. The idea of robot-function grammar can be exploited by designers to formalise and automate design. The focus of this PhD work is not robot-function grammars; even if they have been discovered here. The focus is corpus-based robotics. Robot-function grammars are worth investigating in future projects and can possibly be combined with evolutionary computing to find the optimal design of a robot, whereby the genomes are created from grammar rules to avoid impossible configurations. Grammars have been applied to related ideas such as planning a task and encoding task constraint and operation applicability into the grammar. However in the case of robot function grammars, the design and configuration of the robot is also included into the grammar.

1.3 Overview of the Thesis

The thesis starts with a review of related work in chapter 2 and earlier work in chapter 2.2. The main part of the thesis is in chronological order of the robot design and in the order of the information flow going through the robot. The flow starts with input of gesture (chapter 4), its integration with language (chapter 5), the creation of speech recognition grammars (chapter 6), and continues with reasoning and action upon those inputs (chapter 7). Finally actions that the robot produces are described in (chapter 7.8.6). Chapter 8 describes experiments and results and the thesis is concluded in chapter 9. Along the way important scientific contributions are described which advance human-robot interaction and learning. A particular focus will be true natural interaction which can only be achieved through contributions in multi-modal integration (chapter 5.3), the application of rule frames (chapter 7.3) and the formalisation of the corpus-based approach.

In a nutshell the Corpus-Based Robotics approach can be formally described as the following procedure:

Summary of Design Procedure

1. Collection of the Corpus (Chapter 3)
2. Transcription (Section 3.4) and Corpus Context tagging (Section 6.1)
3. Identification of Primitives (3.5.2, 3.5.3 and Ontology (6.2.2 , 7.4)
4. Implementation of Gesture Recognition (Section 4.1)
5. Creation of Timing Histograms for Multi-Modal Integration (Section 5.2.4)
6. Creation of Grammar (Chapter 6)
7. Implementation of Primitives, State Transition Rules and Robot Low-level Primitives (Chapter 7)
8. Dialogue Design (Chapter 7.10, 8.7.1)

Figure 1-3: Summary of the formal design procedure of Corpus-Based Robotics

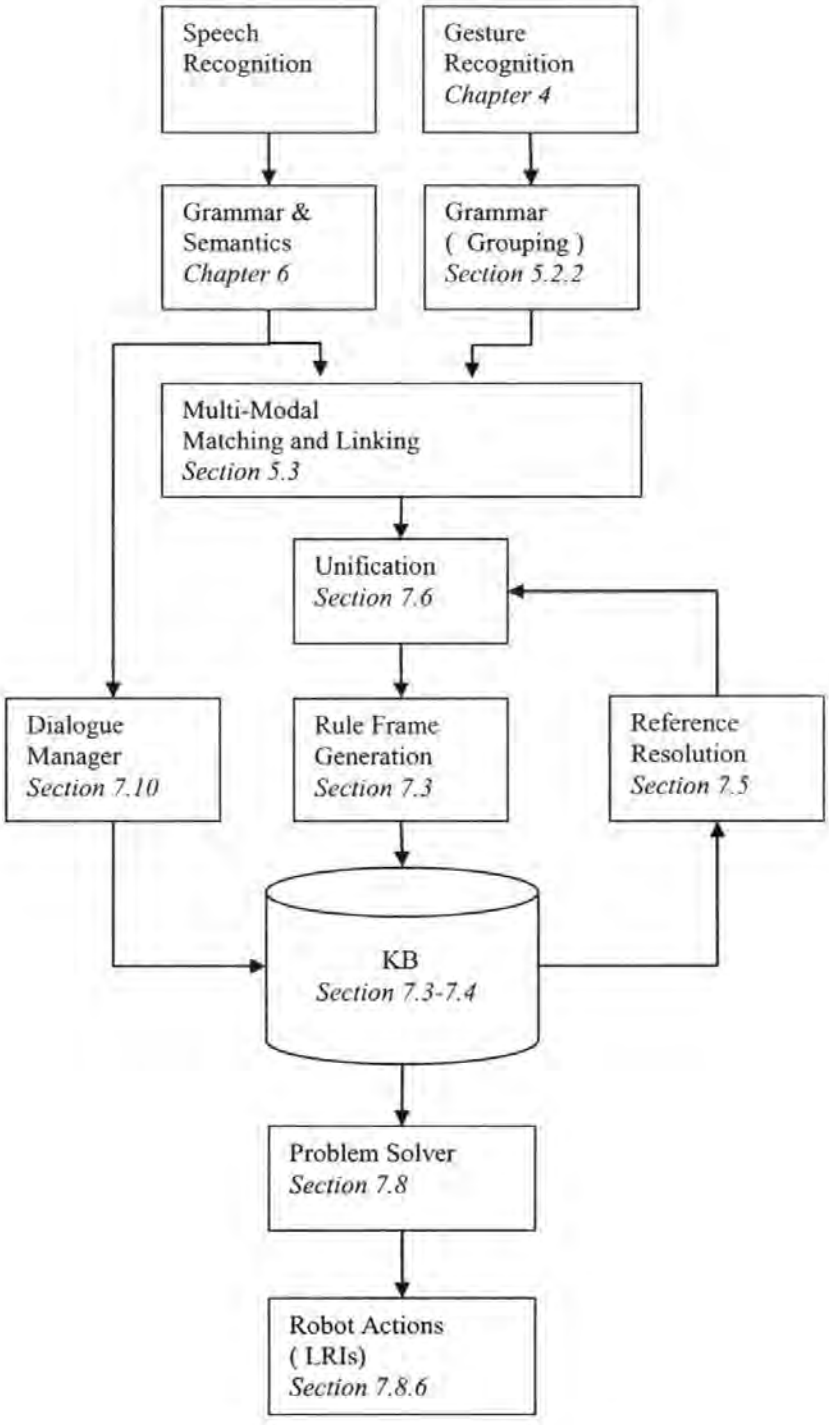


Figure 1-4: Overview of information flow in the MIBL system with Chapters of the thesis. Generally the information flow is from the inputs of Speech and Gesture at the top through to the output of "Robot Actions". This diagram gives an overview so it is easier to see how the parts described in the thesis fit together. Grammar will be described in chapter 6 while the multi-modal module will be described in chapter 5. Refer to the Chapter titles for an explanation of the modules.

1.4 Main Contributions of the Thesis

- *The Corpus-based design method*: the detailed description of the corpus-based design method which applies to robotics, but may extend to a engineering product design method (throughout thesis, Figure 3)

- *A multi-modal integration algorithm:* Includes speech-to-language pairing and unification during unconstrained free flowing interaction (sections 5.3 and 7.6)

1.5 Minor Contributions of the Thesis

1.5.1 Theoretical

- *A High level learning and reasoning engine for a service robot:* Most current service robots are only able to learn simple movement or a simple sequence of actions from a human instructor. The development of rule frames, which are a frame-based intermediate representation for human level instructions (sections 7.3, 7.4 and 7.8) are presented in this thesis as the core of the reasoning engine.
- *The application of linguistics concepts to robotics:* the discovery of robot-function grammars (section 1.2.1)
- anaphora resolution in a natural language discourse using rule frames (section 7.5)
- *A Grammar generation method:* Improves speech recognition through the application of an ontology and a clause-based grammar that fits the corpus and therefore the users most frequent utterances (section 6.4). Reduces overgeneration that can lead to nonsense translations.
- *Gesture Grammar:* Application of context free gesture grammar for grouping of gestures in order to align with the speech modality (section 5.2.3).
- *Observations on human behaviour during teaching card games and rules:* The corpus-based approach uses human behaviour to create recognition and understanding robots. The results described in chapter 8 give descriptions on how humans perform when they teach and how often they make mistakes.

- *An Investigation of out-of-grammar errors in a growing corpus-based grammar:* In a deployment test (chapter 8.8), the investigation will show that the influence of adding new grammar rules to a corpus soon loses its impact when the corpus grows to a considerable size.

1.5.2 Technical

- *An Implementation of an agent:* with associated test results (section 8.7 and 8.8)
- *A Multi-modal transcription tool (MuTra)* (section 3.4.1)
- *An annotated multi-modal corpus:* The corpus can be used for further research (section 3.5)
- *A novel method of corpus collection for Multi-modal corpora* for service robots: The use of a touch screen and not allowing direct visibility between human-to-human gestures provides a novel method of collecting data for free-flowing future human-robot interaction without the need of building the robot first.

2. On Learning in Robotic Systems and Natural Language Understanding

This chapter gives an overview of and discusses the relevant background and literature in Artificial Intelligence and Robotics. In particular natural language understanding and learning robotic systems are investigated. Initially the difference between skill and task learning is laid out and previous work on the IBL project is presented to gain an insight of the background and motivation for this PhD work.

2.1 Skill learning and Task Learning

Learning methods for teaching robots can be divided into subsymbolic skill learning and symbolic task learning.

2.1.1 Skills and Skill learning

Skill learning for a robot means refining the robots closed loop control systems that are responsible for actions. Skill learning also extends to learning to recognise salient features in sensor data. It could be summed up as learning to use the basic sensori-motor system. A typical skill learning example would be the skill to balance and walk or to pick up an object without dropping it. Most skill learning involves negative feedback systems. They may also include a model of the robot and a prediction of the consequence of its own action (Demiris and Johnson, 2003). Many skills, such as learning to balance could be described as skills that a human learns in its early years of childhood, and hence many researchers are inspired by human learning and try implementing these biologically inspired learning mechanisms in robots. This has initiated an investigation into developmental robotics (Lungarella *et al.*, 2003, Asada *et al.*, 2001). Recently the involvement of the motor systems during observation has become of particular interest.

Skills can be defined as the ability to use the sensori-motor system successfully.

2.1.1.1 Learning of Motor actions

Before being able to learn skills from others the robot must be able to drive its own actuators to a desired configuration. This can be achieved by feeding back information about the configuration of the robot's end effectors and compare them to the input. With this method it is possible for the robot to learn to predict the consequences of its own actions stored as a model. Using this model as an inverse it becomes a controller see (Dearden and Demiris 2005). The alternative to this pre-stage is of course to manually implement a traditional control system and to combine it with robot kinematics (McKerrow 1991).

2.1.1.2 Skill Learning by Imitation

(Schaal 1999) defines imitation learning as being concerned with three important issues: efficient motor learning, the connection between action and perception, and modular motor control in the form of movement primitives.
(Calinon and Billard 2007) use principal component analysis (PCA) in the recognition phase to identify parts of the action that is demonstrated. The learning robot from (Calinon and Billard 2007) must then generalise over multiple demonstrations. They describe learning sequential motor actions as challenging and use Hidden Markov Models (HMMs) to encode the sequential actions (patterns of motion).

2.1.2 Tasks

In this PhD work learned motor actions are defined as *action primitives* and a task can then be defined as follows:

A task can be defined as the organisation and application of skills in a sequence to fulfil a goal.

The structure of a task has an inherently symbolic nature. Evidence for that is that a task can be easier explained by verbal communication than a skill. A task such as finding a route can be explained and "learned" verbally, however how to play tennis with a racket

can not be learned verbally, especially without demonstration and imitation. Verbal communication is inherently symbolic.
A robot that is able to learn and apply tasks must previously have learned the skills. This makes skill learning a necessary foundation and therefore more important than task learning. However, to make service robots a reality, both are required and the focus of this PhD work is task learning. The skill learning processes have been minimised by the use of a touch screen rather than a camera and a humanoid robot arm.

In corpus-based robotics skills are called action primitives.
Mental skills are called knowledge primitives.

Skills can be named and listed; therefore the corpus-based robotics approach lets the robot designer identify primitives at a human level. In the IBL and MIBL scenario, these primitives are in the form of task learning rather than skill learning. There are no utterances such as "push a bit harder" which would indicate skill learning. In other scenarios, such as learning to drive a car with a driving instructor would contain many instructions of skill learning. Table 3-9 shows the identified language primitive types: fact, conditional, context, action. In a corpus containing skill learning, it is debatable if new primitive types are required, or if skill learning is part of action primitives.

An early robotic task learning system is described by (Kuniyoshi *et al.*, 1994). The system extracts knowledge to learn a sequence of an assembly by observations of a human. Kuniyoshi shows how visual recognition can be segmented into an action sequence. This sequence has dependencies which are described in a hierarchical task plan. In experiments, Kuniyoshi shows how a robot learns the assembly of blocks on a table and stores all information about the task in these clean hierarchical structures. Typical for task learning systems, is the storage of usually sequential actions into hierarchical structures or frames.

2.2 Previous Work: The IBL Project

2.2.1 Introduction to the IBL Project

Previously a project on Instruction-Based Learning (IBL) was carried out at the University of Plymouth in cooperation with the University of Edinburgh (Dr. Ewan Klein). The Project was running from 2001-2004 and my PhD work, the MIBL project, is partly a continuation of this work. The IBL project focused on route instructions given to robots by naive users. A dialogue such as the following was possible between the user and a robot:

User: "Go to the University."
Robot: "How do I go there?"
User: "Take the third turning to the left..."
Robot: "Next instruction please."
User: "...take the third exit off the roundabout..."
Robot: "Next instruction please."
User: "The University will be on our right."
Robot: "OK, it's done."

The route instructions were then carried out by an 8 cm by 8 cm wide robot in a model town. The robot had an onboard camera to identify road junctions. At the beginning of the project subjects were invited to give route instructions. These instructions were audio recorded and formed the IBL corpus. The corpus contained 144 routes produced by 24 paid subjects instructing 6 routes each (Bugmann 2003). The subjects were told that a human would remote control the robot through the eyes of the onboard camera from another room.

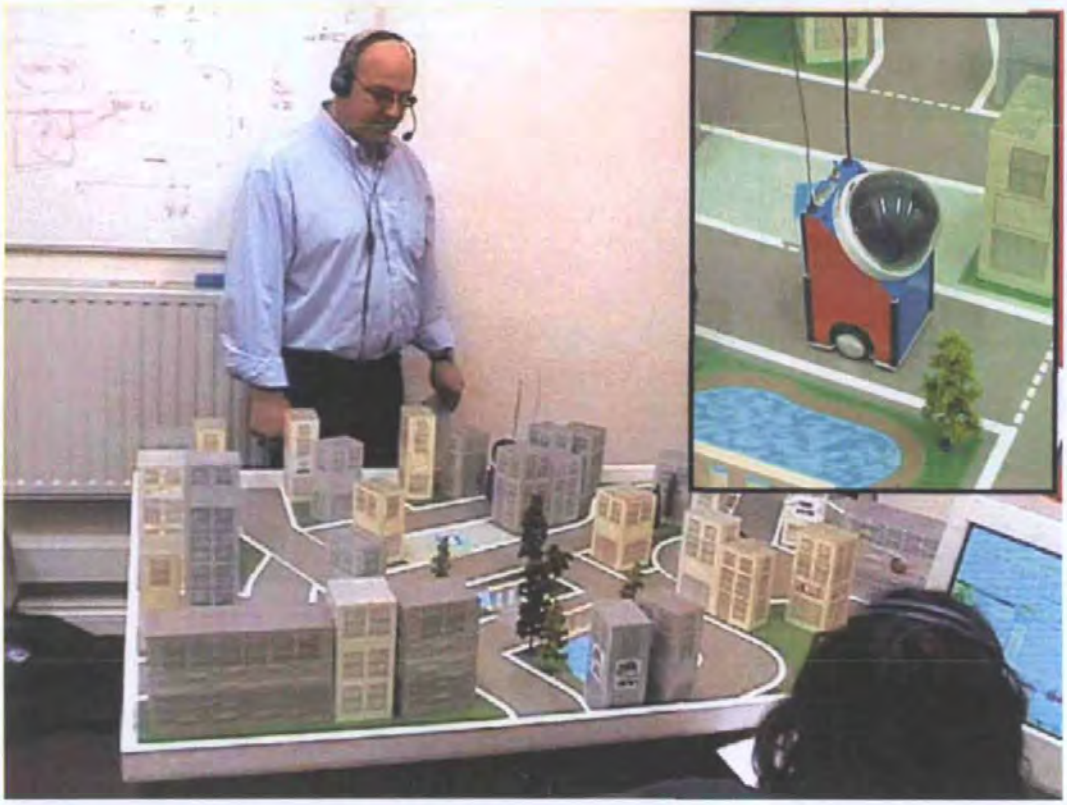


Figure 2-1: Experimental Setup of IBL: Subject giving route instructions to the mobile robot.

Using the corpus of recordings a speech recognition system was build that could recognise the route instructions and convert them into executable procedures.

2.2.2 IBL System Overview

The robot translated human instructions to robot procedures in a two stage process. First the text was converted into an intermediate semantic representation known as Discourse Representation Structure (DRS). From there the structures are mapped to robot procedures by the use of mapping rules defined in Procedure Specification Language (PSL). (Lauria *et al.*, 2002).

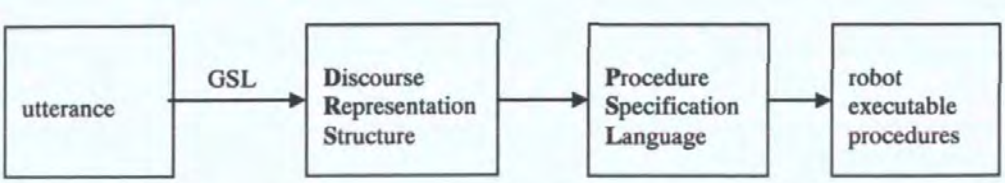


Figure 2-2: Overview of conversion process of speech to robot exec. procedures in IBL

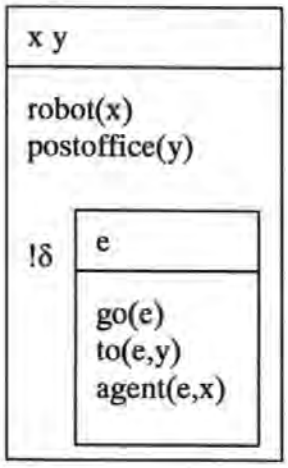
When the robot operates, the speech recognition grammar converts an utterance directly into DRS. The idea of mapping utterances directly into semantics has advantages. An

intermediate parsing stage is eliminated. Possible mismatches, at the intermediate stage that could create strings that do not make sense in semantics, are eliminated, although other problems remain.

The final grammar is in GSL (Grammar Specification Language) format. This is the format used for the Nuance[1] speech recognition software that is used in the IBL project (and in this work).

In order to create an utterance-to-DRS grammar, firstly a context-free backbone of the unification grammar is created (Bos 2002). Johan Bos created a compiler called UNIANCE that will carry out the conversion. It used syntactic features in the translation to non-terminal symbols in GSL. Terminal symbols represented the vocabulary of the corpus. Unification grammars can contain left recursive rules; however GSL only allows right recursive rules. Therefore the UNIANCE compiler eliminated left recursive rules.

The vocabulary and grammar rules have to be limited to the domain, so that speech recognition performance is increased. In order to achieve that, only the grammar rules of the context-free backbone, that were hit when parsing the IBL corpus, are used in the final grammar. The vocabulary used in the IBL corpus became the terminal symbols of the final grammar.



D={d1,d2,d3,d4,d5,d6,d7,d8}
F(possible_world)={d1,d2,d3}
F(robot)={(d1,d4),(d2,d4),(d3,d4)}
F(postoffice)={(d1,d5),(d2,d5),(d3,d5)}
F(action)={(d1,d2,d3)}
F(go_from_to)={(d2,d4,d6,d5)}
F(at_loc)={(d1,d4,d6),(d3,d4,d5)}

Figure 2-3: A graphical representation of DRS of the utterance "Go to the post office". On the left. !δ denotes that there is an action and the action is commanded. e, x, and y are discourse referents to show the dependency between the terms robot, go, to, postoffice and agent. On the right is a text representation of the same command. It shows that DRS is difficult to read.

[1] Nuance Communications, Inc., 1 Wayside Road, Burlington, MA 01803,USA (www.nuance.com) Nuance 8 and 8.5 was kindly provided by Nuance Communications for research purposes free of charge.

Discourse Representation Structures (DRS) are a well understood framework that accurately describes dependencies between the semantics and allows the interpretation of pronouns and other anaphoric expressions (Kamp and Reyle 1993). It allows representation of text in first order logic. Again the "go to the post office" example now in first order logic:

$$\exists w \exists x \exists y \, (possible\ world(w) \wedge robot(w,x) \wedge postoffice(w,y)$$

$$\wedge \exists v \exists a \, (action(w,a,v)$$

$$\wedge \exists e \, (go(a,e) \wedge to(a,e,x) \wedge agent(a,e,y))))$$

While DRT is a valid semantic representation, it is not a procedure that can be carried out by a robot. In general terms DRT is more orientated at natural language structures rather than actual robot functions. In order to map this representation to a robot function (Robot Primitive) a rule base for mapping rules had to be created. The rules of this rule base are described as Procedure Specification Language (PSL). Several utterances that have different DRT representations can still have the same meaning to the robot and must therefore point to the same Robot Primitive. For instance, the expressions "take the next left, turn left, take the first turn left, etc" must all be mapped to the Robot Primitive turn(direction="left",ordinal="first") According to the final EPSRC report (Bugmann 2003) a total of approximately 200 PSL rules were required for the 15 Robot Primitives of the IBL corpus. As an example for PSL the utterance "Go to the post office" can be mapped with the following PSL rule:

event(X) &go(X) &to(X,Z) &$landmark(Z)->

go(prep = 'to'; landmark = $landmark(Z))

to the Robot Primitive procedure *go(prep='to', landmark='postoffice')*

Since the IBL project was using route instructions, the resulting system was developed to deal with sequential instructions. Other forms of instructions, such as general rules, which apply at any time during the task, such as "Stop at the petrol station if you run low on petrol", did not occur in the IBL corpus, and were therefore not investigated.

The system could not deal with conditionals, such as the one above, that were not found explicitly in the corpus (Lauria *et al.*, 2002). In route instructions, sentences starting with "if" instructions are generally just a colloquial way of expressing a sequential instruction, as in the following example from the IBL corpus: "...okay if you carry on straight along this road and if you take the third left you will go over a bridge..."

Therefore, to develop a more general instruction system, there is a need for looking at a different application, where instructions not only include sequences, but also other instruction structures. In imperative programs these would be decisions and repetitions. However, in the declarative paradigm, programs consist of lists of goals and a set of rules (see e.g. PROLOG). It is unclear which paradigm is a more useful representation of human instructions. This is one of the questions that need to be addressed by analysing a new corpus of instructions in a different domain.

2.2.3 Difference between Programming by Demonstration and Instruction-based Learning

It was found that the task in the IBL project was only explained once, and in MIBL project instructions have been explained once and typically the teacher was giving a demonstration with verbal comment after. Following that, the robot / human student had understood the instructions. This is called a one-shot task learning process by (Jung H.C *et al.*, 2007). Other researchers would refer to this as "Programming by Demonstration" (Dillmann *et al.*, 2002). Programming by demonstration can be broadly defined as creating an generalised representation / program of a task that has been demonstrated to the robot. The robot should then be able to execute the learned task using the abstract representation (program).

Defenders of one-shot task learning, including this work argue that a service robot can only be useful and efficient if it can learn a task as fast as a grown up human, in one shot. An adult robot must have learned all basic sensori-motor skills, like a grown up human, to be able to accept one-shot learning tasks. In the robot's "childhood" it must learn its skills, such as how to move its actuators accurately, with methods described by (Demiris and Johnson 2003). A competent robot should be able to do both, "one-shot" learning and skill learning.

Let us try to give a definition of Instruction Based Learning, to distinguish better from skill learning, "Programming by Demonstration" and other approaches of robot learning:

IBL is the process of learning a task from a teacher through instructions, usually verbal. The learning process may be supported by, but is not depending on a demonstration.

Integrating these supporting demonstrations require a multi-modal system, therefore MIBL is defined as Multi-modal IBL.

2.2.4 Conclusions from the IBL Project

The IBL project concluded in the EPSRC final report (Bugmann 2003) that the domain of route instructions only included sequences and no decision making processes and loops. This led to only limited reasoning capabilities of the robot. Attempts were made to check the consistency of an explained route and also to recognise previously learned routes. A state based reasoning approach was taken allowing the robot to predict the consequences of an action, such as "turn left".

In principle the first order logic representation that DRS allowed, is a powerful mechanism for reasoning and representation of rules, as well as sequences. However the lack of grounding of the produced semantics and incompatibility with the robot functions required the translation with PSL. At this point the grounding (mapping) is made between DRS semantics and actual robot functions. However the clear structure of the lambda calculus that would enable deduction and reasoning is lost at this point.

In corpus based robotics the robot is build according to the findings of the corpus. Therefore only being able to process sequences is not a disadvantage. However the aim of generalising and researching the concept of corpus-based robotics and the translation of human instructions to robot instructions, another domain has to be investigated, where decision making and loops is significant. This is a further reason why a follow-up project started which is presented in this thesis.

The corpus-based approach aims at covering the most common expressions that the users say to the robot, and this was demonstrated in the IBL project. The project also has shown that the corpus is never complete, i.e. there are always instructions that have not been covered by the corpus and that the robot then can not deal with. This is a limitation of the corpus-based approach. In a future project, this limitations have to be investigated.

Previous research in our group focused purely on verbal instructions which are sufficient in some cases where a demonstration with physical objects is not required. In practice, many tasks are explained using a mixture of verbal instructions, gestures and demonstrations. Thus, a truly natural interface between human and robots must be multi-modal. This is one of the features included in this PhD work and has been the inspiration of the name of the project: **MIBL** (**M**ulti-**M**odal **I**nstruction **B**ased **L**earning). Multi-modal systems combine gesture and language.

Many ideas and concepts of this PhD work have their origins in the previous work. The idea of Corpus-Based Robotics and the search for language primitives are from the IBL project. Furthermore the idea of verbal communication that appears unrestricted to the user. Whereas Corpus-Based Robotics was coined during the IBL project, in this PhD work the starting point was how to formalise the idea of Corpus-Based Robotics. A major difference in architecture between the IBL and MIBL system is that utterances will be directly converted to language primitives in the grammar, rather than going through the complex DRS and PSL system. The advantage is the simplification of the process, however DRS is a powerful tool showing the relationships between semantics in an utterance and to the whole dialogue. MIBL has a more primitive reference resolution as will be shown later in Section 6.2.4, 7.5 and 7.6.

2.3 Human-Robot Interaction Robots

A general overview of Human-Robot Interaction systems can be found at Fong *et al.*, (2002) , Kiesler and Hinds (2004), Yanco and Drury (2004). However, this review will focus on Human-Robot Interaction systems which have the most similarities in philosophy and implementation to the MIBL project.

The current trend is to focus on the fundamental issues of Human-Robot interaction and general Robot learning from a developmental Robotics point of view. This trend has continued with the start of new research projects around the world. iTalk is a new project with regards to the fundamental perspective since its focus is to create a child robot with the capabilities of a 2 year old (Cangelosi 2007).

A further multi-million project that has started with possible impact on Human-Robot interaction is CoTeSys (Cognitive Technical Systems). CoTeSys explores cognition for technical systems such as vehicles, robots and factories (Buss *et al.*, 2007). The emphasis is on the incorporation of cognitive capabilities such as perception, reasoning, learning, and planning into traditional technical systems. One of the outcomes is the improvement of interaction with these systems. Buss recognises that multi-modal interaction of humans and systems which involves emotion, action and intention recognition lies at the highest and most complex levels of cognitive systems. The main aims of CoTeSys are wider, they are the technical systems will have a form of self-assessment and can therefore learn and improve themselves.

2.3.1 COGNIRON Robot Biron

COGNIRON (The Cognitive Robot Companion) is a European Union funded project that had the objective of the development of cognitive robots whose "purpose in life" would be to serve humans as assistants or "companions" (Kyriakopoulos and Siciliano, 2004). It aims at developing methods and technologies for the construction of such cognitive robots able to evolve and grow their capacities in close interaction with humans in an open ended fashion. Parts of these projects have identical objectives with the motivation behind IBL and MIBL. In particular the investigations by the COGNIRON research groups at the University of Bielefeld (Haasch *et al., 2004*) and the University of Karlsruhe (Dillmann *et al., 2002*) have relevance to this work. In the

COGNIRON project a research groups under the leadership of Kerstin Severinson Eklundh at the Royal Institute of Technology in Sweden worked on the social aspects such as the distances and orientation of the robot when giving commands (Huettenrauch et.al. 2006). A further group of COGNIRON at the University of Hertfordhire concentrates on social aspects and on how a robot can learn new skills from a human demonstrator (Saunders *et al.*, 2007).

The COGNIRON project addressed a large variety of real world human-robot interaction problems and produced multiple HRI robots to carry out the research.

As part of the COGNIRON project, a group at the KTH-Sweden collected corpora on multi-modal human-robot interaction. The corpora were used to study the users behaviours (Green *et al.*, 2006). As in this PhD work, they have identified the importance of user-based studies with multi-modal corpora. The results showed that users can be put into 4 types:

"Directors": actively persistively controlling the robot
"Players": interactive with the robot, passively let the robot act first
"Manipulators": also interactive with the robot, actively controlling the robot
"Pointers": little control over the robot or the environment,
adopting interaction to the situation

These user types have possible robot design implications so that the robot can adopt to the type of user. In the MIBL project, where all these types of users can easily be identified in the corpus, an adoption of the dialogue model to the user types would be useful in future work.

The University of Bielefeld, investigates multi-modal dialogues in a home tour scenario. Their robot, called BIRON, has the capability to detect which person out of a group it has to pay attention to (Haasch, A. *et al.*, 2004). The person can then engage in a simple dialogue with the robot introducing objects to the robot. (see figure 2-4). BIRON can focus microphone beams on the person thus improving speech recognition performance. The domain of the reasoning and speech recognition engine of BIRON is limited to a simple dialogue. BIRON only understands simple sentences that introduce objects, e.g.

"This is a plant". The robot has a vision system with gesture recognition and object recognition, a natural language interface and laser range finders.

Figure 2-4: Typical interaction with BIRON. "This is a plant". picture from (Haasch A *et al.*, 2004 with permission)

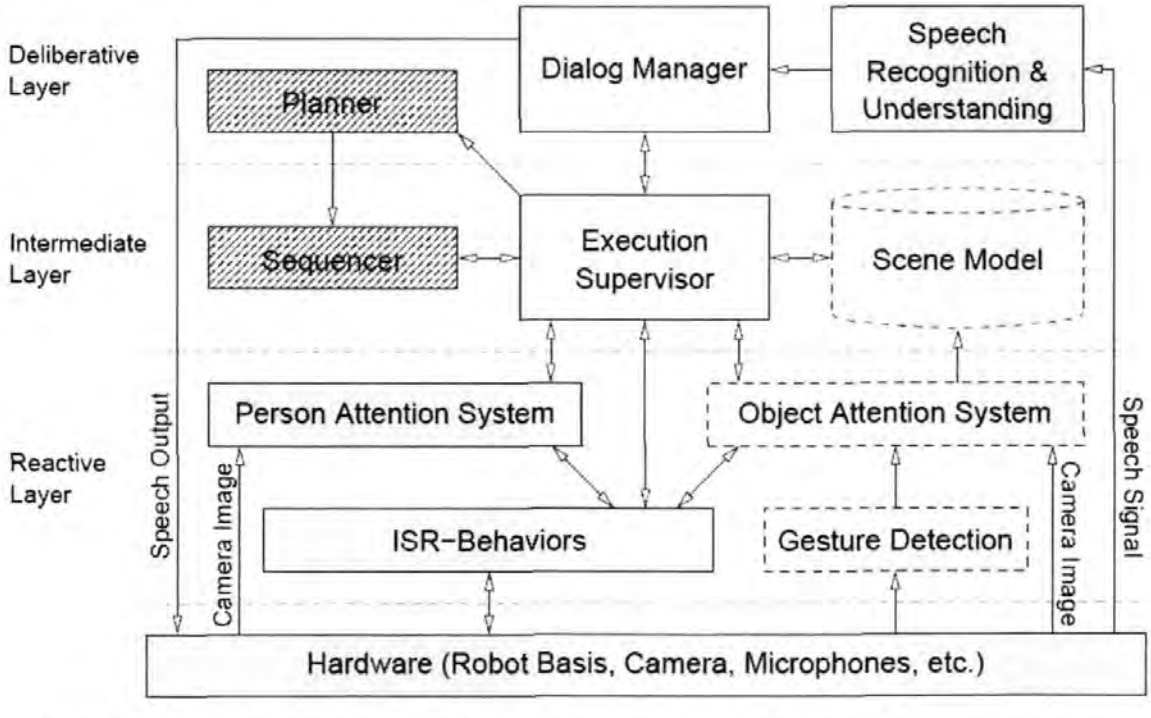


Figure 2-5: System overview of BIRON: horizontal layers in the hierarchy ensure that low level behaviour (reactive layer) continues to operate while high level plans are executed (Intermediate Layer). Speech recognition is based in the deliberative layer, since recognized sentences contain high level spoken instructions that command the robot. The layout of the architecture was inspirited by Brooks 1986. Figure from (Haasch A *et al.*, 2004 with permission).

The Bielefeld group recognized some important points which are relevant to this research:

- Combining uni-modal processing results into a multi-modal data-association framework makes the system robust against errors.
- Human communication partners can not be expected to wear special equipment such as close-talking microphone or data-gloves.
- a semantic-based grammar is necessary to extract the meaning of the sentence (parsing and subsequent interpretation is not acceptable since these kind of parser do not consider semantics and therefore introduce errors)
- missing information in an utterance can often be acquired from the scene with other sensors (Wrede *et al.*, 2004)
- the system uses a horizontal hierarchy (Reactive Layer, Intermediate Layer, Deliberate Layer (see figure 4)

The research in Bielefeld concentrated on the reactive layer (Person Attention etc.). The dialogue and high-level reasoning was not investigated enough to make this service robot execute all commands necessary in its domain. This was not directly the aim of the project, the project scenario concentrated on a home tour where the service robot has just been bought and is shown around the house. The human-user introduces objects in the house to the robot. The robot understands sentences such as "This is a plant", however it might not understand sentences such as "Please water cactuses only once every fortnight and the other plants weekly". That is what people really want to tell the robot. That would be a typical household job. A corpus-based approach would potentially reveal this and this PhD work will develop the methodology of how to approach such complicated instructions.

For further reading, there is another project by Bielefeld University (Steil et. al 2004) about a robot called GRAVIS. The project concentrates on gesture recognition and learning of grasping of objects. The dialogue system is based on an investigation of a corpus of human-human and simulated human-machine dialogs. Language and gesture integration is achieved with a Bayesian network. In contrast the MIBL project tried to avoid probabilistic approaches if they are replaceable by symbolic algorithms.

2.3.2 Karlsruhe Robots

The German Collaborative Research Centre (Sonderforschungsbereich) for "Humanoid Robots - Learning and Cooperating Multimodal Robots" at the University of Karlsruhe has built two humanoid robots (Albert and ARMAR) with the target of interacting with humans in a service robot scenario (Dillmann *et al.*, 2002). Their emphasis lies in building a complete system that can interact through observation and tracking of objects, gesture recognition and speech recognition. The research group recognizes that interactive programming must be a One-Shot-Learning process or it would be very annoying to the user. Another important point from the Dillmann paper is that there seem to be no system so far that integrates the control, basic interaction methods and programming techniques for humanoid robots into a single system. The robot build by the research institute can learn to fetch and carry tasks and can be taught fine manipulations of simple objects.

Figure 2-6: Albert 2 *(figure from Dillmann et.al. 2002, with permission)*

Figure 2-7: Humanoid Armar III : 43 degrees of freedom on a holonomic wheeled platform (figure from Asfour et.al. 2007 with permission)

Data from the recognition of trajectories and grasping is segmented so it can be broken down into a sequence in a semantic format. This system conforms with the ideas of this PhD in this respect. However sequence learning alone is not enough. For more advanced tasks, rule learning is necessary.

2.3.3 Multi-modal Human-Robot Interaction systems

Multi-modal human-robot interaction systems have been investigated by several research groups around the world (Iba *et al.*, 2002; Wermter *et al.*, 2003; Dillmann *et. al.*, 2002). The challenge of multi-modal robots lies in combining the modalities to form a coherent information stream that modifies the internal model of the environment.
One of the first research projects to investigate multi-modal integration is described in Bolt's famous paper "Put-that-there" (Bolt, 1980). Bolt describes a "Media Room" with a virtual space projected against the wall, a DP-100 NEC speech recognition system and a tracking device, strapped to the users wrist. The user can point to objects on the projection and say utterances like "Create a blue square there." The system recognizes the pointing direction with the tracking device at the time the word "there" was uttered. Combining gesture and language is one of the focus points of this PhD work (Wolf and Bugmann 2006).

Multi-modal integration has also been addressed in the past by (Oviat 1999; Johansson 2001; Nigay and Coutaz, 1995 and Chai, 2003), where it is sometimes referred to as multi-modal fusion (see Djenidi *et al.*, 2004). Curiously, most researchers working on multi-modal interfaces do not appear to have addressed the problem of pairing the gesture and language channels before integration. This is probably due to the fact that experiments often constrained the human-computer interaction in such a way that pairing which gesture with which language was not an issue. Constraints such as click-to-speak or limitation in computing power influence the timing of the natural flow of speech and gesture. (Oviatt *et al.*, 2000) gives a good overview of multi-modal integration research projects for further reading.

Long response times of the robot/computer are often the cause of an interrupted flow of conversation. This actually simplifies the pairing problem and therefore may not have come to the attention to many other researchers. The pairing problem, especially in a free flowing conversation is therefore one of the focus points of this PhD work.

2.3.3.1 Early versus Late Fusion

The concept of early fusion interlinks the gesture recognition system with the speech recognition system at an early stage. In this case the recognition systems are usually based on the same computational model. Recently (Schillingmann *et al.*, 2007) investigated Hidden Markov Models and n-gram models to generate action-specific language models, with the goal of early integration. Another computational model that incorporates the early fusion of speech recognition and vision are Semiotic Schemas (Roy, 2005). Roy showed that early fusion improves speech recognition in (Roy and Mukherjee, 2005). Early fusion models have also been used in emotion recognition; see (Wimmer *et al.*, 2008).

In contrast, in the late fusion model, the fusion happens after speech recognition and gesture recognition is completed (Djenidi *et al.*, 2004). In the MIBL project recognition and grouping of actions are processes that are designed to initially be independent from speech processing. This approach corresponds to the late-fusion model. It is the opinion of the author that the method of late fusion is far easier to implement, since an off-the shelf language recognition package can be used. In our case the package NUANCE 8.5. was used.

2.4 Natural Language Understanding Systems

2.4.1 A Brief Historical Overview

Literature in the areas of natural language processing and natural language understanding will be reviewed here. Major historical works include ELIZA (Weizenbaum, 1966,1976), SHRDLU (Winograd, 1971), MARGIE (Schank and Abelson, 1977). All these mentioned above use text input, rather than speech recognition. For further reading on contemporary work see (Mann, 1996; Bos 2002; Bugmann *et al.*, 2004) is recommended.

2.4.2 ELIZA

ELIZA is a natural language processing system that enables a user to communicate with it via a console (Weizenbaum, Joseph.(1966)). ELIZA poses as a Rogerian psychotherapist. A Rogerian psychotherapist is very passive and understanding and lets the patient talk about their problems. Empathic understanding supposed to have psychological healing powers according to Rogers.

This is why Weizenbaum decided to make ELIZA a psychotherapist. When he was confronted with the question: "And what was it that motivated this Rogerian guise?" Weizenbaum answered:

> "From the purely technical programming point of view then, the psychiatric interview form of an ELIZA script has the advantage that it eliminates the need of storing *explicit* information about the real world."

This statement tells us that Weizenbaum recognized that "real world knowledge" i.e. semantic processing using a knowledge base is a difficult thing to implement. The program ELIZA demonstrates also that even it has no "grounded" language it can pose intelligent by replying to the user with sentences that refer to what the user said. For example if the user says "I'M DEPRESSED." , ELIZA is programmed to answer "I AM SORRY TO HEAR YOU ARE DEPRESSED" because it was programmed to do so by a simple statement along the lines of:

```
IF sentence has Subject="I" AND Verb="am" AND object="depressed"
THEN Answer="I AM SORRY TO HEAR YOU ARE DEPRESSED"
```

Even if the program only responds to key-words, the users are under the impression to be understood by ELIZA. As the example above shows, however, there is no attempt to connect the rule sets to infer new knowledge or even to ground it to the physical world. ELIZA became a very popular program, since it was one of the first attempts to imitate humanlike communication.

2.4.3 SHRDLU

SHRDLU is a program written by Terry Winograd between 1968 and 1972. It is able to understand natural language text input. He showed by this implementation, that if language is confined to a domain ("a micro world"), the computer is able to understand and act upon user requests. The micro world he chose is a table with blocks, cubes, pyramids and a box. These objects have colours and sizes assigned to them. This representation has become quite famous in A.I. under the name "Blocks World" as an idiom for simplifying a problem by restricting the complexity of the environment. It has a vocabulary of around 200 words.

Winograd recognized that syntactics, semantics and logical inference are inseparable in his PhD thesis, (Winograd, 1971). He represents knowledge as procedures, rather than as declarative statements. A procedure can make use of:

- grammar
- semantics
- deductive logic
- other procedures

As the system parses a sentence it will make use of the grammar procedures which can also call semantic interpretation procedures during the parsing process. This is a flexible and powerful method of language parsing.

This increases the flexibility of his representations, since a procedure can call and combine with any other procedures. This is the reason why Winograd has chosen to implement SHRDLU in Lisp. Lisp has the capability to ignore the difference between procedures and data.

The grammar used in SHRDLU is a form of context sensitive grammar called systemic[2] grammar. Systemic grammar helps to organize the correlation between features of natural language constituents and their semantics. This is important for understanding systems, and this was probably the reason why Terry Winograd has chosen systemic grammar. Winograd recognized that context free grammars are over-generative. The grammar rules are written in "PROGRAMMAR", a general parsing system which compiles the grammar to Lisp code. Winograd admits that it was not practical to implement the whole of systemic grammar, and that the resulting grammar is more "practical". It should be noted that the implemented grammar is not a complete valid grammar for English language. And it is definitely not a standard English grammar. However, it enables the extraction of the semantics of most sentences in order to build a natural language understanding system.

2.4.4 Schank's natural language understanding systems

In the late seventies and eighties Roger Schank developed several natural language understanding systems. Schank was working with a group of scientists (Cullingford, Rieger, Goldman, Abelson, Riesbeck, Lehnert and others) perusing the same basic ideas i.e. : creating a methodology that leads towards the eventual computer understanding of natural language (Schank and Abelson 1977).

MARGIE was one of the first parsers that created conceptual representations directly from the input text without doing an intermediate syntactic description of the sentence.

SAM (*Script Applier Mechanism*) is a natural language understanding program in the domain of stories. It is a successor of MARGIE (Schank and Abelson 1977). SAM was created by Richard Cullingford and Riesbeck in 1975.
Schank goes into great detail of what "understanding" means. To clarify the level of understanding, systems build upon his theory have, the following characteristics are given below.

[2] Systemic functional grammar (SFG) is a model of grammar developed by Michael Halliday, see (Halliday, (1976).

The system is able to:

- create a linked causal chain of conceptualizations that represent what took place in a story (a paragraph of written text).
- make inferences from the created concepts
- turn created concepts back into text in any language. (paraphrasing)

Since the programs use background knowledge the following is possible with the systems:

- Inferences can be made which are specifically mentioned from the given text.

In order to encode background knowledge of a particular context, Schank invented the idea of using "*scripts*". A *script* is a structure that describes appropriate sequences of events. *Scripts* are used if a situation has a stereotyped sequence of action. Stereotype sequences are situations that are a well known series of events. For instance in the context of a customer going shopping the following *script* could be used:

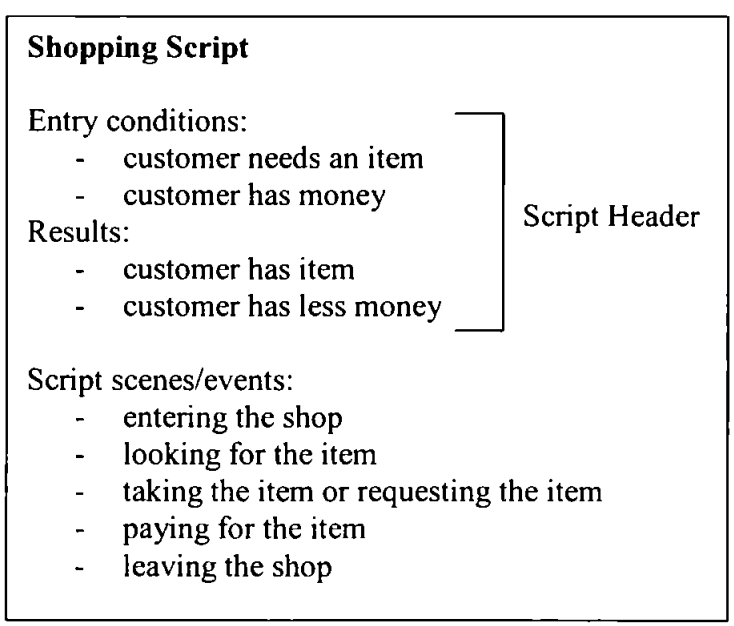


Figure 2-8: example of a script

Script items are first hypotheses of events that are going to happen in a particular situation.

The events are in an order, one event happens after another. Schank calls this a "*causal chain*". As the natural language text is processed script events are instantiated with values - a kind of slot filling. If an event happens it can enable the occurrence of another

event. In the example above: If the customer has taken an item off the shelf then the event "paying for the item" is enabled. Since a customer in a shop can only pay if there are items he/she wishes to pay for.

Unfortunately scripts only work for stereotypical situations; therefore they are by no means the answer to how to understand natural language text. Like the title of Schank's book says "Scripts, Plans, Goals and Understanding" (Schank and Abelson 1977), there are three theoretical entities necessary, namely Scripts, Plans and Goals to understand natural language.

2.4.4.1 Plans & Goals

If there is no script available, there needs to be a method of understanding a text. The first thing to do then is to identify the main "*goal*" of the entities in the text. Suppose the text starts with "John is hungry" then the goal of John is to find food. There might be several sub-goals that are identified during the processing of the text, such as going to a location where food can be found.

If a *goal* can be identified then the computer is able to:

- make prediction what might happen
- build up a script on how to achieve the goal by following the text
- put the text and word meanings in the right context (not specifically mentioned in Schank's book)

To deal with situations, that are not available as scripts, mechanisms (conceptualisations) that underlie the normal scripts must be accessed. Any conceptualizations that are instantiated must be placed so that it is possible to trace a path between them. The path is called a "*plan*". Although Schank's scripts, plans and goals idea lacks flexibility, it may be the most practical approach since a service robot is confined to a limited set of skills. Especially if a practical/commercial service robot with natural language interface would be build at present or in the near future it would most likely use a script based learning approach. Its practical nature makes it so attractive, and commercially feasible, a further reason to consider here that hopefully brings service robots closer to reality.

The programs for natural language understanding (NLU) developed by them make use of *conceptual dependency theory*. However, the inventor of *conceptual dependency theory* John Sowa argues that the implementations that Schank's research group used, does not explore the full potential of conceptual dependency (Mann 1995). For example, a word is assigned to a single meaning or word-sense where a word could have multiple meanings.

2.5 Natural Language Understanding Systems with Speech Recognition

This section reviews speech recognition architecture and tools required for natural language understanding systems with speech recognition.

2.5.1 Spoken vs. Written Language

Spoken language is different to written language. This has to be taken into account. Spoken language is more spontaneous and instant. It has a looser construction and unnecessary repetition. Often the speaker is rephrasing and stops in the middle of a sentence (Crystal 1997). On the other hand spoken language is part of a conversation, and the other parties can communicate to ask clarification questions immediately. The grammar of spoken language is different from written language, and if natural language grammar and parsers are applied they must therefore be built for spoken language. The use of formal grammar for written English was a major limitation in the IBL project. Only 60% of the corpus was covered by the grammar (Bugmann 2003).

2.5.2 Architecture

A typical Natural Language Processing System is organised in a Pipeline Architecture. The components are organised in parts that are not necessarily from the same software package. The components in order of the information flow in the pipeline are typically: speech analysis, morphological and lexical analysis, parsing, contextual reasoning, application. And from the application the pipeline can go back to speech synthesis in a similar fashion by going through utterance planning, syntactic and morphological steps to speech synthesis. Some examples are GATE (Cunningham *et al.*, 1997), NUANCE 8 (Nuance App. Dev. (2005)) or the open source Natural language toolkit NLTK.

The pipeline architecture of natural language processing systems has been under criticism see (Graça *et al.*, 2006; Marciniak and Strube 2005; Leidner 2003; Daelemans and van den Bosch 1998), however it is still the most common structure since it is the best method to implement a natural language system from a software engineering point of view. Also the natural language system introduced in this PhD work will use the pipeline architecture. There is no escape from it. The criticism is mainly aimed at the

problems introduced by the possibly independent language tools that are used in a chain. It is hard to give feedback to a previous stage and due to the transformations from stage to stage information may be lost and errors may therefore be introduced. One of these pipeline tools is typically a syntactic parser that parses recognised text. These parsers are often not trained on the specific context of the domain and therefore introduce errors.

2.5.3 Hidden Markov Models

Modern speech recognition systems utilise Hidden Markov Models (HMM) to recognise phonemes, words and phrases in a multi-layered model (Cunningham 2000). HMMs, in the context of speech recognition, are statistical models of how likely a word follows another. Or at a lower level, which phoneme or acoustic feature most likely follows another. A common way to extract acoustic features is by using Fast-Fourier Transforms. The acoustic features and the probably of their occurrence are an inheritably sub-symbolic (statistical) process. For a good introduction see (Rabiner 1989). However these models use symbolic building blocks: phonemes, words and phrases. Their relations are expressed as grammar. The HMM returns the most likely interpretation (with the highest overall probability in the markov chain). By accepting this as the interpretation text, the sub-symbolic audio data has become a text.

2.5.4 Interpretation

In case of natural language understanding, where the emphasis is on understanding, the text alone is not sufficient. The concept of "understanding" puts the text to a meaning, a relation that the robot can reason with, and particularly important, the concept of "understanding" means that the text and relations the robot reasons with are connected in the robots action and perception. Therefore the grammar is connected to an interpretation (called interpretation grammar (Nuance App. Dev. (2005)), slot filling or semantic grammar (Rosner and Johnson (1992)), which is usually expressed as an attachment to a grammar rule. Grammar acts as the defining language to connect speech to semantic interpretation.

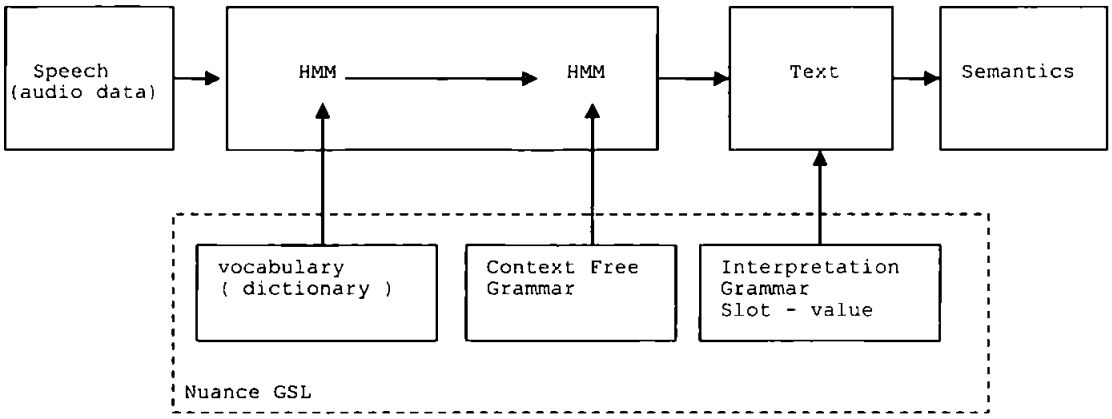


Figure 2-9: A typical Natural Language Interpretation system uses grammar to define multi-layered HMM models from phonemes to words and words to sentences. These models serve as mapping between Speech and text. Traditionally the text output is parsed (syntactic analysis) by a interpretation grammar to determine the meaning of the text

2.5.5 Grammar

The Nuance speech recognition system, used in this project, combines the CFG (context free grammar) and the interpretation grammar into one. Every CFG grammar rule can have slot-and-value semantics attached to it. Parsing recognised text to extract an interpretation has been widely criticised for the same reason as the pipelining architecture, because the grounding of the interpretation is disconnected from the text and speech recognition that are preceding in the pipeline. In practice this means that text is recognised that the robot cannot understand because it does not make sense. One of the problems with IBL was that it recognises "turn the tree", which is correct in English but does not make sense. It was introduced by generalising a CFG from sentences like "pass the tree" and "turn left".

The CFG grammar in this case is:

```
S   → V NP
S   → V ADJ
NP  → DET N
DET → the
N   → tree | corner
V   → turn | pass
ADJ → left
```

Disconnecting meaning from grammar, such as in this case, produces unwanted overgeneration. A correct syntax does not always lead to sentences meaningful within the domain of correspondence of the robot.

Chapter 6 describes how the combination of CFG and interpretation is used as an advantage to improve speech recognition. In a nutshell, the combination allows the prevention of unwanted generalisation by abstracting syntax rules from the corpus.

2.6 Semantic Representation Theories

2.6.1 Semiotic Schemas

In an effort to create a non-symbolic (computational) system that can make the connection to symbols, Deb Roy from MIT created a framework (Roy 2005), which is outlined here. The framework for semiotic schemas is built upon creating a meaning from sensor data and motor acts. It is therefore a so called bottom-up approach to machine learning systems. Every piece of knowledge stored in the robots "brain" can be referred back to the physical world through sensor data and motor acts. It is a grounded system. The knowledge can also be used to make predictions about the future and compare these to actual sensations.

This is called an *analog belief*. So sensors are mapped to *analog beliefs*. See the notation below in figure 2-10.

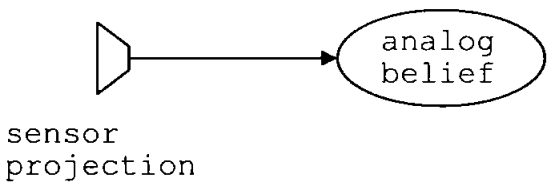


Figure 2-10: a sensor (natural sign) is monitored to create an "average" belief state

One may wonder how this "analogue" statistical distributions can be put into categories. Deb Roy introduces categorizers as a link between analog beliefs and discrete *categorical beliefs*. In the graphical notation analog beliefs are oval and categorical beliefs are rectangular.

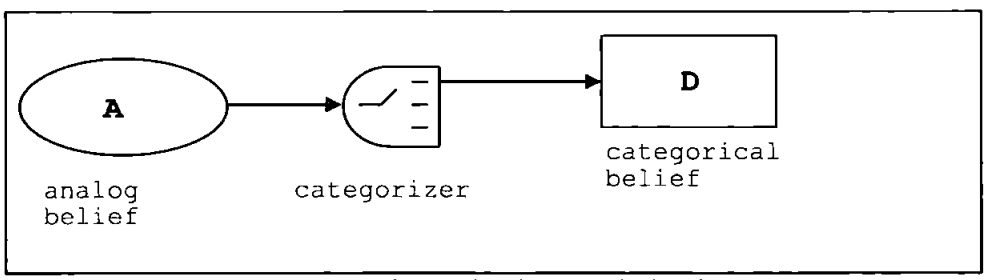


Figure 2-11: a categorizer makes discrete decisions based on an analog belief

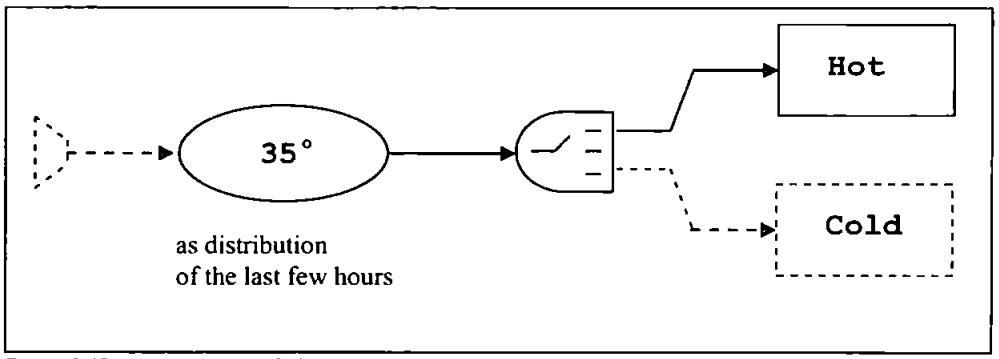


Figure 2-12: a robot that can feel temperature and belief if it is hot or cold

Deb Roy implemented this framework into a robot called Ripley, which has a 7 degrees of freedom arm, vision system and a speech interface. The robot was designed for grounded language experiments (Roy *et al.*, 2004). The framework is an attempt to connect the symbolic world of language to the non-symbolic world of sensors and actuators. Roy argues that not-grounded systems would need a human in the loop during design and implementation to connect sensor data to a representation system in the robot, whereas his approach enables statistical mapping between the sensors/actuators and the introduced symbols. The framework of semiotic schemas is used as an inspiration to this work. The most relevant concept for here is the idea that physically grounded analogue data can be converted to symbolic categories. Categorical believes could be used to represent locations of cards and the recognised actions/gestures. This allows symbolic processing and the integration of language into the robots advanced reasoning system, even though the robots low level AI (skill-learning and pattern recognition) is subsymbolic.

2.6.2 Conceptual Graphs

Conceptual Graphs (CG) are related to semantic networks. They were invented by John F. Sowa. Conceptual Graphs can represent concepts and their relationships. They are a powerful tool to create a knowledge base. CGs have the following useful properties: They are human readable (hence they can be turned into natural language expressions. They can be created from natural language expressions. They can be turned into predicate logic statements (with certain constrains).Conceptual graphs are best explained by an example. Below an example of the sentence:

"John is going to Boston by bus" taken from (Sowa J.F. website)

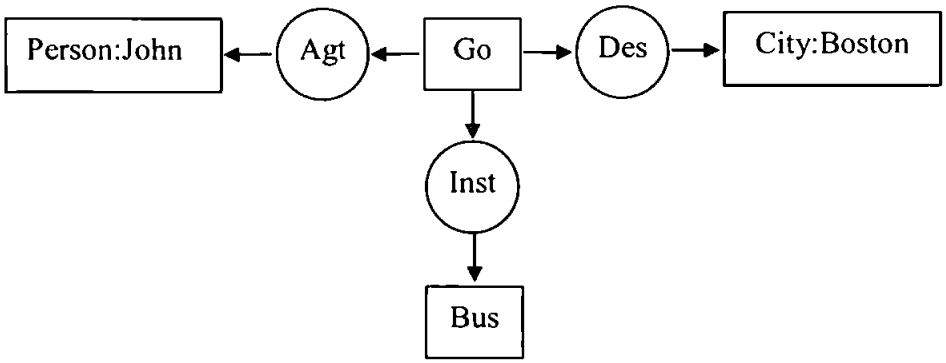


Figure 2-13: "John is going to Boston by bus". The square boxes indicate concepts and the circles indicate relations. Note that the concept Person has a referent "John" while the instantiation (referent) of the Bus is unknown.

In IBL and MIBL a primitive function can be defined to match this example. The primitive itself is "Go" and its parameters are Agent, Destination, and Instrument.

```
go ( Agent, Destination , Instrument )

Whereby the allowed word classes could be:
- Agent of the word class Person
- Destination of the word class City
- Instrument of the word class vehicle
```

A concept always has a Type and can have a Referent. A referent is a particular object/concept. The Type must be based on ontology. (Ontology is a tree of types starting with the most general type at the top). A concept can either stand alone or be connected to a relation. It is not allowed to connect two relations directly with each other.

```
[Type: Referent] <-(Relation)-> [Type: Referent]
```

A single concept may be: `[Bus]` Which means "There is a bus".

```
[Proposition:
  [Woman: *x]->(Attr)->[Beautiful]
]
```

"There exists a woman x who is beautiful."

Language can be mapped into a conceptual representation using a conceptual parser. The conceptual representation is a representation of the dependency of the parsed text.

In this PhD project, conceptual graphs have been a useful representation to clarify the structure of sentences and to extract an ontology design of the domain. This clarification enables the system designer to map sentences into logic

2.6.3 Frame-based systems

Frame-based systems are knowledge representation systems that use frames. Frames combine domain knowledge in a structure for representing a stereotypical concept or a situation. A frame can have several kinds of information attached that describe the concept or situation further. The first to recognise the ongoing common trend in the 70s to represent knowledge in frames was Minsky (1975).

A Knowledge representation system that was inspired by Minsky's ideas is KRL (Bobrow and Winograd 1977). Goldstein and Roberts (1977) in turn were inspired by KRL when writing their frame-based system NUDGE. In their paper Goldstein calls the frames Frame Gestalts. The reference to "Gestalts" comes from the Wertheimers Theory of mind (Wertheimer 1923), that has fit in very well with frame-based systems.

Fikes and Kehler (1985) explored what is common in frame-based systems, quoted here:

- frames are organized in (tangled) hierarchies
- frames are composed out of slots (attributes) for which fillers (scalar values, references to other frames or procedures) have to be specified or computed
- properties (fillers, restriction on fillers, etc.) are inherited from superframes to subframes in the hierarchy according to some inheritance strategy.

This structures are remarkably similar to object oriented programming principles of C++ and Java. To some extend also SQL. Object oriented programming probably have roots in these systems.

Schank's Dependency Theory and causal chain, described in Chapter 2.4.4 earlier and in Schank (1975) is also a frame-based system.

2.6.4 Ontolological reasoning

Organised information is the key to deduction and reasoning. A list of nouns has no meaning unless the relationship between them is given. Aristotle was one of the first to

recognise the power of logic and organisation of information so it can be used for logic deduction (Aristotle, transl. 1989). A famous example of Syllogism, the logic that Aristotle defined, is:

"All men are mortal; Socrates is a man; therefore, Socrates is mortal."

A further advantage of organising information is the possibility to constrain the grammar to produce only rules that not only are grammatically correct, but also make sense. In section 6.2.2 the concept of word-classes has been introduced. As a reminder, word-classes are a group of words that belong semantically into the same category. For example the word-class "colour" has "blue, green, red..".

These word-classes are also used as primitive parameters. Section 6.2.3 on full corpus coverage shows in figure 6-2 how a corpus utterance that has become a grammar rule is extended with a word-class.

It is of advantage for consistence in reasoning and the search for information in the knowledge base, to combine all word-classes to a complete model that contains all concepts that the robot is dealing with in the domain. This model of word-classes is best organised in a hierarchical taxonomy, since a semantic category is often part of another more general category, sometimes referred to as superclass. Such taxonomy is conveniently represented by a tree.

Scientists have been studying on the structure of such taxonomies and their applications since the great philosophers Plato and his student Aristotle. These structures are often referred to as semantic networks or ontologies. The science of "ontology" is concerned with finding ways to structure taxonomies and how to apply these structures. Sowa, whose work has been briefly introduced in chapter 2.6.2, presented a "global" ontology which is at the top-level and every concept can be derived from it, see (Sowa 2005). His conceptual graphs are grounded in this ontology. It is the concern of ontology researchers to build top-level ontologies that capture very general concepts so they can be expanded to every possible domain. It is a philosophical question, how such a top-level ontology may be organised. From a Corpus Based Robotics point of view it is not necessary to cover more than what is found in the corpus, which simplifies the problem.

Ontologies guide the generalisation process for grammar and primitive parameters, the reflection of the ontology in the knowledge base gives the robot the ability to generalise concepts. For example it can infer that a "queen of spades" is a "card", or it can compare concepts with each other, when searching for suitable objects. See section 7.8 on problem solver, and specifically 7.8.4 on generalisation and references. This is important since referents in natural language are often underspecified, but referents have to be resolve within the rule frame.

One may wonder if existing ontologies could be utilised in an application. Unfortunately it is not a straight forward process to select the right meaning from existing ontologies, in the given context. For example, WordNet 3.0 (Miller 1985) defines a "card" in many ways, such as a calling card, a circuit board or and identity card. It is difficult for a system to reason with WordNet, since the class "card" has so many meanings in different contexts. WordNet also defines "playing card". WordNet attaches "suit" to "playing card", which is correct, but fails to connect "hearts", "spades", etc as semantic classes under "suit".

The created ontology becomes a world model of the robot. To create a complete world model, not only concepts (word-classes from section 6.2.2) are required. Also instances of objects are required. For example, Aristotle is an instance of a human. Furthermore he therefore "inherited" all the properties of humans, such as being mortal. In case of the MIBL projects, the robot has in his world model, an ontology of 3D objects, which can be manipulated. Further down these 3D objects are cards. Instances of cards are stored in the knowledge base.
What is in philosophy an ontology reminds a computer scientist of object-oriented programming. In fact, knowledge of physical objects and their properties are stored by the robot in an object-oriented format.

2.6.5 Newell and Simon General Problem Solver

Newell and Simon were the first to implement the idea of problem as a program. Their first program, the "Logic Theorist" was presented at the Dartmouth Summer Research Conference in 1955 (Newell and Simon 1956). Later an extended version, that separated

the problem definition from the solver was called GPS, the General Problem Solver (Newell and Simon 1972).

Typical problem solvers have need several critical steps: the definition of the problem space in terms of the goal to be achieved and the transformation rules. Simple problem solvers would use the means-end-analysis approach, to divide the overall goal into subgoals and attempt to solve each of those. Some of the basic solution rules include: transforming one object into another, reducing the difference between two objects, and applying an operator to an object. A table that specified what transformations were possible is required.

Given a robot that can specify its environment as states and actions on the environment as state transitions, a problem solver algorithm can be applied. A problem solver is a search algorithm that applies production rules (state transition rules) to manipulate a given state until a target state (goal) has been reached. The applied production rules can be stored as a solution path to the goal. Given that the robot knows the consequence of each action then state transition rules can be applied to its memory instead of carrying out the action immediately. Hence a problem solver is also a planner.

2.6.6 Lambda Calculus

The lambda calculus is a notation for mathematical expressions and functions. It was rediscovered as a versatile tool in computer science. The syntax of the computer language Lisp was inspired by the lambda calculus. The lambda notation requires operators, to be written before the parameters (prefix), like Polish notation.

For example expressions "x + 3" becomes "+ x 3", and "x^2" becomes "* x x".

LOGICAL OPERATORS OF THE LAMBDA CALCULUS
$\wedge$ conjunction (AND)
$\vee$ disjunction (OR)
$\neg$ negation (NOT)
$\Rightarrow$ implication
$\Leftrightarrow$ equivalence
$\exists$ existential quantification
θ universal quantification
$\approx$ equality

Table 2-1: logical operators in the lambda calculus

Functions:

The λ notes the function

$f(x) = 3x$ in lambda-calculus becomes $\lambda x.*3x$

The lambda calculus is used in natural language understanding to describe dependencies between words. A natural language expression, i.e. a sentence can be converted into a logical formula. Usually this starts with determining the parts of speech and using a syntactic parser. The resulting structure of noun (N), noun phrase (NP), verb phrase (VP), etc shows dependencies between the words. These dependencies can be expressed in a formal way with the lambda calculus. For instance consider

"John believes something is false"

Expressed in lambda calculus:

$\exists x(x \in L \wedge bel(John, x) \wedge false(x))$.

A translation can be defined from the lexical words, such as "believe" into the semantics "bel(y,x)". Like Discourse Representation Structures (DRS), the lambda calculus can represent dependencies and is an intermediate step. Lambda calculus expressions are logical and can be used for inference and reference resolution.

In order to learn and carry out instructions a robot must use inference or some form of mapping to extract the instruction from an expression in lambda calculus. For this process, background knowledge is required. This can be especially difficult for colloquial expressions, such as "what's up?".

3. Corpus Collection

Linguistic corpus collection is defined by acquiring and storing a corpus (Latin for body) of example dialogues usually by recording and transcribing or by gathering existing texts. In this chapter an experiment will be described where conversations between two people are collected to form the MIBL corpus. Corpora are not restricted to spoken and written text; they can include transcriptions of any interaction data, such as hand gestures, eye movements or a mouse cursor. Combinations of any of the listed modalities are collected in so called multi-modal corpora (Baldry and Thibault, 2006). Collecting a corpus is the first step when applying the corpus-based robotics approach. In order to create the corpus, the recordings are transcribed using the multi-modal transcription tool MuTra (described in section 3.4.1). The transcriptions include start time and duration of gesture and speech. The corpus provides a starting point to create a speech recognition grammar. The transcription process could be simplified by adding speech recognition software. However, all transcribed text has to be confirmed manually since the corpus provides the reference data for speech recognition and all further system development.

The corpus is to be analysed in order to design an agent that will be able to interact and perform actions that are found in the corpus.

3.1 The Instruction Domain

3.1.1 Experimental Constrains

With the experience and motivation from the previous project (IBL), criteria for the selection of a new application domain were determined. The criteria are aimed at investigating and extending IBL by applying scientific method[3].

i) The task must contain a wide range of instruction types. (rules, sequences , repetitions). So they can be investigated.

ii) Ideally the task should be scalable from simple to complex. So a range of complexity can be investigated.

iii) The task should preferably have a small vocabulary (less than unique 1000 words). So the transcription and implementation is manageable.

iv) The task must be part of the natural environment of the instructor (user) so that instructor and student already posses the basic skills required.

v) The task should contain meaningful set of gestures / actions (multi-modal).

vi) The task should be unknown to the subjects beforehand, to set up a genuine teaching scenario.

vii) The domain size should be predictable. This can be achieved by measuring the rate of discovery of unique ways of expressing an instruction.

The constraints mentioned above, especially point *vi)* have to be determined by a pilot study. Furthermore a pilot study is a required step in the application of corpus-based robotics. A pilot study would require a corpus collection, transcription, search for language primitives and their types using 3-5 subjects.

Generally, there are many ways of expressing a verbal instruction, even in a restrained domain. Restrictions *iii)* and *vii)* are there to harness these restrains.

Given these constraints, game instruction seemed to be a good choice. In particular, card games come in a great variety of type and complexity, yet their vocabulary is restricted. All two player games listed in "the Oxford A-Z of Card Games" (Parlett, 2004), 23 different card games were investigated by counting the number of instructions in the form of a clause or sentence. It should be noted that the instructions from a professional book are more compact than verbal explanations of the same rule. As an

[3] The scientific method: hypothesis, experiment, observations, tests, confirmed theory

example, instruction sets from table 3-1 would count as one instructions each, per sentence. From the investigation, it was found that a typical card game has on average 38 instructions, with a std. deviation of 17.46.

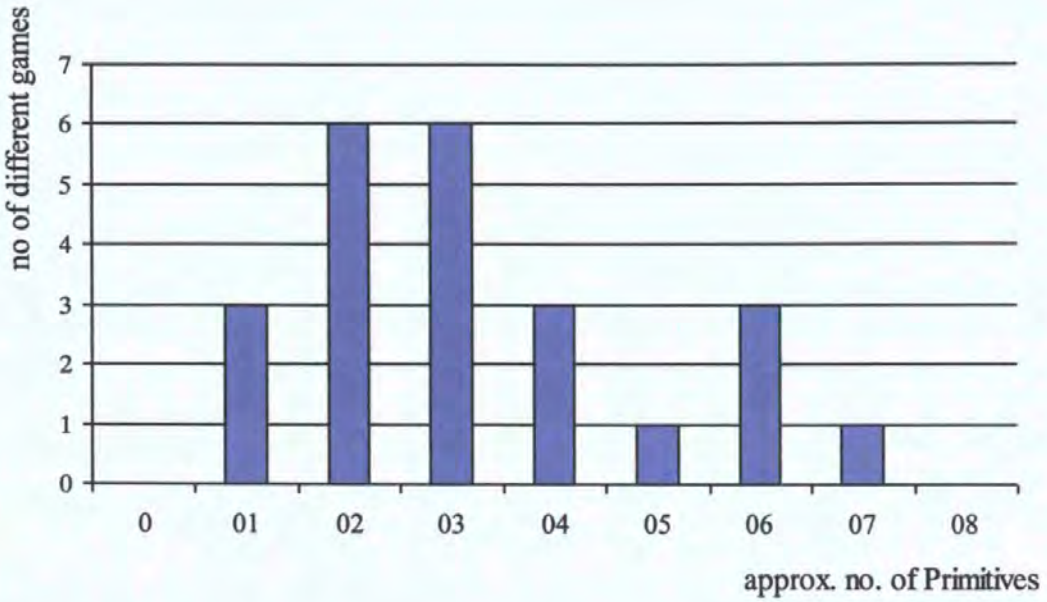


Figure 3-1: Game Primitives: A survey of 23 card games and their number of instructions presented as a frequency distribution showed that a typical game has an average of 38 instructions. See the list of games in Appendix A7.

In order to create a robot that would appear to be a reasonably intelligent card game player it was decided to choose a card game that has between 30 and 40 instructions.

3.1.2 Scopa

The Italian card game Scopa was chosen, since it had 35 instructions and is virtually unknown in the United Kingdom. That the card game is initially unknown is important as the investigation is about teaching. Yet all basic skills such as dealing a card or comparing cards are generally known to UK residents.

In Scopa, initially 4 cards a laid out face up on the table. Another 3 cards are dealt to each player. Scopa is a fishing-type card game (Parlett, 2004). A fishing game means that there are several cards face-up on the table, and the players have to match cards in their hand with the cards on the table. Matching cards on the table can be captured by the player in order to score. The game was originally played with a deck of traditional Italian cards. For the French deck (most common deck) the eight nine and tens have to be removed from the deck. Instead jack, queen and king are worth 8, 9 and 10 points respectively.

3.2 Procedure of Corpus Design and Corpus Collection

Multi-Modal corpora are still rare and contain very specific data; in particular there was no multi-modal card game corpus publicly available that would be suitable. As such we decided to setup an experiment to collect a multi-modal corpus.

3.2.1 Corpus Design

Corpus Design is concerned with decisions such as: how many subjects will be interviewed, what data will be recorded and how the data is formatted. Corpus Design decisions must be carefully considered with respect to the research that will be carried out with the resulting data. In a scenario where corpus-based robotics is applied, rather than researched, the domain is given initially. For example a company requires a vacuum-cleaning robot that can be naturally instructed. In the case of researching corpus-based robotics (specifically the MIBL project), the aim is to design the corpus to cover an as large as possible variety of features in the vocabulary and langue primitive types.

The corpus designer must consider that the collected data is stored in a format that can be used later for testing the performance of the developed system. Furthermore the data must be collected using the sensors on the robot, from the robots point of view. It is not advisable, for example, to use overhead cameras if the final robotic system will not have overhead cameras. A Wizard of Oz experiment is a proven way to collect a corpus. (Dahlbäck *et al.*, 1993, Kyriacou 2004).

In a teacher student scenario, the teacher, who knows how to play Scopa, will explain the game to a student. After some practice the student can now become a teacher to explain the game to another student.

An informal pilot study has been carried out that consisted of 5 teaching sessions and their transcription of a very simple card game. It revealed that a teacher subject tends to use verb phrases and methods similar to those used when he/she was taught. In each dialogue the vocabulary and explanation techniques changed. Only some utterances of the dialogue were taught exactly like the teacher learned it. This led to the assumption that a longer teacher-student chain contributes to a larger variety in the corpus afterwards. Some teachers, however, may not come back to teach the game to another

subject, which would break the chain. To increase the chances of a longer chain a teacher was invited to teach two students in separate sessions. See figure 3-2.

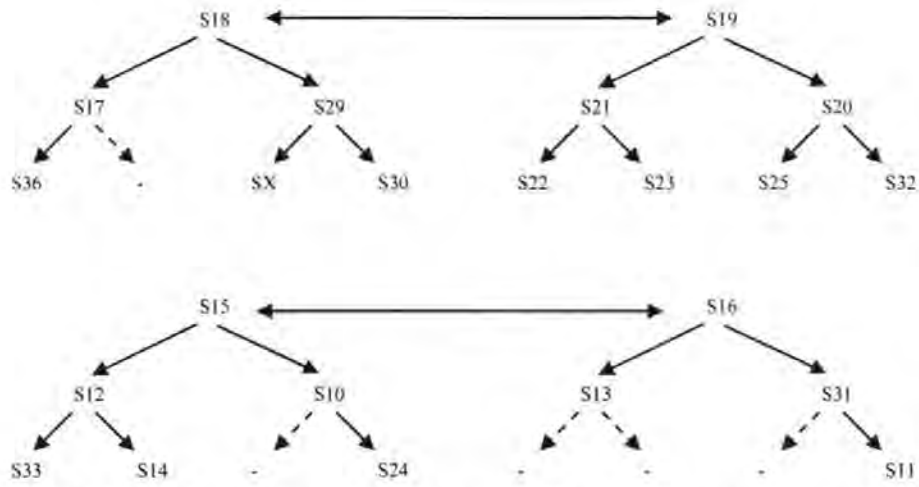


Figure 3-2: Tree of teaching dialogues. Two trees of this type were used to record dialogues. There are 6 dialogues in each tree, represented by the arrows and organized in three layers. Si is the subject number i. If one of the two subjects failed to attend, the chain was broken, here shown with a dashed dialogue line. These two trees are from now on refered to as data set-1 and data set-2.

It was also assumed that a longer break between learning and teaching the game reduced the similarity in the vocabulary used.

Initially two teachers had to be made familiar with the game. To avoid a bias as much as possible they were given two complete sets of written instructions of the game (Seed Set1 and Seed Set 2). Each instruction was written on a separate paper. The set was mixed so that the rules did not appear in a particular order. The two teachers studied the rules sets quietly for a few minutes. They re-ordered the sheets to help learning the game. Then they were invited to try to play the game and to clarify the set of rules by communicating with each other. This communication has been recorded with the experimental setup, but has not been used for any further analysis. Subject S18 was given Seed Set 2 and clarified rules with subject S19 who was given Seed Set 1. A further two teachers have been invited to do the same experiment whereby teacher S15 was given Seed Set 1 and subject S16 Seed Set 2.

Below is a list of instructions to the teachers.

INSTRUCTIONS TO THE TEACHERS
Instruction Seed Set **1**
Two players use a 40-card pack running A234567JQK in each suit.
Deal three cards each in ones, face down. And then four face up to the table.
When everyone has played their three cards, deal three more each from stock. Continue until all cards have been used and captured.
Each in turn must play a card from hand with a view to capturing one or more table cards.
Table cards may be captured by pairing or summing.
Pairing: An Ace takes an Ace, a Two a Two, and so on. Only one card may be paired in one turn, and if the hand-card can capture in either way it must do so by pairing.
Summing. A hand-card takes two or more table cards totalling the same as itself. For this purpose, cards count at face value from Ace 1 to Seven 7, followed by Jack 8, Queen 9, King 10. Thus a Seven will capture two or more cards totalling 7 (A+6, 2+2+3, etc).
When summing: Only one such combination may be made at a time
When you make a capture you place both the captured and the capturing cards in front of you and end your turn.
If you capture all the cards on the table, leaving none for the next player to take, it is a sweep. You indicate this by leaving the capturing card face down in your winnings pile, and will score 1 point for it at end of play.
You must play a capturing card if you can. If not, you must 'trail' by playing any card face up to the table and leaving it there. This is inevitable after a sweep.
When no cards remain in stock, the last player to make a capture (not necessarily the last to play, since he may be forced to trail) takes all the other table cards with it. This does not count as a sweep, even if, technically, it happens to be one.
Players sort through their won cards and score as follows: 1 point for taking the most cards. If tied, no one scores. 1 point for taking the most diamonds. If tied, no one scores. 1 point per sweep, as indicated by face-down cards.
The winner is the player with the highest score at the end

Table 3-1: Instructions to the teachers, Set 1

INSTRUCTIONS TO THE TEACHERS
Instruction Seed Set **2**
A 40-card pack is used. A234567JQK in each suit. The game is played with two players.
Deal three cards for each players hand. Don't show them to your opponent. After, deal four on to the table. (face up)
When players don't have any cards left in their hand, deal three more each from stock. The game ends when the stock has been used up and all cards have been captured.
Each player, in turn, plays a card from hand. The target is to capture one or more of the cards on the table.
There are two ways of capturing cards from the table: pairing and summing.
Pairing means a card in your hand pairs with a card on the table and you can take them to your stock. You must do pairing if you can.
Summing. A single card in your hand card can take multiple table cards which have as a sum the same value as the card in your hand. Cards count at face value from Ace 1 to Seven 7, followed by Jack 8, Queen 9, King 10. For example a six will capture two or more cards totalling 6 (A+5, 2+A+3, etc), and so on.
When summing: Only one hand-card can be used in one turn.
When you make a capture you place the involved cards in front of you onto your own pile.
If you capture all the cards on the table at once by summing, it is a sweep. You indicate this by leaving the capturing card face down in your winnings pile, and will score 1 point for it at end of play.
You must play a capturing card if you can. If not, you must put down any card face up anywhere onto the table and leave it there. Basically every player gets rid of one card every turn.

When no cards remain in stock, the last player to make a capture (not necessarily the last to play, since he may be forced to just add a card to the table from his hand) takes all the other table cards with it. This does not count as a sweep, even if, technically, it happens to be one.
After the game players sort through their won cards. Points can be scored as follows: 1 for taking the most cards. If tied, no one scores. 1 for taking the most diamonds. If tied,no one scores. 1 per sweep, as indicated by face-down cards in your pile.
The winner is the player with the highest score at the end

Table 3-2: Instructions to the teachers, Set 2

The invited subjects were mostly university students between the age of 20 – 30 years, only one female person. A significant number were not native English speakers. The pilot experiment and the final online experiments (chapter 8) had a similar distribution of age, gender and occupation.

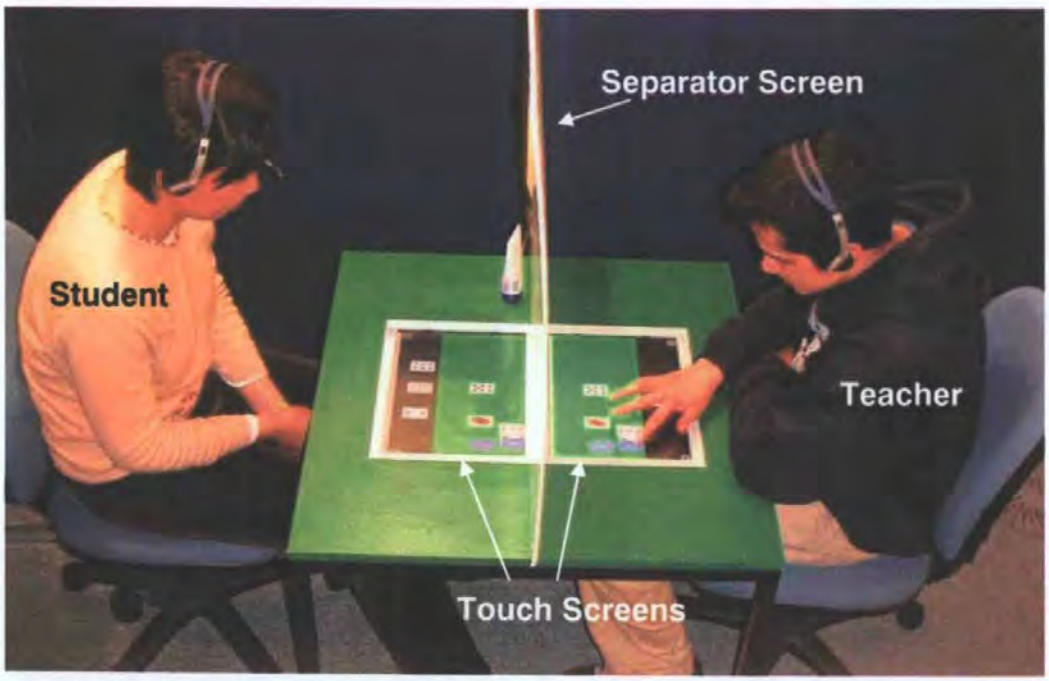


Figure 3-3: Experimental Setup for Corpus collection

The recordings if they are to be used later to test speech recognition must be of a good quality. The subject wore cost effective Plantronics headsets to improve the sound quality while recording. Each subject was recorded in uncompressed 16bit PCM WAVE Stereo format. Later experiments were carried out with an external Roland/Edirol USB soundcard to further reduce the signal to noise ratio.

3.2.2 Data Sets

DIVISION OF THE CORPUS INTO DATA SETS

Name	ID numbers of the experiment session
Data Set 1	03,06,07,10,11,12,14,19,20,21
Data Set 2	04,05,08,09,13,15,16,17,18

Table 3-3: Data sets of transcriptions

3.3 Multi-Modal Interface

3.3.1 A Robot with simulated Eyes and Arms

As argued in section 1.1, a truly natural unconstrained human-robot communication interface must be multi-modal. In future robots, multi-modal interfaces will require complex sensory processing, such as gesture and face recognition. As this project focuses on the problem of task learning, it was decided to devise a simplified interface that would still allow natural communication with human users, but simplify gesture recognition and robot actions.

The solution to the problem is to use a touch screen that allows at the same time to acquire human gesture information by the robot (without complex sensory processing) and execution of game moves (without complex actuators). The screen represents the world as the robot would see it through its vision system. The user is able to point at and manipulate objects on the screen as a demonstration of how to do the task. At the same time the user gives verbal instructions. Touch-screens have been used in multimodal human-robot interfaces for different applications, for example by (Perzanowski *et al.*, 2001), or for investigations in human communication (De Ruiter *et al.*, 2003).
A great advantage of using a screen representing the robot's world is that the robot can be simulated (a software agent), while the interaction and interface to the robot does not change. It also allows focusing research on human-robot interfaces without having to build a robot first.

Figure 3-4: A person is dragging a playing card on a touch screen. Gesture (Action) recognition through the touch screen is translated to movements of the card in the virtual 3D environment.

The robot is not embodied by an arm or face on the screen. The cards appear to move "magically" on the screen if the robot is acting. The robots voice can be heard through

the speakers. The same applies if two people interact, their voice can be heard but they can not see each other. If one moves a card it moves on the other screen as well.

A finished embodied robot could still have an attached touch screen mounted representing an Augmented Reality (AR).

3.3.2 Potential as Human-Computer Interaction interface

The investigation of human-robot instructions, in this project, is carried out with speech recognition and a touch screen interface. Essentially this reduces the human-robot interface to a human-computer interface. Human-computer interaction has a wider research and user community than human-robot interaction.
Touch screen interfaces are currently gaining popularity in the form of mobile phones, multi-media players and PDAs (Personal Digital Assistants).

3.3.3 Simlator Software

This section describes the simulator that simulates the environment and lets the users interact with it through multiple displays. The software developed to display the playing cards is based on the Qt (Trolltech®[4] 2005) and the OpenGL® API[2] and is platform independent. Qt is a cross-platform C++ GUI development library. OpenGL® is a standard for a 3D/2D cross-platform Graphics API. The playing cards are described as objects with parameters such as size, texture, position, orientation, static or movable. Therefore the system can be used not only for card games. The display software could display the image of any real world object. The user can manipulate these objects intuitively. The computers used for displaying the cards are linked to a server via TCP/IP. Performance is increased by having a computer for each Display Client due to the computational load for rendering the 3D environment. The server initially sends the display-clients the objects that need to be shown on the screen.

---

[4] *Qt is a trademark of Trolltech in Norway and other countries.*
http://www.trolltech.com/

[2] *OpenGL® and the oval logo are trademarks or registered trademarks of Silicon Graphics, Inc. in the United States and/or other countries worldwide.*

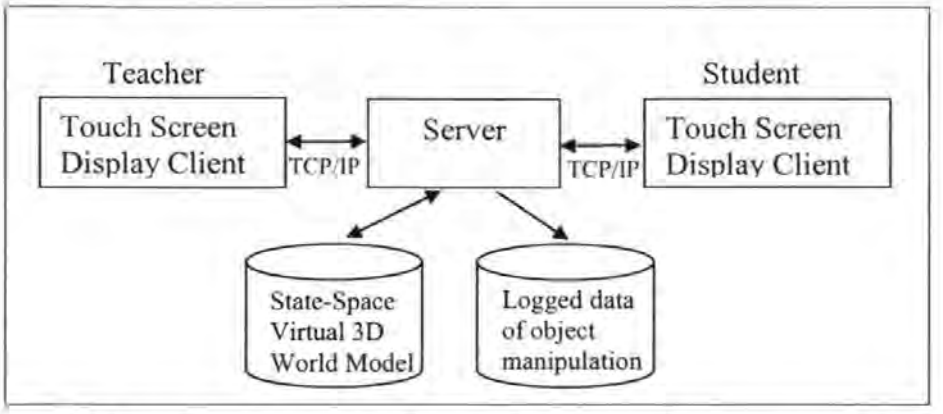


Figure 3-5: Corpus recording with two touch screens. If an object is manipulated in one screen it is also visible in the other, since the data is forwarded by the server. All data is logged and can be replayed for transcription.

All events of object manipulations are logged at the server (figure 3-5) and forwarded to all other connected clients. So if objects are moved on one screen, they move on the other screens as well. All events are also represented in a internal state-space world model (see section 3.3.4).

3.3.3.1 Interaction Control Protocol

A simple human-readable TCP/IP protocol called ICP (Interaction Control Protocol) has been created to communicate object creation and manipulation between server and client. Every packet is logged by the server into a human readable CSV (Comma Separated Value) file in the same format as it was sent or received by the server, with a time-stamp added in 10 Hz resolution. More than one event can be logged and forwarded within the 1/10 of a second time period; however the timestamp of these events would be the same. A 10 Hz sampling rate was found sufficient for the gesture recognition of this object manipulation task and has been used in related systems (Rybski. and Voyles, 1999; Davis and Shah, 1994; Oviatt *et al.*, 1997). The 10 Hz timer for generating timestamps was synchronised with the real-time clock (RTC) of the server PC to maintain accuracy during long term recordings.

Initially the server accepts connections from the clients. In this implementation there are only two connections allowed, one teacher client and one student client. After the TCP/IP connection is established the client must report if it is a student or teacher terminal. The server will reply by sending the 3D world offset to place the client on the right side of the virtual table. After all connections are established the researcher will initiate the 3D objects (table and the cards) to be sent to the clients. Every 3D object is

assigned a unique ID at creation for referencing. Upon the first creation of the objects, the server software emitted an audible ring sound which can be used later to synchronise audio recordings with the timestamp of the gesture log file. The human-readable protocol format allows easy and transparent analysis of the communication system. Table 3-4 shows the protocol between server and display clients.

PROTOCOL BETWEEN SERVER AND DISPLAY CLIENTS

Name	Direction (C)lient to (S)erver	**Command,** Parameters Description
Ready	C→S	***R,*** *ClientName* Client Reports that it is ready, followed by ClientName which is either "Teacher" or "Student"
Offset	S→C	***O,*** *Offset_X, Offset_Y, GLWin_Width,GLWin_Height, Viewscale* Offset of the Camera in Global OpenGL coordinate frame
Move	C→S→C (forwarded)	***M,*** *X, Y, UniqueID* Move the object with ID *UniqueID* to position *X,Y* in the OpenGL coordinate frame
Turn	C→S→C (forwarded)	***T,*** *UniqueID* Turn object *UniqueID* 180 degrees around the Z-axis
Front	C→S→C (forwarded)	***F,*** *UniqueID* Bring object *UniqueID* to the front of the OpenGL drawing list, which will make it appear in front of other objects.
Angle	S→C	***A,*** *Theta_X, Theta_Y, Theta_Z, UniqueID* Set the object *UniqueID* rotation in all three axes, rotating about its local object centre. 0 degrees is in line with the global OpenGL coordinate frame.
Create	S→C	***C,*** *X, Y, Size_X, Size_Y, isStatic, FileName, att1, att2, UniqueID* Create a card which is in effect a 3D OpenGL Box with a texture loaded from *FileName* and will be given the ID *UniqueID* from now on. (In this simplified version NonStatic objects are assumed to be playing cards and static objects are 3D Boxes which are push-buttons and the playing table)
Quit	C→S	***Q*** The client has quit the connection.

Table 3-4: Protocol between Server and Display Clients.

3.3.4 World Model for Simulator

As described above, every creation and manipulation of objects through the touch screen interface is recorded in a log file. To redraw the OpenGL world and give information about the objects, the state of the world and the information about the objects is saved in an internal representation (C++ Object CState). In order to use this world model in the transcription software, it is required to jump to any point in time of the recording, like on a tape recorder. This required the state-space of all world objects

at any point in time of the recording to be kept in memory. This approach is memory intensive but fast.

An object state holds:

PHYSICAL OBJECT IN THE WORLD MODEL OF THE SIMULATOR

Name	Description
UniqueID	its unique identifier for communication
Static	static or movable
Att	attributes, such as texture
x,y,z	world frame coordinates
xRot, yRot, zRot	Orientation relative to world frame
scale	scaling factor of
sizex, sizey	size of the object (default thickness is 4.0)

Table 3-5: Properties of the Physical Object in the World Model of the Simulator

As shown in figure 3-5, every communication is logged to a transcription file. Every state-space also contains a variable indicating the file position in the logfile (FilePos), the last action (Activity), the object ID of the object that was last involved in the action (ACT_ObjectID) and the number of objects in this state (NoOfObject).

The world model has no physics engine attached. This decision was taken since playing cards have a negligible mass. When an object is touched by the touch screen, it becomes the first object in the new state-space list so that it is drawn in front of the other objects. Any dragging move with the finger down on the screen will now immediately move the object that is under the mouse cursor relatively to the same position as the cursor / finger, to give an impression that one can drag objects around.

3.4 Multi-Modal Transcriptions

3.4.1 Transcription Tool

Recorded dialogues are transcribed to form a multi-modal corpus. A transcription has to capture the salient features of the gestures and utterances. Timing information has to be preserved in the transcription. The recorded audio, video and object manipulation data must each always carry timing information to allow a time-synchronised playback for the transcription. Similarly (Kvale *et al.*, 2004) uses timestamps to synchronize inputs from touch screen and voice.

Annotation tools to create a corpus of multi-modal human-robot interaction have been used in related projects, such as the BITT Corpus for Topic tracking (Maas, and Wrede, 2006) or for instance "the Corpus-Viewer" of (Koide *et al.*, 2004) , or the Home-tour scenario from (Green *et al.*, 2006).

Transcriptions are commonly stored in Extensible Markup Language (XML), (Witt, 2002). The advantage of XML is the range of available parser and browsers that can be used to process the transcribed information. Existing multi-modal transcription and annotation tools such as ELAN, EXMARaLDA, TASX, MacVisTA consist of a Video playback window, a timeline window and an optional transcription window (Rohlfing K. *et al.*, 2006). Most multi-modal transcription tools analyse video data. In corpus-based robotics, however, pre-processed sensor data, such as object coordinates can be available for transcription. This data must be imported into the transcription tool. In this research project the data is represented in a 3D environment on a screen rather than a video. Occasionally it is required to turn over a card, in order to check its ID for transcription. Converting the 3D environment recording into a video recording would remove the possibility to manipulate objects or change the camera viewpoint during transcription. Therefore a new transcription tool was designed called MuTra (Multi-Modal Transcription Tool). It was often necessary to go through menus and do several mouse clicks to transcribe a single utterance, since it is multi-modal; this was a further reason for creating a new tool. It is hard to develop a multi-modal transcription tool that suits all research projects due to the large variety of types of sensor data.

Figure 3-6: Mutli-Modal Transcription Tool MuTra, produced for transcribing the MIBL corpus.

3.4.2 XML Tags

There is no widely accepted standard for multimodal transcriptions. Most research groups create individual XML-tags to meet their needs. The wide range of data types in gesture annotation add to the diversity of XML-tags and the problem of standardisation. One could define, hypothetically, a XML tag for every action verb in English language. Bird and Liberman (1998), present a general specification body for multi-level annotations. The nesting of tags is limited in XML. It is not possible to represent partly overlapping events directly with partly overlapping XML tags.

The XML tags used for transcriptions are also defined in a DTD file (autotranscriber/mmtransdef.dtd) which can be used by tools such as Expat to check the syntax of the XML transcriptions. DTD (Document Type Definition) files contain a definition of XML tags and their syntax.

Example	Description

`<mmtranscript>`
...
`</mmtranscript>`
	root tag, indicating a multi-modal transcription

`<objmove t="315"`
`user="s"`
`ID="D/7"`
`from="+Table+Stock"`
`to="+Table"`
`until="327">`
`</objmove>`
	linear movement of an object ***ID*** by ***user*** at time ***t*** from location ***from*** to location ***to*** ending at time ***until***. Whereby from and to are a semantic label of an area. If areas overlap they are concatenated with +.

`<objrot t="388"`
`user="s"`
`ID="D/3"`
`roty="0" />`
`</objrot>`
	The ***user*** rotates object ***ID*** at time ***t*** around the y-axis to ***roty*** degrees. (the duration of the rotation is assumed to be 0.1 second if the ***until*** parameter is not given)

`<tv t="2396"`
`until="2428">`
`text`
`</tv>`
	An utterance with the teachers voice (tv) starting at ***t*** lasting until ***until***. The transcription text of the utterance follows before the closing tag </tv>

`<sv t="2424" until="2426">`
`text`
`</sv>`
	An utterance with the students voice (sv) starting at ***t*** lasting until ***until***. The transcription text of the utterance follows before the closing tag </tv>

`<corrected> text </corrected>`	The text enclosed in this tag will be filtered out when the transcription is used to build grammar. The transcriber has here the opportunity to remove repeated words and hesitations. In some cases also unfinished utterances, where the speaker changed his mind in the middle of the utterance.

Semantic Annotation Tags
(domain dependent)

`<dealing>...` `</dealing>`	Indicates the dealing phase of the game
`<game>...` `</game>`	Indicates the playing phase of the game (after dealing)

`<counting>`
 	
`</counting>`
	indicates the game phase of counting the points after the outplay has finished

`<pairrule phase=example no=1>`
 	
`</pairrule>`
	indicates the description of the pairing rule

`<sumrule>`	indicates the description of the summing rule
`<value>` `</value>`	indicates the description of the value of cards
`<exist>` `</exist>`	indicates utterances that describe which cards have or have not been taken out from the deck (exist-rule)

Table 3-6: Transcription Tags. All timestamps in the transcriptions are given in 1/10 of a second. For example t=315 means 31.5 seconds.

3.4.3 Transcription Method Guideline

In order to reproduce the method of transcription the guideline is described here, part of the transcription for the MIBL corpus was done by a specially hired person. Each utterance is transcribed. If an utterance contains several instructions they may be transcribed separately. Hesitations are transcribed and marked with <hesitation></hesitation>. If a sentence was incomplete, aborted or corrected by the speaker, the <corrected> - XML tag is used to remove text so that a sentence is left over that makes sense. The transcriber must strictly follow the audio data. It is not allowed to invent and add words to complete a sentence. The teacher is transcribed with <tv> tags and the students voice with <sv>. Audio is marked first, which creates these tags with accurate timing information. The timing information must be accurate enough so that it can be used to cut the audio file of the whole session into utterances. Any gestures that belong to the utterance must be put inside the <tv> tag. If more than one gesture belongs to the utterance, they are all put inside the <tv>. These gestures are regarded as a gesture group. The conversation has to be transcribed from the start when the use of the touch screen was explained until the end of the first test game.

3.5 Initial Corpus analysis

3.5.1 Quantitative

The MIBL corpus has 21 recordings. The first two recordings are between the 4 initial teachers, done while they were clarifying the rules. They have not been transcribed and analysed. The two pairs of initial teachers discussed the written rules to clarify between themselves. This minimised the possibility of unnaturally influencing the subjects by the experiment designer. There would otherwise be a danger of the experiment designer to express sentences and rules with the robotic reasoning in mind. This leaves 19 teacher-student recordings which have been transcribed. After transcription, some initial figures on the size of the corpus can be made:

The MIBL corpus has 35322 words in total. That is approximately the size of this thesis. Of these words, there are 1136 distinct words. The most common words, sorted by frequency are "the", "you", "and", "so", "ok", "a", "I", "cards", "yeh", "to", "that", "of", "one", "er", "card". Figure 3-7 shows their frequency and Appendix A1 lists all distinct words in the corpus and their frequency. In comparison the IBL corpus had 6600 words and 330 distinct words. Working with a corpus, that has to be manually transcribed and analysed, can be a mammoth task. Every decision that requires going through the transcription again has to be carefully considered, since it is time consuming. To go through the MIBL corpus to annotate a particular occurrence, such as a reoccurring action, in the text takes approximately 20 man-hours.

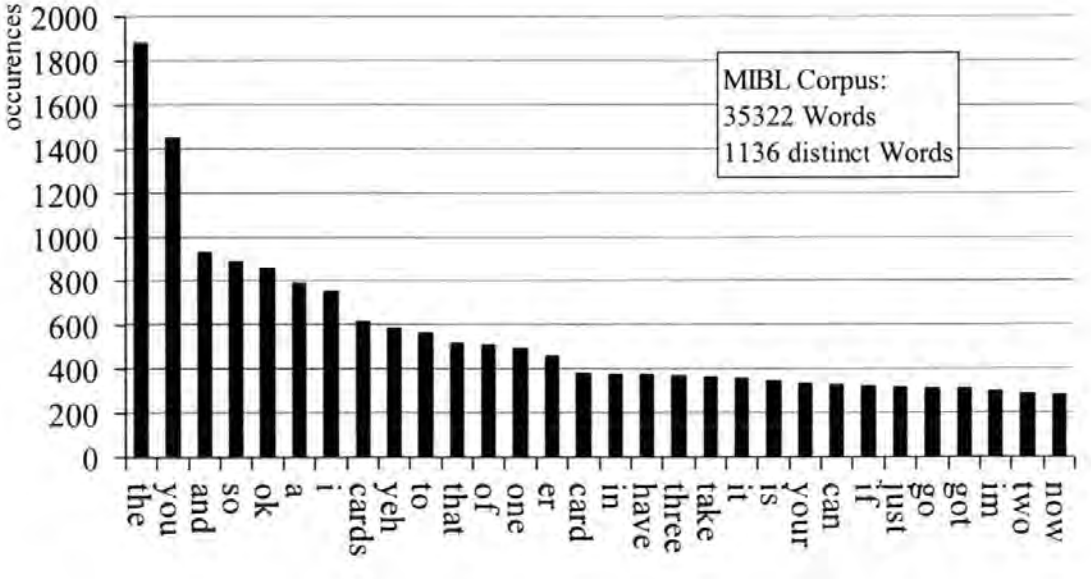


Figure 3-7: Word Frequency of the 30 most common words in the MIBL corpus

3.5.2 Types of Instruction Primitives

Analysing the utterances of the transcriptions reveals primitive procedures which the robot has to be able to carry out before learning from the end-user can start (the robot's "prior knowledge"). Such "language primitives" are specific to the level at which humans communicate with each other. They can constitute complex robot procedures in the background. These language primitives can be categorized into:

facts , sequential actions , context indicators , conditionals

Some examples of transcriptions and corresponding primitives are shown here:

TRANSCRIPTION WITH PRIMITIVE

Type	Utterance transcription	Primitive
facts	21.xml/696-723: "erm ok so the deck were playing with" 21.xml/729-748: "is a forty card deck" 21.xml/758-779: "with the eights nines and tens **removed**	not_exist(cardname=08, cardname=09, cardname=10)
sequential actions	03.xml/2431-2481: "er what you do first of all is… er you **deal** three cards for yourself face down"	move(cardname=cards, num_of_cards=3, source=?, target=hand2)
context indicator	11.xml/1588-1619: "and er this is how the game goes um"	context(indicator=new_case)
context indicator + conditional	07.xml/1167-1188: so if i **had** a four in my hand	context_type(type=imaginary) ifloc(cardname=a-four, num_of_cards=1, location=+hand1)
conditional	06.xml/-: " so if **youve** got a two in your black area"	ifloc(cardname=a-02, num_of_cards=1, location=hand2)
conditional	17.xml/1434-1507: " i could **capture** a card in the middle by **getting** another jack and then..."	ifloc (cardname=a-card, num_of_card=1, location=+table (ifcond (cond.type=equal, comparison=value, lhs =a-card, rhs1=jack, rhs2=?)

Table 3-7: Examples of transcriptions and their primitives and primitive types. Verbs that help to identify the primitives are marked in bold font.

As for context indicator and conditionals, examining the corpus for game rules reveals that a game rule is constructed from:

1. initial *context indicator*: e.g. "suppose you" or "if"
2. *conditionals*: e.g. "have an ace" or "cards with equal rank"
3. a sequence of *instructions* that have to be carried out when the conditionals are satisfied

These primitives are inserted into the transcriptions as XML tags manually. The question mark in the semantics (table 3-7) means, that there is missing information. Missing information is completed by combining semantics, multi-modal integration and by requests to the teacher. See sections 6.2.4, 7.5, 7.6 for detailed description on missing information.

SAMPLE OF IDENTIFIED LANGUAGE PRIMITIVES

Type	D/P	Language Primitive
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 3-8: listing of some primitive functions found in the MIBL corpus.
D=Declarative primitive, P=Procedural primitive. For a complete list of primitives see Table 3-9

In the case of action-primitives (require the robot to do physical action), the verb is often a synonym to the primitive name itself. Facts, such as `exist, value` contain a form of "to be". Conditional primitives can always be rephrased to start with "if". Since "if" is not always used, they are not as easy to identify as other primitive types.

3.5.3 List of Language Primitives

COMPLETE LIST OF LANGUAGE PRIMITIVES

Name	parameters	Description
context	*new_case*	a context change that will trigger a new rule-frame to be created
context_type	*imaginary*	the situation that the teacher describes is imaginary
context_type	*real*	the situation that the teacher describes is real (objects and actions exist in front of the robot)
sequencetrigger	*explain*	a initial sentence saying that the teacher will explain the rule, this is required for multi-modal integration.
players	*n*	Number of players in the game
imitate	*now*	request or implicit request to imitate the teachers action or follow a verbal command immediately
loopinicator	*each*	meaning that the action has to be repeated for each player
clause_relation	Prim,Prim,*modify*	modify last clause with this clause
clause_relation	Prim,Param,*query*	Teacher asks a question, the primitive is modified to become a question
clause_relation	Prim,*ns,not*	Negate, not true, make primitive negative
query		User asks question to the robot
move	CardCatNP,Loc,Loc	moving cards from Loc to Loc
move (pointing)	CardCatNP,Loc,Loc	pointing: whereby source and target location are the same Loc
turn		turn cards over
value	CardCat , Value	describes the value of a card in points
not_exist	CardCat	cards that have been taken out from the game/stockpile
exist	CardCat	cards that exist in the game
ifloc	CardCatNP,Loc	introduction of a card that is at a specified location
ifcond	Type,Property,LHS, RHS1,RHS2...	comparison of objects (cards) Type is the type of comparison i.e. "equal","smaller".. , Property is "colour","rank","value". LHS=objects on Left hand side of comparison, RHSx = right hand side
Dialogue Manager (dm)		
dm*(game_one)*	*game_one*	Dialogue move to switch from greeting the user to learning a specific game
dm*(ok)*	*ok*	positive confirmation "yes" , "okay"...
dm*(wants_play)*	*wants_play*	user wants to play with the robot, switches from greeting or learning to a playing state
dm*(shuffle)*	*shuffle*	will trigger the robot primitive on shuffling the deck
dm*(exit)*	*exit*	end of the dialogue
dm*(start_again)*	*start_again*	triggers the knowledge base to be erased and the teacher can start from the beginning
dm*(clever)*	*clever*	to deal with impolite expressions from the user
dm*(erase)*	*erase*	"forget that", "erase that" or similar expressions will trigger a deletion of the most recent rule-frame instruction
dm*(pardon)*	*pardon*	user did not understand what the robot said, which will trigger a repeat
dm*(robot)*	*robot*	user wants the robots attention by saying its name
dm*(turn-human)*	*turn-human*	"its your turn" or similar expressions will trigger this
dm*(turn-robot)*	*turn-robot*	"its my turn" or similar expressions will trigger this
dm*(dummy)*	*dummy*	dummy event required for implementation
reply		Reply to a question from the Dialogue Manager

Table 3-9: List of language primitives and their description. CardCatNP consists of a noun phrase that can include the number of items refered to and optionally a location, for example "the three cards here". This list shows not only the primitives from the initial analysis, it is a complete list.

Shown in the table above is a list of language primitives that have been analysed and implemented from the corpus. At the initial corpus analysis the primitives will not be as clear as described above to the eyes of the developer. Primitives can be formed as the developer looks for similarities and tags them with XML while keeping notes on possible primitives.

There are only 3 action primitives that were dominant during the dealing and game phase: moving, turning and pointing. At the end of the game, the cards are counted, which would be another action primitive, however this phase was not analysed in this work. A further possible action primitive could be visual-attention, which was not implemented since the robot is always looking at the card table and has a mental image of all the cards locations. The dialogue structure is expected to be different between human and robot therefore the dm-primitives (Dialogue Manager) are not completely created from corpus analysis. dm(erase), dm(pardon) and dm(robot) come from the experience with dialogues in IBL.

4. Action and Gesture Recognition

The term "gesture" has no commonly accepted definition. The Oxford Advanced Learner's dictionary (Hornby, 2000) defines gesture as "a movement that you make with your hands, your head or your face to show a particular meaning". In general terms, gesture could be defined as the act of non-verbal communication using body parts. An action could be defined, for example, as manipulation of objects. Demonstrating an action to another person or robot is also an act of non-verbal communication using body parts. Cadoz (1994) laid out three major *functions for gesture*:

- **semiotic**: it is used to communicate meaningful information; (for example sign language, pointing)
- **ergotic**: it is used to perform manipulation in the real world; (manipulating, creating of objects, for example cooking)
- **epistemic**: it is used in learning from the environment through tactile experience (by touching and manipulating objects).

The *functions of gesture* may be augmented by using an instrument according to Coutaz and Crowley (1995) .

In this respect gesture recognition systems link gesture and action and can be treated the same from a computation point of view as well. Therefore there is no differentiation between the terms "action" and "gesture" in this chapter.

The MIBL corpus contains some semiotic gestures while the majority is ergotic. Epistemic data can be found in the recordings where the teacher initially explained how the touch screen works. This epistemic data has not been used for this research. In card games, gestures can be pointing gestures, gestures moving cards from one place to another, e.g. stack to table, hand to table, re-arranging gestures, making a group of cards look tidier and turning over gestures.

4.1 Gesture Recognition by spatial mapping

Action and gesture recognition is in essence like speech recognition, where analogue sensor data streams are mapped into categories. These categories are more meaningful for further interpretation and reasoning. Commonly, machine learning algorithms such as Hidden Markov-Models (HMM) or neural networks are applied for this task. In some cases simple thresholding algorithms are sufficient.

The MIBL system uses a touch screen to simplify gesture recognition. The touch screen operates as an additional mouse to a computer. The effect of a user touching the screen is signalled as a mouse button-down event in the operating system. The MIBL GUI-Client translates these events to a "touch" of a virtual object. Moving cards on the touch screen is intuitively done by touching the card and dragging it to another position. The resulting data is a trail of X, Y coordinates of where the card is going. In case of a real service robot, this tracking data of cards on the screen could be the output to the service robot's vision system.

The X,Y data of the moving card must be categorised to complete the gesture recognition. The starting position, where the card was first touched (Figure 4-1) is compared to area boundaries which have been predefined (Figure 4-2). Combined with the type of action and object ID, this provides the initial parts of the gesture primitive:

For example:

<objmove t="2416" user="t" ID="D/5" from="+Table+Stock" to=? until=? ></objmove>

The target position "to" and end-time "until" is still unknown at this point. An algorithm continues to collect X,Y data of the manipulated card until the card comes to a rest or another card was touched. The final X,Y position is then categorised again by comparison to predefined areas. The gesture primitive is now complete.

<objmove t="2416" user="t" ID="D/5" from="+Table+Stock" to="+Temp2" until="2442"></objmove>

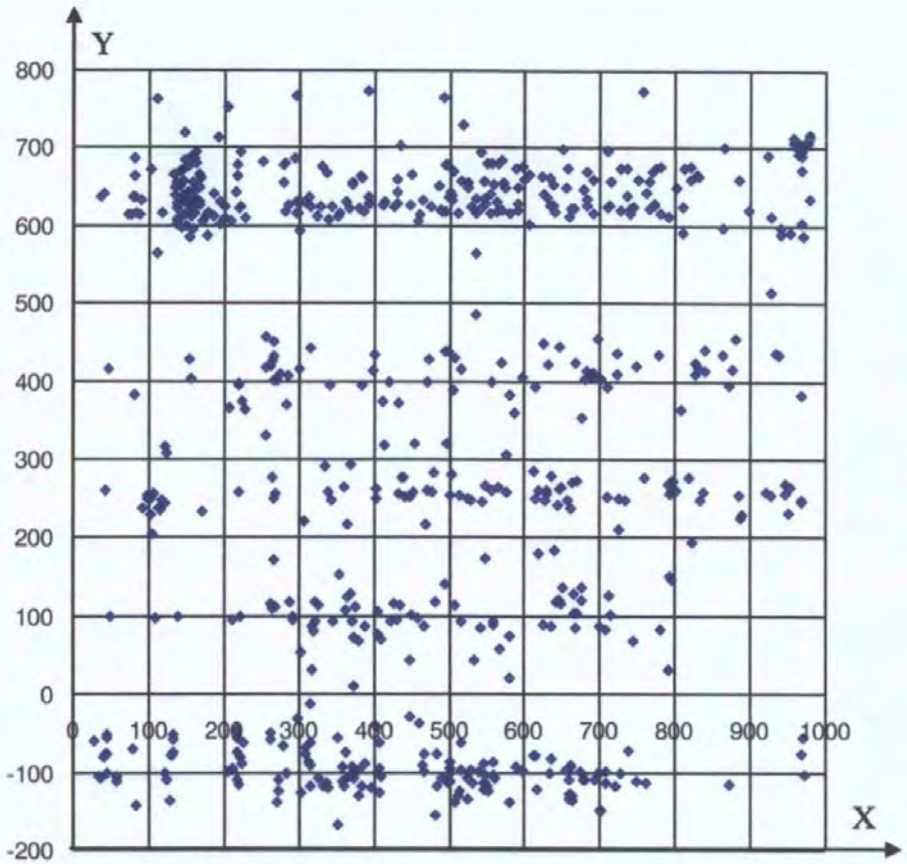


Figure 4-1: Resting Positions of Cards. Area map of the card game virtual area. Each dot indicates a resting position of a card during dialogue session 03.xml. The card positions concentrate clearly along 5 horizontal lines. These lines are labelled "hand2" (student's side), "temp2", "table", "temp1", "hand1" (teacher's side). Furthermore a concentration around 150,650 indicates the winning pile "side2" (see Figure 4.2)

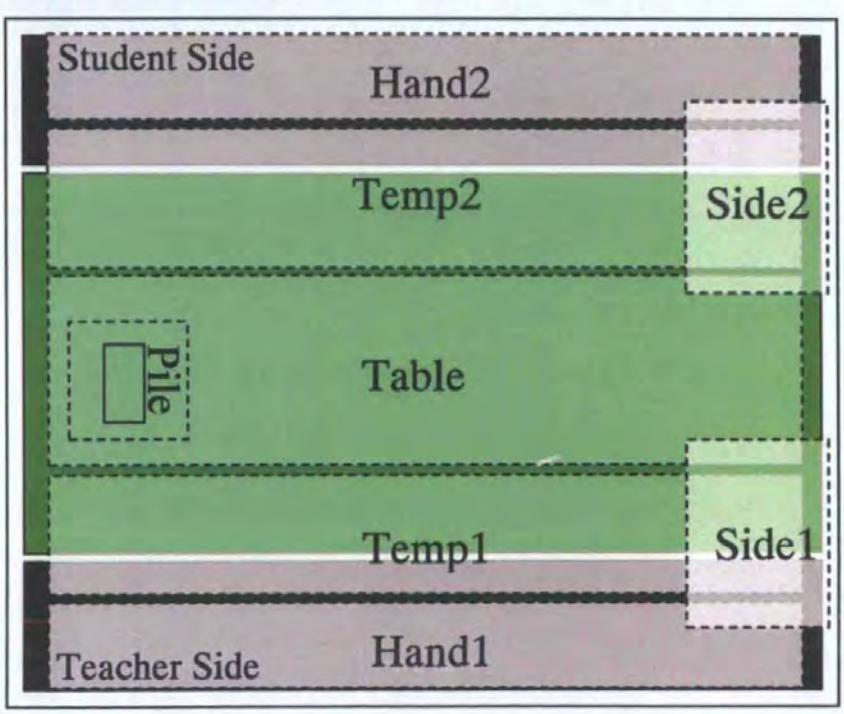


Figure 4-2: Area definition. After observing the actions of the users in the corpus, areas have been defined on the touch screen. Temp1 and Hand1 are on the teacher's side. Temp2 and Hand2 are on the student's side. The teacher and the student can only see and manipulate cards in their hand area and on the table.

Gesture recognition is a skill set, as mentioned in Section 2.1, skills can be learned but that is not the aim of this work. Therefore the gesture recognition skills are pre-programmed as a categorisation algorithm.

Gestures are recorded in a transcription file in XML format including time of start and end of movement, player doing the move, card identity and start position and destination, such as for instance:

<objmove t="2416" user="t" ID="D/5" from="+Table+Stock" to="+Temp2"
until="2442"></objmove>

Move-gestures that have the same start and final position category are often pointing gestures, or local rearrangements (tidying) of cards. In some card games or other domains, spatial relations are important. At this point an algorithm would have to be included that can recognise relative positions such as "A left-of B" or "A on-top-of B". In the game Scopa, which was investigated here, relative relations however, were not of importance.

The gesture recognition process is used in the MIBL robot module called autotranscriber (Wolf 2008, MIBL Manual). The gesture recognition process can also be used to assist multi-modal transcriptions. Once the initial area categories have been determined, gesture recognition is automatic. A smart transcription tool that could be trained while transcription is going on and suggest gesture transcriptions to the user is of advantage, since the transcription process is tedious. At the stage where utterances are transcribed, it is useful to have gestures already transcribed/recognised to indicate any connection between gestures and utterances in the transcription. This provides reference data for multi-modal integration algorithms.

GESTURE PRIMITIVES

Gesture Primitive	Description
`move(object,from,to)`	Moving an object. Whereby `object` is a physical object and `from / to` are locations from figure 4-2
`move(object,loc,loc)`	Pointing at an object. Whereby `object` is a physical object and `loc / loc` are the same, since the card is only moving very slightly when touching it for pointing at it. locations are from figure 4-2
`turn(object,side)`	Turning an object. Whereby `object` is a physical object that is rotated around its Y axis. `side` "up" is defined as 0° and "down" as 180°

Table 4-1: Table of Gesture Primitives in the Corpus

Occasionally pointing gestures are also found in the corpus. The teacher wiggles a card. This can be recognised as a special case of moving, whereby from and to locations stay the same. Furthermore cards are sometimes rearranged to look tidy. This can be falsely recognised as pointing.

The screen should have enough space for all objects to spread out. In a too small setup, humans will otherwise start rearranging objects because there is not enough space, which makes gesture recognition unnecessarily complex. Unfortunately, the MIBL corpus is affected by rearrangement gestures, especially when the game progresses many cards clutter the screen.

4.2 Gesture Production

Following the corpus-based approach, every gesture primitive is not only recognised, but can also be performed by the robot. The robots planner produces low-level robot instructions (LRI) (Chapter 7.8.6) which have the same level and syntax as the primitives. While the robot carries out its actions (move / turn), the robot sees the consequences with its vision system. Even though in the MIBL example this is going on a touch screen, there is still the separation between the LRI-module which generates the gesture and manipulates the object and on the other side the autotranscriber-module, "the robots eyes" which recognises its own actions. This feedback loop gives the robot the opportunity to recognise a failure of its actions. For example if the teacher interferes with the process.

4.3 Advanced Gesture Recognition

Statistical learning algorithms could be used instead in a real service robot, if the vision system output coordinates are noisy or if there are no clear cut boundaries. In these experiments it was found that a statistical categorization is not required because of the clear separation shown in figure 4-1. In other cards games however, the situation can be more complicated. It is advised to display histograms of start and end points of manipulators in order to determine the predefined areas. These boundaries of these areas could be defined by the density of the resting positions, based on data such as in figure 4-1. The separation into areas could further be defined with Voronoi diagrams. A Voronoi diagram is a set of points equidistant to two or more obstacles in configuration space (Russell and Norvig 2003)

This work deliberately used a simplified method for gesture recognition in order to concentrate the research on higher level reasoning. The simplified method could be summaries by using predefined Areas, and if a card has moved to that area a gesture is recognised.

With real robots, proper gesture recognition systems and skill learning systems would be needed. The research field of imitation learning and human motion recognition provides systems for this purpose. These proper systems could "plug-in" to MIBL. For instance Roy (2005) analogue beliefs and the categorizers, reviewed in chapter 2.6.1. Alternatively the HAMMER framework (Demiris and Khadhouri, 2005) that can learn from observing the demonstrator. The advantage of the HAMMER architecture is that actions can be recognised (perceived) and reproduced within the same architecture. Another system that can do human motion recognition specifically aimed at categorising these actions into sequences is from (Loesch *et al.*, 2008, Otero *et al.*, 2006). It uses a body model of a human to match tracked motion data against the model. All three mentioned systems are designed specifically for human-robot interaction.

5. Integrating Gesture and Language

5.1 Challenges of Multi-Modal Integration

As established in the introductory section 1.1, natural communication between a human and a robot requires a multi-modal interface. Typically the modalities of speech and gesture are used to achieve natural communication going beyond the traditional input modalities of keyboard and mouse. The subject of multi-modal integration has its origins in Human-Computer Interaction (HCI), see for example Bolt (1980), but has become more and more important in Human-Robot Interaction (HRI) probably due to the fact that robots are embodied agents. On a first encounter of a non-expert user, speech, touch and gesture will be the first modalities of interaction, especially if a keyboard or screen is absent. With multi-modality comes the problem of multi-modal integration. The subject of multi-modal integration is concerned with combining data from modalities into a coherent form so that an interpretation is possible. Multi-modal integration is taking advantage of the additional information available, compared to a single modality, by removing redundancy and minimising ambiguity and resolving references such as pronouns. Redundancy in the information channels can be taken advantage of in multi-modal integration for the purpose of synchronisation and increased confidence. Fig 5-1 shows a simple block diagram of information flow in human to robot instruction which illustrates the "communication channels" and the common knowledge connecting the teacher and student.

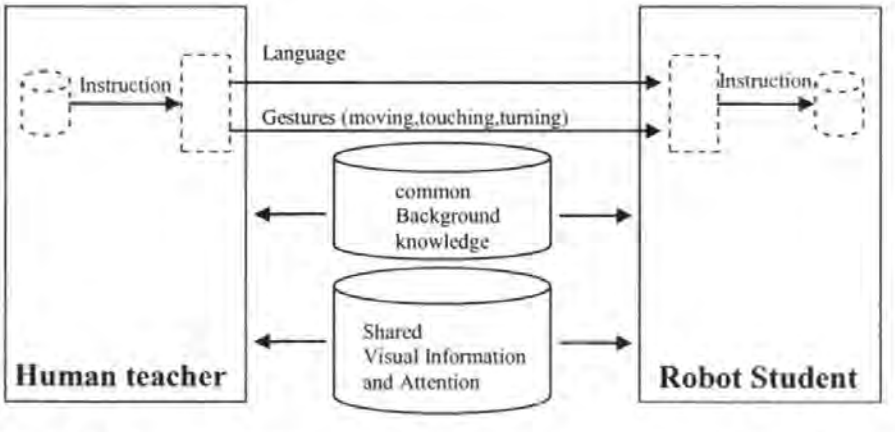


Figure 5-1: Information exchange: Instructions from a human teacher to a human student or robotic student are divided between two communication channels. Both share common background knowledge about the domain. Teacher and Student are both looking at the task with their eyes and share visual information, i.e. the existence and position of objects in space.

Instructions are explicitly exchanged by communicating in the obvious modalities of gesture and language. However, often these instructions are underspecified. To complete the missing information it is often required to draw on visual information, background knowledge and discourse. While the integration is described in this chapter, the completion process using background knowledge and discourse is described in chapter 7.

5.1.1 The Timing Problem

From a robot perspective the modalities are two communication channels that need to be recombined into one message. This problem appears to be related to the time-synchronization problem in communication engineering. Another related engineering discipline is multi-sensor fusion, which is often solved by using Kalman filters (Maybeck 1979). However, the integration of free flowing speech and gesture has added complexity.

The timing problem can be brought down to the fact that confusion arises about which utterance has to be linked to which gesture, because they are not strictly synchronised. Figure 5-2 shows an example of relative timing of speech and gesture from the collected corpus of card-game instructions. In, addition it will be shown that, in free flowing conditions, timing does not provide sufficient information for correct multi-modal integration, which needs to rely also on semantic criteria. Therefore the term "pairing" is used rather than the term "time-synchronization".

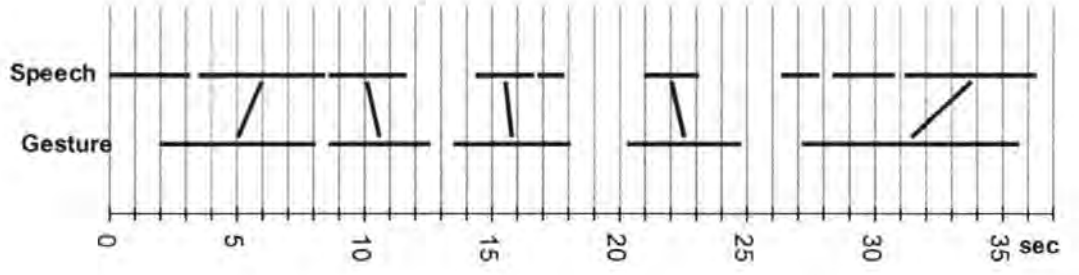


Figure 5-2: Time-lines of speech and gesture, where diagonal lines indicate which utterance and gesture are paired together.

This example of the corpus confirms that utterances can follow up very closely behind each other, this was also observed by (Oviatt 2001) who writes: "*It is a myth that speech and gesture always have time overlap if they belong together.*"

From a human perspective, the problem of multi-modal pairing can be referred to as "acoustic packaging" of actions using language. Brand and Tapscott (2007) investigated infant directed speech. They found that infants can segment a flow of actions into segments of actions by the help of speech from the age of 9.5 month. Segmentation is part of recognising an action and its intention. It is suggested that infants in their early live use intonation and speech to mark the beginning or end of an action. These actions can be novel to the infant. This speech acts coincide in timing with the action. Later on in live as the "acoustic packaging" has been bootstrapped (bottom up), the exact coincidence between speech and action becomes less important since the actions are not novel anymore and can be segmented by recognition.

5.1.2 Pairing and Unification Problem

As discussed in the review of multi-modal human-robot interaction systems (section 2.3.3), the integration problem has also been addressed in the past by Oviat (1999); Johansson (2001); Nigay and Coutaz (1995); Chai (2003), where it is sometimes referred to as multi-modal fusion see Djenidi *et al.*, (2004).

In this thesis a late fusion model is described that can successfully select (pairing) language utterances and gestures and combine (unify) the information stored in both to a single representation of a primitive. A simplified block diagram of the process is shown in figure 5-3.

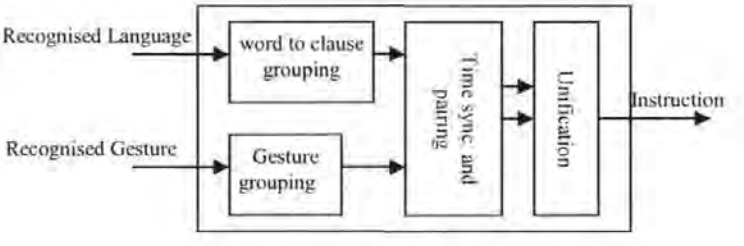


Figure 5-3: Multi-modal integration system

Figure 5-3 is clearly different from an early fusion model, such as Roy (2005), where fusion would happen already at the recognition phase.

In order to integrate multi-modal information, the data types that overlap need to have a coherent format. Furthermore a decision needs to be taken on how the overlapping information can be unified into a single data set. The coherent representation and unification steps are described in section 7.6 after language semantics and pairing have been introduced in this chapter.

5.2 Multi Modal Data Characteristics

5.2.1 Distribution of Information in Multi-Modal Data

Within one dialogue there can be a different distribution of information between the two modalities. Table 5-1 shows examples of the MIBL card-game corpus. The first of these examples (06.xml) is a uni-modal instruction using only speech. The second example is a multi-modal instruction, compiled of both speech and action on cards.

```
06.xml:
<tv t="1092" until="1141">
  the idea is to win the most cards from the middle until your hand goes
</tv>

07.xml:
<tv t="949" until="995">
  and if you flip these three cards they are your hand
  <objrot t="954" user="t" ID="C/KK" roty="0"/>
  <objrot t="960" user="t" ID="D/2" roty="0"/>
  <objrot t="966" user="t" ID="C/7" roty="0"/>
</tv>
```

Table 5-1: Examples from the card game corpus.
`t` and `until` represent the time of start and end of the event.

In the extreme, one might see cases where multi-modal instructions only include action-demonstrations, e.g. "to deal cards you do as follows", followed by a long sequence of actions. This would correspond to the extreme left end of the diagram in figure 5-4.

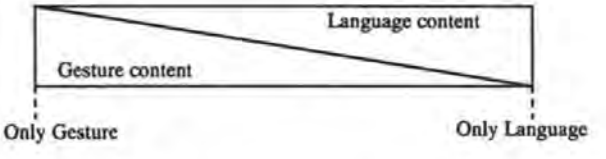


Figure 5-4: Spectrum of information content divided between gesture and language in a conversation

In general, a multi-modal instruction unit is composed of an utterance associated with a sequence of actions of variable length. In the MIBL corpus there were no instructions that were purely described by gesture. Utterances themselves are sequences of words that are grouped through the setting of timeout limits in the speech recognition engine. In the following section, the method used for the grouping of actions (gestures) is

described. The method has been developed by investigating the dealing-phase of the corpus of game instructions.

5.2.2 Gesture Groups

One instruction utterance can be accompanied by several gestural instructions. For instance, "deal three cards onto the table" entails 3 card displacement actions. Or in an example of another domain: "use two spoon of coffee per cup" entails a repetition of elementary actions. Thus, single actions need to be grouped into a unit that complements the single spoken clause.

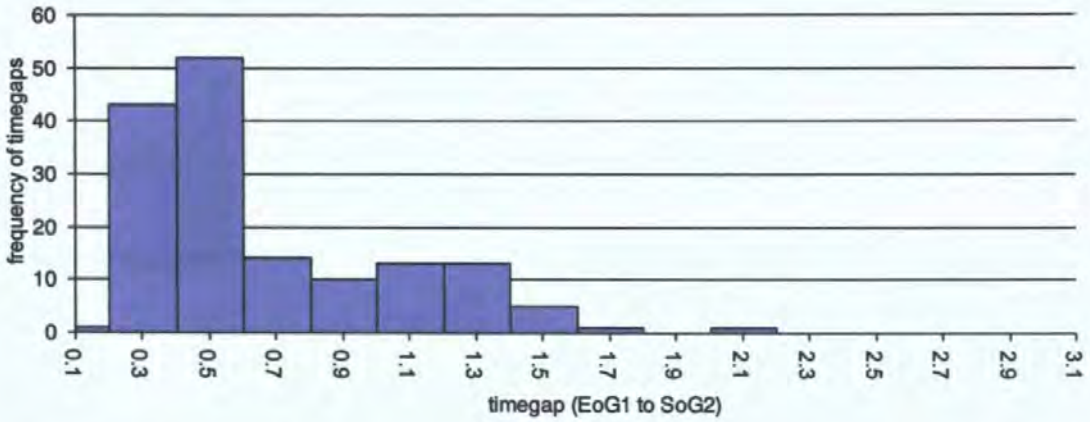


Figure 5-5: Timegap between two gestures . Histogram. Timegap is measured from end-of-gesture to start-of-next-gesture in a group of gestures that belongs together

From measurements of the corpus it was found that generally actions that follow within 2 seconds of each other could potentially form a group. However the criteria for which individual actions make up a group are not known before considering the utterance. For example the sentence "...take three cards for yourself..." (03.xml, t=2431) is different from "...you deal three cards to each player" (11.xml, t=1240). One time the instructor refers to 3 actions and, another time, to 6 actions in this two player game. Therefore the grouping algorithm devised for this corpus proposes more than one possible group of actions and leave it to a later stage, when the verbal instruction is considered, to choose the right group for unification. This is called a group "hypothesis". All group hypotheses are stored in the multi-modal integration algorithm and the one that fits to the utterance will be used.

So far investigations into dealing cards found that two criteria for grouping were sufficient.

1. If the objects involved in the gestures share the same location or
2. if the objects involved have been manipulated in a similar way, i.e. turned over or moved, they potentially form a group.

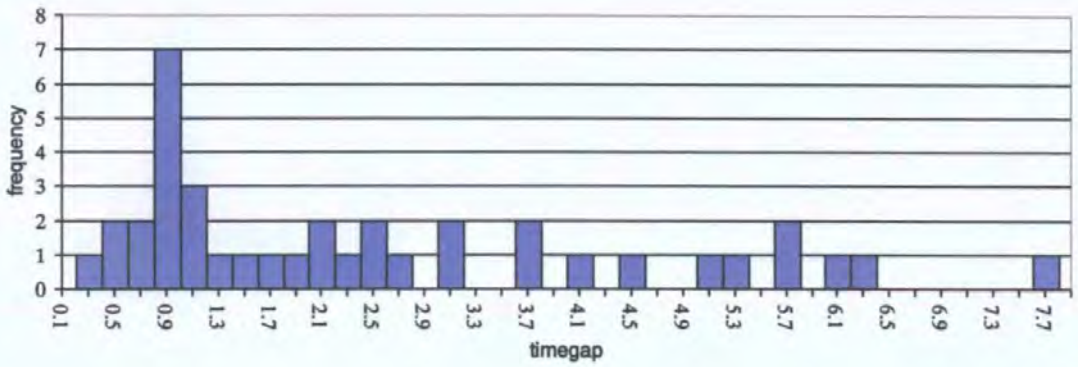


Figure 5-6: Timegap between two gestures-groups . Histogram. Timegap is measured from end-of-gesture-group to start-of-next-gesture-group. Note that there is often only a timegap of less than a second, which means timing information alone is not enough to perform grouping, and semantics i.e. the type of gesture is required to group gestures.

The grouping algorithm can not rely on timing alone, since figure 5-6 demonstrates that there is often less than 1 second between two groups that describe different action primitives. The only way to distinguish these groups is by looking at semantics, i.e. type of gesture and location.

From a theoretical view these grouping principles coincide with some of the Laws of Organisation in Gestalt theory (Wertheimer 1923). For example the proximity law, which states that objects or events that are near to one another (in space or time) are perceived as belonging together as a unit. Once candidate-groups of actions are established, the next step is to determine which utterance corresponds to which group. One approach is to consider temporal relations shown in section 5.2.4 .

5.2.3 Grammar for Gestures

A series (group) of words form a sentence and a grammar can be defined for it. The same can be said for a series of gestures:

A series of gestures can also be written as a gesture grammar

Figure 5-7 shows a CFG implementing the two gesture grouping rules mentioned earlier. An interesting question is: if there is a grammar for every modality (in this case gesture and language), is there a multi-modal grammar? Could it define multi-modal integration? This hypothesis is not answered in this thesis, though it presents an interesting thought for future investigations.

```
GG   → GGTT | GGTH | GGMT |...
GGTT → TT GGTT
GGTH → TH GGTH
GGMT → MT GGMT
TT   → turn at-hand
TT   → turn at-table
MT   → move from-table
```

Figure 5-7: Example of a right-recursive context free grammar for gestures. GG=gesture group, GGTT=gesture group of turning gestures in the hand, GGTH=gesture group of turning gestures on the table...

CFG Grammar for gesture recognition has been used in the past. Recognition with CFG is often combined with HMM. (Ogale et al., 2005) looked at recognition of a sequence of full body motion. However, usually only a single modality is considered. In the case of multi-modality, the CFG must be build so that the output is a group of gestures that are on the same level as a language primitive.

5.2.4 Investigation of Timing for Linking Gesture Groups to Language

In the collected corpus of card-game instructions, instruction-gestures can start before, during and after a corresponding verbal utterance. Figure 5-8 shows an example where the gesture-group G2 starts before the corresponding utterance U2. Table 5-2 show the transcription corresponding to figure 5-8.

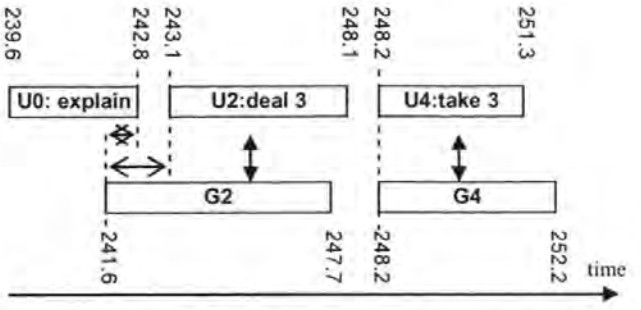


Figure 5-8: Timing Diagram This figure shows three incoming utterances (U0,U2,U4) and two incoming gestures-groups (G2,G4). The timestamps of start and end are given in seconds (timeline not to scale).

EXAMPLE DIALOGUE (SESSION 03 FROM MIBL CORPUS)

No	Time in sec.	utterance text or gesture semantics
U0	239.6-242.8	"I will just explain how you deal the cards"
U2	243.1-248.1	"er what you do first of all is.. er you deal three cards for yourself face down"
G2	241.6-247.7	move(D/5,C/2,H/QQ, Stock , Temp2)
U4	248.2-251.3	"and I will take three"
G4	248.2-252.2	move(D/QQ,D/KK,D/AA, Stock , Temp1)
...	...	...

Table 5-2: example Dialogue

In order to explore the possibility of using timing for linking action-groups with their corresponding utterance, the relative timing of their onsets and offsets was analysed. By creating a histogram using the data set-1 of the MIBL corpus, it is possible to see the relationship between start-of-speech vs. start-of-gesture-group, see histograms figure 5-9 and 5-10.

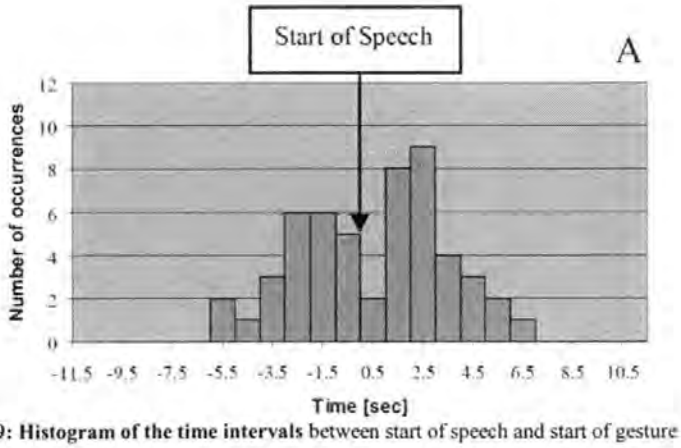


Figure 5-9: Histogram of the time intervals between start of speech and start of gesture

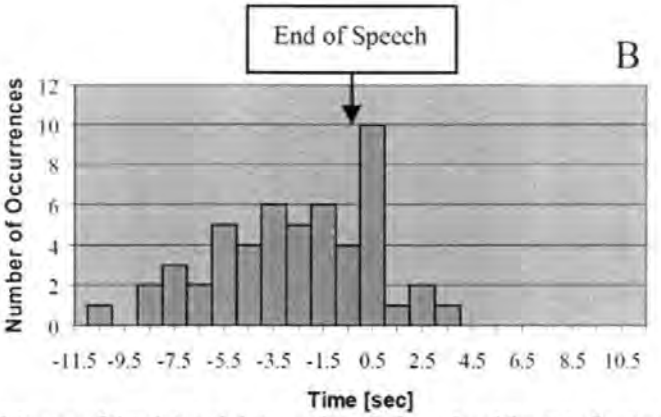


Figure 5-10: Histogram of time intervals between the end of speech and the start of gesture. Only gestures associated with speech events are plotted.

Figure 5-9 shows that gestures never start more than 5.5 sec before speech starts. Figure 5-10 shows that gestures never start later than 4 sec after corresponding speech ends. These observations suggest that a time window around the speech duration could be used to group speech and gesture. The time window borders are based on the maximum extends of the histograms. A graph that compares end-of-speech to end-of-gesture is not shown here, because potentially an action can take a very long time, and there would be no useful correlation.

5.3 Multi-Modal Integration Algorithm

5.3.1 Time-Based Integration

Results in the previous section suggested that a time-window-based method could be used for pairing speech with gesture. In figure 5-10, the markings in diagonal strips are time-windows. If a gesture group starts within the time window of a utterance, then the utterance and gesture-group could be paired.

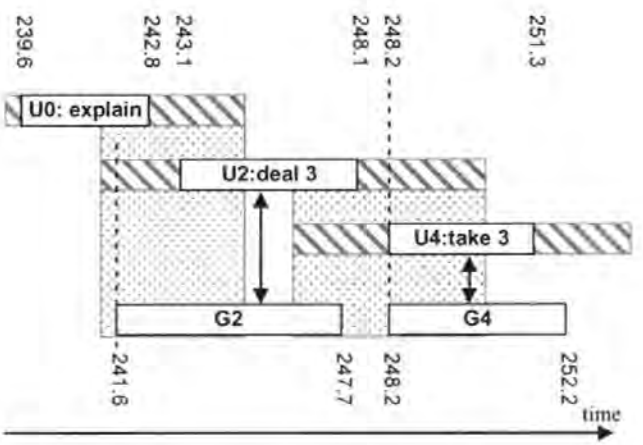


Figure 5-11: Time windows: This figure shows the three utterances with their corresponding time-window. If a start-of-gesture falls within that range, it is a candidate for pairing with the utterance.

However, utterances often follow up very closely behind each other in free flowing speech. In such a case, time-windows tend to overlap. The time-window overlap is indicated in figure 5-11 as a grey zone. Only 40% of start-of-gesture times do not fall in a grey zone and allow an unambiguous pairing. If, in ambiguous cases, the pairing is done with the nearest neighbour a rate of 78% of correct pairings is be achieved when compared to manual transcription (the remaining 22% are incorrect assignments). A nearest neighbour is found by comparing if the start-of-gesture is nearer to the end of the previous utterance or to the beginning of the next utterance.

The experiment was carried out by using the dealing-phase of the collected corpus. The figure of 78% can appear as a good result. However the remaining 22% of erroneous groupings can cause problems during semantic integration (unification). If the erroneous groupings are not rejected by the unification, then the robot would learn incorrect information. In a symbolic system this has to be avoided at all costs. Therefore it is

worth seeking an algorithm that can pair gesture to language without error, ideally 100% correct before unification.

5.3.2 Semantics for Multi-Modal Data Integration: Algorithm and Results

Investigating the erroneous groupings results from the nearest neighbour approach suggests that it may be possible to achieve perfect pairing by using semantic information, e.g. comparing the action referred to in the utterance with the action type in the gesture. The main verb in the sentence indicates the action. In conjunction with the object of the sentence the action semantic "*primitiveverb*" can be inferred, see extract of the grammar below. Parsing the sentences given in table 5-3 would result in U0(explain) having no primitive-verb, U2(deal) would be mapped to move and U4(take) would also be mapped to move.

```
PrimitiveVerb [
        (move) (<primitiveverb move>)
        (pull) (<primitiveverb move>)
        (drag) (<primitiveverb move>)
        (put)  (<primitiveverb move>)
        (take) (<primitiveverb move>)
        (turn) (<primitiveverb turn>)
        (turn them over)      (<primitiveverb turn>)
               (over) (<primitiveverb turn>)
        (flip) (<primitiveverb turn>)
        (deal) (<primitiveverb move>)
       ...
]
```

Table 5-3: simplified extract of the grammar assigning of the semantic primitiveverb. This grammar was created manually (pre-programmed).

The "primitiveverb" is a semantic category of what type of action is going on. Comparing the primitiveverb from the language with the type of action in the gesture enables semantic testing if the gesture is linked to the utterance. i.e. if the utterance is about moving a card, then the gesture can only be about moving a card, not turning a card.

The new algorithm applied primitive matching only in cases of uncertainty, when the start-of-gesture lies in an overlapping time-window. The nearest neighbour method ranks which one of the two utterances is considered first.

Some utterances in the corpus don't have action verbs. For example 03.xml/2396-2428:"I will just explain how you deal the cards". In a case where two utterances follow very closely behind each other, the grouping should be rejected if the action-verb

is missing. The complete algorithm using the concepts described in this section is presented in table 5-4:

```
consider the last two utterances
if:gesture starts outside overlapping window
{
        if: gesture starts in window[last]{
                found gesture-utterance match
}
        if: gesture starts in window[last-1]{
                found gesture-utterance match
}
}else:gesture starts in an overlapping timewindow
{
        choose the nearest utterance to the gesture
        if: the semantics to nearest neighbour match {
found gesture-utterance match
}else: semantics don't match {
       consider semantics of other utterance
       if: other utterance semantics match{
       found gesture-utterance match
       }
}
}
```

Table 5-4: Algorithm for integration of utterances and gestures

The algorithm only considers the two most recent utterances. The algorithm is called after an end-of-gesture-group is detected. On the data set-1 of our card-game corpus, it achieved only correct pairings, so that no false information was generated. The algorithm however rejected 11% of utterance-to-gesture groupings that should have been assigned.

Investigating the 11% of pairings that have been missed out, it was found that they are partly due to under-specification in the utterance, a missing action-verb, or due to grammar-errors in the robust parser. The current grammar attempts to split utterances at conjunctions like "and" or "then", to avoid processing two instruction steps in the same utterance. This approach does not solve the problem in all cases. It is therefore likely that further work will reduce the number of rejected pairings.

5.3.3 Pointing Gestures in Multi-Modal Data Integration

Martin, J. C. (2005) provides an overview of Multi-Modal systems that involve pointing gestures. Evidence by Oviatt et.al. (1997), shows that pointing gestures coincide in timing closely to the reference word (here, there, etc) in the utterance. Oviatt found a maximum of 3 seconds between the deictic reference and the corresponding pointing gesture. Table 5-5 shows a complete list of deictic words and how often they occur in the MIBL corpus.

TABLE5: DEICTIC WORDS FOUND IN THE MIBL CORPUS

DEMONSTRATIVE DEICTIC WORDS		POSSESSIVE DEICTIC WORDS		SPACIAL AND OTHER DEICTIC WORDS	
word	frequency	word	frequency	word	frequency
the	862	your	282	**Other**	
that	498	its	197	them	133
this	203	my	141		
those	85	ones	16	**Spacial**	
these	57	our	13	there	156
		their	3	here	69
		his	1		
		her	0		

Table 5-5: Deictic Words

Below is an example of the MIBL corpus:

EXAMPLE DIALOGUE (SESSION 11 FROM MIBL CORPUS)

No	Time in sec.	utterance text or gesture semantics
U0	162.2 -168.6	"what you have to do is er if you take one card from your er three cards"
U1	168.8-170.3	"and you have to either"
U2	170.4-172.2	"like im doing here"
G0	171.5-173.5	move(C/7, , Hand1 , Temp1)
U3	172.3-173.8	"you either match it with a card on **here**"
G1	173.9-175.6	move(D/7, Table , Table) **(pointing)**
...	...	...

Table 5-6: Example Dialogue

The example shows that the word "here" does not overlap with the pointing gesture, however the time of gesture start and the speaking of the word "here" (173.6 – 173.8) is only 0.1 seconds.

An algorithm to find the a near pointing gesture has been devised. It is consulted by the language primitive `ifloc` if the location semantics contain "`+here`" or "`+there`" rather than a specific location.

5.4 Discussion on Multi-Modal Integration

Unlike other proposed methods (Johansson 2001, Johnston *et al.*, 1997) the algorithm does not perform a full unification to check for semantic compatibility. Full unification checks every primitive parameter for contradiction. I found that performance was improved by making grouping and linking algorithms simple, with a minimum of symbolic intelligence and rely on nearest neighbour for ranking. By a minimum of symbolic intelligence it is meant that only a check for compatibility, such as if the primitive-type is the same, i.e. move matches move, turn matches turn. If later, while unification is performed, the primitive parameters don't match (contradiction) then the user is asked to clarify the instruction. The improvement in performance can be explained by the fact that it is easier for the system to recognise different primitives than it is to recognise the correct primitive parameters. Nigay and Coutaz (1995) use the term "melting pots candidates" for the gestures and utterances that are in the buffer for linking, whereby the buffer is a memory that contains the most recent gestures and utterances. Coutaz has considered the problem in some depth.
Although the algorithm has not been tested in other domains, its approach of action-verb matching and nearest neighbour matching has the potential to be a quite general and robust method of multi-modal integration. The algorithm has been trained on the dealing-phase of the corpus, where actions occur particularly fast and frequently. In the game explanation phase (after dealing), the integration problem became less important, since actions were less frequent.

Oviatt (Oviatt *et al.*, 2004) describes that there are two types of people when it comes to using a multi-modal interface namely "simultaneous" and "sequential" integrators. In our experiment this is not the case. In the domain of card-games, demonstrations can start before, at the same time and after speech. See histogram in figure 5-7.

5.4.1 Advantages of the integration algorithm

The integration algorithm shown in 5-4 is a late-fusion model, because it deals with information in symbolic format and on an utterance level. The advantage over early fusion models using HMMs or similar is that it needs less training data. Training data in this case is used to determine the limits from the histograms. It is generic as the form

presented in 5-4 does not show any context or domain. It can be analysed and understood easily compared to early-fusion models. The algorithm, when trained on the corpus, did not cause any erroneous pairings on the corpus.

5.5 Conclusions on Multi-Modal Integration

This work shows that it is possible to pair speech and gesture as occurring in unconstrained free-flowing human-to-human instruction dialogue. The proposed pairing algorithm combines timing and semantic information. An important property of this algorithm is that it does not make erroneous pairings.

Further work will explore if this algorithm allows processing unconstrained free flowing multimodal instructions through direct human-to-robot interaction, rather than by playing back teacher actions of a human-to-human corpus. In these experiments, the touch-screen interface will be the same; therefore the timing of multi-modal inputs is expected to be similar to that of human-human interaction. Tests have been carried out successfully and a robotic agent that can learn how to deal cards has been implemented, see chapter 8.

6. Corpus-Based Clause Grammars

Section 3.4.2 showed the form of the transcribed corpus in XML. Section 3.5.3 showed the language primitives that are extracted. The aim of this chapter is to introduce how corpus transcriptions are tagged to find language primitives and furthermore to show how the utterances are mapped to the language primitives. In a nutshell, the mapping from utterances to language primitives is done by a defining grammar.

6.1 Corpus Tagging

The transcription is stored as a XML file indicating which utterance belongs to which gesture. Each teacher explanation was tagged so that tasks and sub tasks are hierarchically divided. In this case the explanation has been divided with XML tags into the three game phases of dealing, game-play and counting points at the end. This could be referred to as *context tagging*. Inside the dealing phase 6 sequential instructions of moving and turning of cards (sub-tasks) were found. In the game-phase, the rules on pairing/capturing cards and the description of the ranking of cards were tagged manually. The tagging process also helps the developer to break down complex explanations into logical parts for which robot functions can be implemented correspondingly, step by step. An example below (`20.xml/912-955`) shows the tagging of the value-rule, which describes the value of a card in points. Example figure 6-1 shows the tagging and grammar of the pair rule.

```
<dealing>
...
</dealing>
<game>
   ...
   <value-rule>
      <tv t="912" until="955">
         and the jack queen and king become the eight nine and ten
      </tv>
   </value-rule>
   ...
</game>
```

Figure 6-1: xml-transcription. The utterance of teachers voice <tv> has been tagged as being part of the game-play <game> and as being an example of a <value-rule>. The value-rule, like in many card games, describes how many points the cards are worth.

The tagging process is also useful for automatically generating a grammar later.

```
<dealing>
...
</dealing>
<game>
   ...
   <pairrule>
   ...
   <tv t="2027" until="2060">
     ive won those cards there and now youll down to the three and then its your go
     <grammar>([ive (I have)] won those cards there)</grammar>
     <grammar>
          (and now [youll (you will)] down to CardCatNP:cn)
          {return( strcat(strcat("move=" $cn)  ",?,+side2:") )  }
     </grammar>
     <grammar>
          (and then its your go)
          {return("dm=turn-human")}
     </grammar>
   </tv>
   ...
   </pairrule>
   ...
</game>
```

Figure 6-2: xml-transcription with grammar. The utterance of teachers voice <tv> has been tagged as being part of the game-play <game> and as being an example of a <pairrule>. This example contains also the GSL grammar with language-primitive move() and dm() being passed to the knowledge base.

The example in figure 6-2 shows also that "I" and "you" is used interchangeably in explanations. Therefore the robot has to assume that it has to do all actions even the human says "I will" in his demonstration.

6.2 Clause Grammar

6.2.1 "One Clause One Primitive" Principle

Table 3-7 showed the verbs in bold font. Verbs can in most cases be tied to primitives in the form of VERB (ADJ1, OBJ1..). whereby the verb is a synonym or related verb for the primitive. For instance "deal" is related to the primitive verb-name "move".

The smallest natural language structure that contains a verb is a *clause*. By definition a grammatically complete clause must contain a *finite verb* and a subject. Therefore a grammar that maps natural language to primitives is easy to define if it is clause by clause. This means that sentences that consist of two primitives also have two clauses and should be split in the grammar definition. For instance, 17.xml/1434-1507 shown in the table above shows the word "capture" and "getting" in the same sentence, but split into two clauses, so it is possible to split it into the two primitives. In this example the primitives share the objects, therefore it would be wrong to split them completely, however in many cases the primitives are independent. Typically in sequences, when the phrase "and then" is used. This lead to the assumption that it is save to use a parser to cut utterances into clauses when certain sentence structures and keywords such as "and then" occur. This is described in more detail in section 6.2.1.2. . Lets first investigate dependent clauses before going on.

6.2.1.1 Clause relations

In English grammar *Dependent clauses* are clauses that can not stay alone as a sentence. Dependent clauses modify or add information of the previous clause. A special primitive (`clause_relation`) is necessary to show the dependency. The `clause_relation` primitive also shows temporal relations between clauses. A database of clause relations has been added:

Clause	Language Primitive
`clause_relation(move,move,modify)`	further information about a move
`clause_relation(turn,turn,modify)`	further information about a turn
`clause_relation(FIRST,SECOND,sequence)`	the word 'after','then', ... indicates a sequence relationship of clauses
`clause_relation(FIRST,SECOND,first)`	the word 'first','initially' ... indicates a first-in-rule relationship of clauses
`clause_relation(FIRST,SECOND,last)`	the word 'in the end','finally' ... indicates a last-in-rule relationship of clauses

Table 6-1: *Expressing* relationships between clauses

By default, the robot assumes a sequence of primitives, whereby every clause/primitive produces new semantic knowledge. `clause_relation(_,_,modify)` is an exception. It indicates that a previously specified clause/primitive is modified.

An alternative representation of clause relations and semantic relationships in sentences in general is the Lambda Calculus. A short introduction of the Lambda Calculus was given in chapter 2.6.6. When comparing the Lambda Calculus to the structure presented here, it can be seen that the structure here (language primitives and parameters) is more simple and can be easily written down by a non-specialist designer without thinking about complex relationships between the words. One of the simplifications is that to uniquely identify and hold together the semantics, its timestamp is used in the background, rather than multiple nested brackets and variable definitions.

6.2.1.2 Parsing to cut utterances into clauses

Corpus utterances have been cut into clauses by the help of a natural language parser. Cutting is done if clauses are linked with words like "and", "and then" or "so". For identifying the end of a clause, The Apple Pie Parser by Sekine and Grishman (1995) provided most accurate results on the MIBL corpus. The Cass Partial Parser (Abney 1991) in conjunction with the Brill Tagger (Brill 1992) was tested on the corpus. It struggled with the spoken language and ambiguous expressions such as "take the five" like most parsers would, however the main reason why it was not used, is that it failed to identify the ending of clauses in many cases, compared to the Apple Pie Parser.

The Apple Pie Parser output was searched for 3 rules to detect and cut clauses: the phrases "*so*", "*and then (SS*" or "*so then*". A total of 914 utterances were considered in

the corpus tagged by <game> as being part of the first game-phase in data set-1, which follows after the dealing phase. Of these 914 utterances in 681 cases there was no "and" in the remaining cases there where

- 84 cases with "*and*" as the first word in the utterance, which means they don't need to be cut, because the clause is already at the beginning of the chunk
- 95 cases with "*and*" in a noun group which means they don't need to be cut because a noun group with "*and*"s is a list of nouns rather than a start of clause
- 54 cases remaining contained "*and (SS*"

6.2.1.3 Single tree structure

A problem of speech recognition is the fact that the speaker may pause before finishing the sentence (inappropriate end of speech). At this stage, assume that partial sentences are complete clauses. To cope with multiple clauses, they are linked at a higher level of the grammar.

The implementation of this grammar is written in Nuance GSL (Grammar Specification Language), which utilizes the slot filling concept. When a grammar rule (in this case made of a clause) is hit during speech recognition, variables (slots) are filled with values. These slots are in form of a first semantic interpretation such as "*go=forward*". Slots are the interface variables between the grammar and the application specific software that processes the interpretation. To preserve the order in which the clauses were said during interpretation, the semantic information of each clause is concatenated and passed to the reasoning system through a single slot. The single tree structure of context free grammar preserves order of semantics. This is achieved by structuring the whole grammar into a single parse tree. In other words the context free grammar rules must come together to a single grammar expression at the top. The top expression contains the slot. Now a user can arbitrarily give one or more instructions in one utterance while the order of the instructions is preserved in the order of the slot semantics.

6.2.2 Word Classes

In order to create a grammar that has over-generation only in appropriate places, such as generalizing over numbers or colours, the developer has to create semantic classes for words and phrases. Word-classes are semantic categories. For example the number of

cards (quantity) is a different class to the rank[5] of cards, see appendix A8. Using the same sub-grammar would lead to overgeneration in the wrong place. One could, for example, refer to "1 card" but not to a card with rank "1" since the smallest rank of a card is "2". These word-classes are then also used as a class in the knowledge representation scheme and as language primitive parameters, when passing information from the language recognition module to the reasoning system.

WORDCLASSES

Word Class Name in Grammar	Word Class Name in Knowledge Base	Members
NumberCat	rank	two,three,four,five,six,seven,eight,nine,ten,jack,queen,king,ace
ReturnNumber	n / value	zero,one,two,three,four,five,six,seven,eight,nine,ten,eleven, twelve, thirteen
SuitCat	suit	spades,clubs,hearts,diamonds,spade,club,heart,diamond
LocationN	location_label	"your black area","your hand","my hand","the middle","the table", "the deck", "the green area",...
CardCat	cardname	either or a combination of NumberCat and SuitCat

Table 6-2: Word classes

6.2.3 Full Corpus Coverage & Generalisation

Clauses have been grouped by their language primitive using the context tags during annotation of the corpus. The advantage of semantic grouping is that it ensures correct overgeneration at a local level. Semantic grouping also helps the grammar designer to structure and identify semantics, which initially looks as an overwhelming task when looking at a corpus of thousands of utterances. The definition of word-classes is an easy task for non-specialists in natural language processing. The resulting structure is a grammar template which is easy to convert into GSL grammar. The most trivial solution to convert a corpus into a grammar is to simply copy all transcriptions into the grammar. Hence all clauses become GSL grammar rules. This ensures that the grammar initially covers the whole corpus. This template is the starting point for the grammar designer.
Our aim is to reflect the corpus as accurately as possible. Even colloquial expressions are kept. For example:

```
14.xml/17428-17440: "thats right innit"
```

[5] "rank" refers to the ranking of cards, typically A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2

This is important for real world applications when the robot is used in a household. It is indeed possible to add sentences in good (corrected) English to the corpus, in a controlled way.

The primitives are now attached to each utterance in the grammar, manually, as shown in figures 6-3 and 6-4. This is a labour intensive task. Like the transcription, every utterance is considered. To go through the MIBL corpus to annotate a particular occurrence, such as a reoccurring action, in the text takes approximately 20 man-hours. Assuming annotation has already been done, the all primitives of the same type can be listed together and attaching the grammar is then quick. In the MIBL corpus the primitive "ifloc" occurred 91 times and "move" 163 times. If every grammar attachment takes 1-2 minutes, it is a matter of a few hours to do one primitive type.

Alternatively, this could be done automatically if the XML transcriptions have the utterances already annotated with primitives (semantics). Now, the bespoke word-classes can be substituted to create controlled over-generation. An example on how to achieve correct overgeneration (generalisation) is given below. First a version of the grammar without and then with generalisation:

```
Simple:
(if you have say a five)
(return("context_type=imaginary:" "ifloc=05,+hand2"))

Generalised:
(if you have say a CardCat:c)
(return("context_type=imaginary:" "ifloc=" $c ",+hand2"))

Generalised with Noun-Phrase Grammar:
(if you have say CardCatNP:cn)
(return("context_type=imaginary:" "ifloc=" $cn ",+hand2"))
```

Table 6-3: simplified GSL-grammar showing the generalisation of the noun-phrase "a five". The word "say" refers to the fact that the situation is only imaginary and not happening at this moment. Example is part of 03.xml/3849-3911.

For every utterance a grammar and primitive is written into the transcription. This step in the grammar implementation is a time consuming but easy process. In order to expedite this process, a smart editor could be used that keeps track of word classes and suggests substitutions automatically. The smart editor could also suggest language primitives, based on previous related sentences. A related editor has been suggested by Wang and Acero (2001, 2006) and a graphical grammar editor has been suggested by Monaco (2002), although these do not have the full functionality required for the above task. I have developed the MuTra transcription tool (section 3.4.1) and incorporated a function to call a parser upon a keyboard button. Highlighted text becomes the input to

a custom parser (via batch-file) that replaces the highlighted text with the parser output. This is the basic support for automatic substitution suggested earlier. This functionality was used to test if for instance a grammar has already been defined earlier in the corpus. This constitutes a simple version of a smart editor. If it has been defined the NUANCE grammar testing parser nl-tool would return a result and also produce the necessary slots.

Once the grammar and slots for semantic interpretation have been designed, the output (slots) is ready for reasoning. The first step in reasoning is to resolve references. Here multi-modal information can be used.

6.2.4 Underspecified primitives

A short utterance often results in underspecified primitives have missing parameters. Missing parameters are indicated by question marks ('?').

A natural language parser will hit the interpretation of a shorter sentence before hitting the interpretation rule for the longer (more specified) sentence. Therefore the grammar must be tuned to put well specified grammar rules before underspecified grammar rules.

```
(?then I MOVE CardCatNP:cn onto the table)~1.0
       {return( strcat(strcat("move=" $cn) ",?,table:") ) }

(?then I MOVE CardCatNP:cn )~.99
       {return( strcat(strcat("move=" $cn) ",?,?:") ) }
```

Table 6-4: Underspecified Grammar-to-Primitive matching: The first grammar rule specifies the location "table" while the second is underspecified. A probability reduction is attached to lower its priority in interpretation and the parsing order.

6.3 Overall Grammar Structure

After the procedures described table 6-8 have been completed, the grammar can be combined and compiled. This is an automatic process. This section describes how the grammar is structured, and the parser tools required for automatic combination.

Nuance GSL grammar is a context free grammar, which means that there is only one symbol on the left side and there is one or more terminal or non-terminal symbols on the right side of the grammar rule. Following this structure, the clause grammar of each clause and word classes need to be combined to a complete tree. This calls for a multi layered architecture of the grammar see (table 6-5).

Clause Link Level	← chain of clauses ("utterances")
Corpus Level	← all clauses of the corpus
Clause-Grammar Level	← clauses of the corpus with slot-filling grammar attachment
Phrase Level	← noun phrase grammar, word classes, verb phrases

Table 6-5: Levels of the grammar

6.3.1 Clause Link Level

At the top most level of this clause-based grammar is the linking of clauses (Clause Link Level).

A context free Clause Link Level grammar for combining one, two or three clauses (sufficient long for utterances) is shown here:

CLAUSE_LINK_LEVEL → CORPUS_LEVEL

CLAUSE_LINK_LEVEL → CORPUS_LEVEL CORPUS_LEVEL

CLAUSE_LINK_LEVEL → CORPUS_LEVEL CORPUS_LEVEL CORPUS_LEVEL

6.3.2 Corpus Level and Clause-Grammar Level

On the next lower level are the individual clauses of the corpus, i.e. the Corpus Level. Every clause of the corpus becomes a grammar rule at this level. This ensures that the grammar matches the corpus. This is part of the full corpus coverage design philosophy described earlier in section 6.2.3. At this stage the grammar is essentially a list of corpus

clauses. Even if there are no slots attached to some utterances in the corpus, they are now part of the language model and therefore a text interpretation can be obtained by speech recognition.

Some clauses have had a `<grammar>` tag attached in the XML transcription. These clauses are substituted in the list, so that the grammar from the XML transcription becomes part of the overall grammar. The substitution process is in two stages. First all content of the `<grammar>` tags is extracted (parsed) with `xml-filter.exe`. xml-filter.exe is a parser made for the MIBL project using the Expat[6] library. The extracted grammar is called the Clause-Level Grammar. In order to combine the Clause-Level Grammar with the Corpus-Level Grammar (list of corpus clauses), the Nuance tool `nl-tag-tool.exe` substitutes all clauses in the Corpus-level, that match, with clauses from the clause-level.

6.3.3 Phrase Level

The lowest level of the grammar is the phrase level. It is essentially a collection of word classes embedded in a noun-phrase grammar. A grammar for noun-phrases has been created to describe cards as shown in tables 6-6 and 6-7.

[6] Copyright of the Thai Open Source Software Center Ltd, Clark Cooper,Expat maintainers.
Corpus level grammar building in xml2grammar-test.bat

GRAMMAR	DET.	NOUN	LOCATION	AMOUNT	EXAMPLE
(ReturnNumber:n CardCat:c)	ns	$c	ns	$n	three cards
(DDD CardCat:c)	ddd	$c	ns	?	the seven, these sevens
(DDD)	ddd	?	ns	?	those
(CardCat:c)	?	$c	ns	?	ace
(NumberCat:n)	?	?	ns	$n	take three (missing article suggests number)
(DPD CardCat:c)	dpd	$c	ns	?	your card
(them)	them	?	ns	?	them
(them all)	them	?	ns	all	them all
(the DetAmountOf:n them)	ddd	?	ns	$n	the three of them
(it)	it	?	ns	1	it
(a CardCat:c)	a	$c	ns	1	a five of spades
(DetAmountOf:n DDD CardCat:c)	ddd	$c	ns	$n	all of these cards
(DetAmountOf:n DPD:p CardCat:c)	dpd	$c	$p	$n	all of my cards , any of my cards
(DetAmountOf:n DDD NumberCat CardCat:c)	ddd	$c	ns	$n	all of those two diamonds
(ReturnNumber:n more CardCat:c)	ddd	$c	ns	$n	three more cards
(DetAmountOf:n others)	ddd	?	ns	$n	two others
(DDD DetAmount:n CardCat:c)	ddd	$c	ns	$n	those two diamonds
(DDD DetAmount:n)~.9	ddd	?	ns	$n	those three (cards)
(the ones Location:l)	ddd	?	$l	?	the ones on the table
(DetAmountOf:n the ones Location:l)	ddd	?	$l	$n	all of the ones on the table
(DetAmountOf:n DDD CardCat:c Location:l)	ddd	$c	$l	$n	any of the cards on the table
(DDD CardCat:c Location:l)	ddd	$c	$l	?	those cards on the table
(DetAmountOf:n DPD:1 DetAmount CardCat:c)	ddd	$c	$l	$n	one of your three cards

Table 6-6: GSL grammar of noun phrases. The 4 slots DET, NOUN, LOCATION and AMOUNT are returned to the knowledge base. Later, anaphora resolution is used to resolve what the noun phrase actually refers to. ddd = determinative demonstrative deictic reference, dpd = determinative possessive deictic reference. ns = not specified. The $ sign is passing a variable from a subgrammar. For subgrammars see table 6-7 and Appendix.

```
DetAmountOf
[
      (DetAmount:n ?of)              (return( $n  ))
]

DetAmount
[
      (both)              (return( "2" ))
      (all)               (return( "all" )) ;Universal
      (any)               (return( "any" )) ;Existential
      (ReturnNumber:n)    (return( $n  ))
]

DDD [
      (this) (these) (that) (those) (the)
]

DPD [
      (my)                (return("+hand1"))
      (your)              (return("+hand2"))
      (mine)              (return("+hand1"))
      (our)               (return("ns"))
      (his)               (return("ns"))
      (her)               (return("ns"))
      (its)               (return("ns"))
      (their)             (return("ns"))
]
```

Table 6-7: GSL-grammar, Simplified supporting grammar for noun phrases. For the full phrase-level grammars see Appendix A8

6.4 Summary on Corpus-Based Clause Grammars

The process of creating a grammar from the transcriptions is summarised in table 6-8.

Summary of Procedures

1. tagging of the tasks and subtasks (see section 6.1)
2. identification of primitives (see section 3.5.2, 3.5.3)
3. identification of parameters for primitives and word classes (see section 6.2.2)
4. organisation of word classes into an ontology (see section 6.2.4 on generalisation and 7.4)
5. writing of grammar for word classes (see 6.2.3 and Appendix A8)
6. extension of word classes grammar to noun phrase grammar (see section 6.3.3)
7. adding grammar to clauses in the corpus (table 6-3, 6-4)

Table 6-8: Summary of the procedure to create a clause-based grammar from the corpus

After transcription the utterances are grouped hierarchically by phases of the task and finally utterances are grouped by language primitives through context tagging (table 6-8 Point 1). Utterances are split by a parser so that sequential instructions/clauses are separated. A grammar is generated, containing all corpus sentences. During context tagging primitives are identified (Point 2). At this stage it also becomes clear what the structure of the ontology of the domain could be (Point 3,4). Ontological classes are created, which are then also used as sub-grammars for targeted overgeneration (Point 5) and as language primitive parameters. These sub-grammars are wrapped into noun phrase grammars (Point 6). To link the overall grammar together, all corpus sentences from the transcription are attached with a grammar rule (Point 7). The extraction of these attached grammar rules from the transcription into a complete context free grammar tree is an automatic compilation process and was described in section 6.3.

7 Knowledge Representation of Tasks

7.1 Rationale

Task planning and problem solving was a focus of artificial intelligence, when it was first invented; see for example Newell and Simon's (1956) "logic theorist" or "general problem solver". It was recognised from early on that task planning and execution is at the highest level in the architecture of autonomous systems. A limiting factor of such problem solver systems is that the search space quickly increases with complexity of the task. Although until now such problem solving systems are far from the level of humans, they are perfectly capable of solving simple tasks in household service robot domains. The key is that there are only a limited number of robot capabilities and objects in the environment (i.e. 1 knife, 2 sauce pans, 2 onions, 4 hobs...).

In recent decades a change on focus has occurred towards connectionist and behavioural intelligence. Much progress has been made by employing these approaches, such as neural networks, to perception and recognition. It was suggested by Brooks (1987) that planning should be avoided in robotics completely and the behaviour of a robot should be composed of local behaviour loops connecting sensors and actuators.

However, the same limitations apply also to connectionists, with increasing complexity of the task, the organisation of the information and the computational power causes problems and limitations (Brooks 1990).

In the short term, to make household service robots available, it is best to *combine the strength of both approaches* in a possible layered architecture. Namely using the strength of the statistical and behavioural theories in low-level reactive behaviour and for recognition and simple actuator control. And symbolic reasoning for planning actions and higher level behaviour.

The work in this chapter shows how knowledge can be represented and planning can be used at the top most level in the domain of card games. The challenge is to piece together human instructions in a usable form to carry out the task. We shall first investigate these human instructions.

7.2 Human Level Task Instructions

In corpus-based robotics, the application domain will reveal the instruction primitives and their level of complexity. In the IBL and MIBL corpus, it was found that these 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 3-7 chapter 3.5.2). Teachers always assumed that the student knows what "dealing" is and what the symbols on the cards mean. 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.

In corpus-based robotics, if the application scenario would involve a small child as the receiver of instructions, the future child-robot would only need to be able to reason like a child, since the idea is that the corpus forms the starting point of the design.

A useful future service robot should have an adult-like (experienced) prior knowledge of the domain, for maximum efficiency during instruction receiving. A household service robot is meant to replace a experienced professional adult, the cleaner, butler or maid. A professional attitude would make a service robot usable for industrial and office use, where time is of essence. 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 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. The following sections on Rule Frames, Ontological Reasoning and Problem Solving show how the knowledge is represented in MIBL. An overview of the system is given in Figure 7-1.

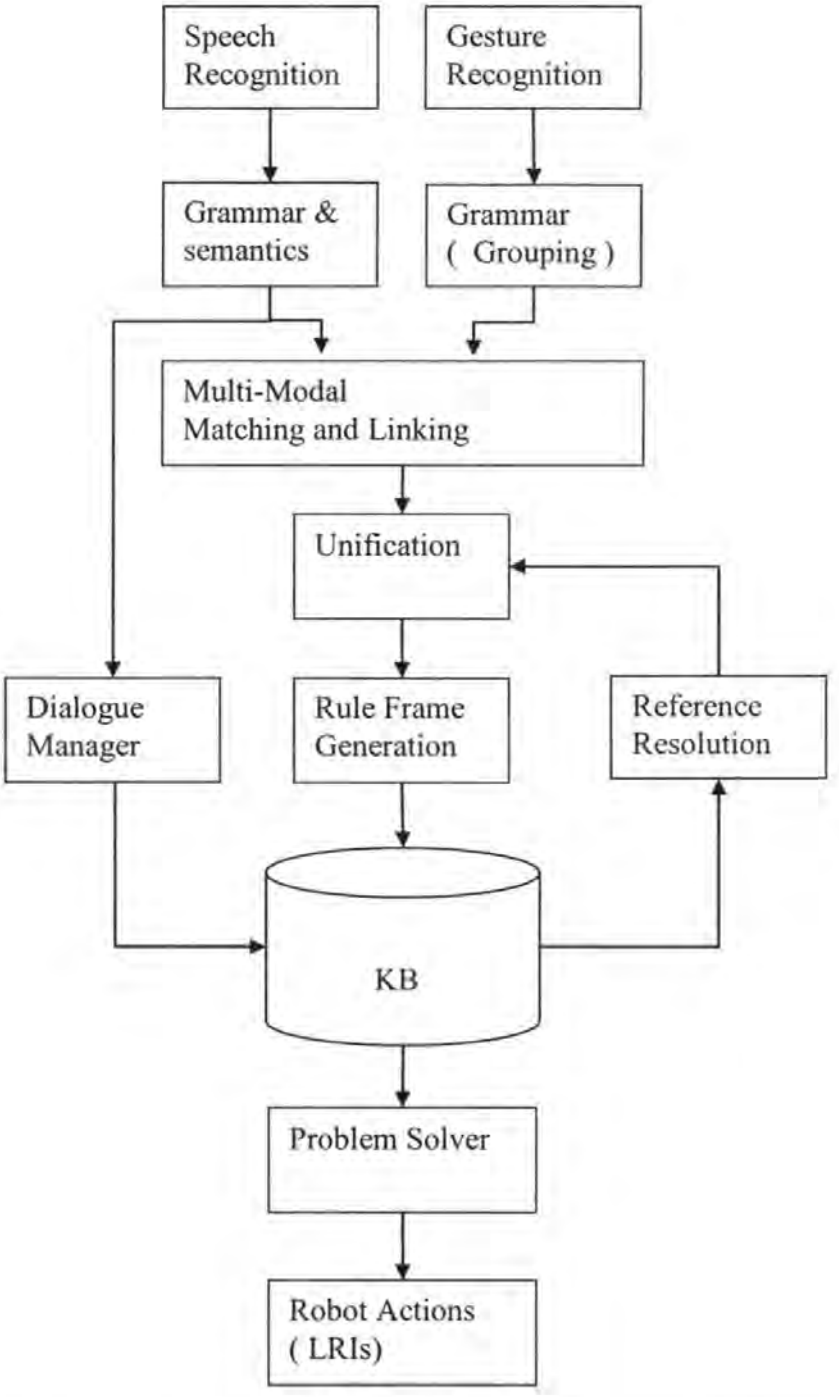


Figure 7-1: Overview of information flow in the MIBL system. Generally the information flow is from the inputs of Speech and Gesture at the top through to the output of "Robot Actions". These Actions include text-to-speech and manipulation of objects or changing the robots attention to initiate new recognition. Grammar has been described in the previous chapter 6 while the multi-modal module was described in chapter 5. This chapter deals with the remaining modules shown in this diagram. Refer to the Chapter titles for an explanation of the modules.

7.3 Rule Frames – an Intermediate Representation

7.3.1 Rule Frames

So far, it has been established in chapter 3.5 that human instructions are presented as primitives and these primitives have to be eventually converted to an executable program. The executable program will be presented to a problem solver for execution. If every primitive were complete and executable, this would be an easy procedure. The nature of human explanations however puts hurdles in the way of making this a straight forward procedure. The hurdles are listed here and examples can be found in the corpus given by references.

Human verbal and gestural explanations

1. include references to previously mentioned instructions and objects
   (03.xml/2540-2563 "you take these into your black area")
2. consist of a sequence of utterances, which belong together
   (03.xml/3919-3948 "...you got a three and a two.. ", belongs to 03.xml/3982-4012 "you take the three and the two because that's equal to five")
3. can be inconsistent or ambiguous
   (07.xml/1231-1254 "and i would win that card")
4. can be incomplete
   (03.xml/2540-2563 "you take these into your black area" leaves questions about how many, where are these, order of movement...)
5. may not be executable in the order they where explained
   (03.xml/3761... explains rule but forgets to say that the card has to be brought forward in the first explanation)
6. contain contextual information (not executable)
   (03.xml/2396-2428 "i will just explain how you deal the cards")
7. contain declarative information (not executable)
   (again 03.xml/3919-3948 and 07.xml/1231-1254)

This calls for an intermediate representation of the instructions between primitives and program, so that all problems and missing information can be resolved before a program is created. This is especially true for complex explanations where the human teacher is prone to making mistakes in the explanation.

To address the problems listed in 1-7 above, it is required to store information given by the teacher in an organised form. If it is in an organised form, the robot knows how to use the knowledge and determine whether any information is still missing. Since all the information is symbolic, it can be stored in a knowledge base. This knowledge base is organised to capture information about and dependencies between:

1. instructions primitives and their parameters
2. contextual information
3. declarative information
4. the robots prior knowledge

The human instruction primitives used in the MIBL project have fixed parameters making it easy to copy all parameters into a predefined frame. Complex rule explanation consist of a series of utterances with instruction primitives that all belong together, referring to a single rule. A frame is used to keep the rule together and in context.

The idea of storing knowledge in frames has been used widely in the past by (Schank 1975; Minsky, 1975; Bobrow and Winograd, 1977), also see section 2.6.3. The first systems such as Winograd's SHRDLU (Winograd, 1971) did only process individual sentences and frames where not required. The introduction of systems that collect primitives into some sort of frame came with the desire to write text story understanding systems for natural language. This started in the mid 1970s, see for example Wilks 1973, Rumelhart 1976, Schank 1975. Although the definition of terms has differed. Schank and Abelson 1977 (See section 2.4.4) report on scripts, which have the same properties as rule frames, namely they contain a sequence of instructions in the same context.

A rule frame must have a context property and at least one instruction primitive.

Complex tasks such as a card game consist of several phases. In the case of MIBL the phases are dealing, game and counting points. The context property (*context indicator)* describes in which phase the rule can be applied. In many cases it consists of more than one instruction primitive. The figure below shows a more complex rule frame for the capture rule in Scopa.

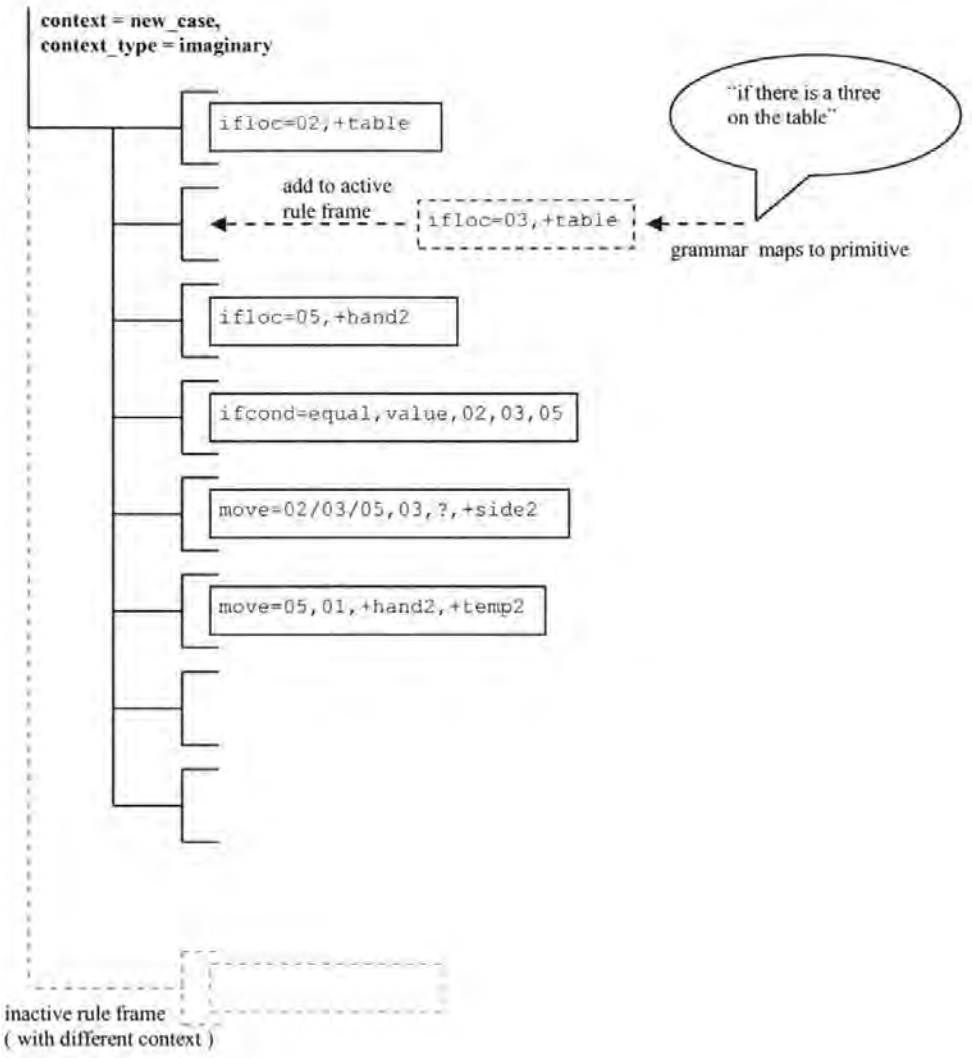


Figure 7-2: Rule Frame, instruction primitives and conditionals are connected together, since the utterances were in the same context. This rule frame is used to construct a function that the robot can carry out to play the game/ perform a task. If a teacher says and utterance it is added to the currently active rule frame. Context primitives can create new rule frames or switch between them. If there are question-marks left, or ambiguities, the robot can clarify information with the teacher before creating the new robot function.

Every time a user says and utterance which contains a instruction primitive, the primitive is added to the currently active rule frame. In general terms, a rule frame is a collection of hints on how the complete rule may be constructed. The problem solver has to determine how the rule works exactly.

A rule frame may contain *conditionals*. These have to be satisfied before the rule can be applied. In the MIBL project the *conditionals* are the primitives `ifloc` and `ifcond`. These typically come from game rules like "if you have a five you can match that five with one on the table…". However *conditionals* are very general and also apply to other

domains, for instance "if there a no more bin bags you need to ...". These conditionals could be compared to (Schank and Abelson, 1977) script header.

Additionally to *conditionals* a rule frame contains and a sequence of *instructions* that have to be carried out. These instructions will be carried out if the all *conditionals* are met. For instance figure 7-2 shows two `move` action primitives, which are the instructions of the rule frame. They will be carried out if there is a "2" on the table, a "3" on the table and a "5" in the hand of the player. Furthermore the condition 2+3=5 must be met by considering the values of each card. But how to generalise the rule and how to infer from the name to rank to value of a card, is not straight forward. This will be described in the next in section 7.4 when discussing ontological reasoning.

Every rule frame and rule frame instruction is given a unique index in the knowledge base (see figure 7-3 "move635"). To every instruction the noun phrase semantics (`np`), the time stamp (`timestamp`) when it was recognised and the type (`instruction-type`) are also stored into the frame.

```
move635
    ├ P: instruction-type=action-move
    ├ P: np=ddd-cardname-ns
    ├ P: object=cardname
    ├ P: timestamp=1347
    ├ P: n=4
    ├ P: source-loc=+table+stock
    └ P: dest-loc=+table
```

Figure 7-3: single Rule frame Instruction as it appears in the knowledge base

7.3.2 Scope of the Rule Frame notion

Since the idea of storing knowledge in frames has been used widely in the past, it can be hypothesised that rule frames work for many different domains, not only for the card-game Scopa. A different domain merely requires the implementation of different primitives. With the currently implemented primitives, one could already teach variations of fishing-type games, where some cards need to match in order to score.

7.4 Applied ontological reasoning

The exact structure and implementation of the ontology used by the MIBL robot is described next. For an introduction to ontologies, see section 2.6.4

7.4.1 Implementation of Ontology

The robot has an innate prior knowledge of playing cards and their properties. Properties are attributes of a class. Classes are structured in a tree taxonomy. For example a playing card is a 3D-object. A 3D-object has coordinates. See figure 4.

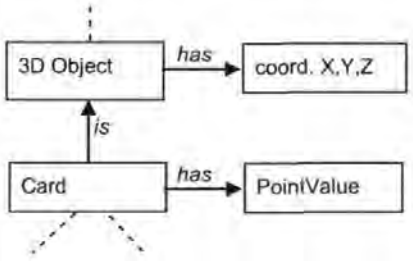


Figure 7-4: Extract of the ontology of cards for the MIBL reasoning system

The modelling of this ontology closely follows Smith (2006) and Spear (2006), who suggest ontology design based on Basic Formal Ontology (BFO). Essentially the `is_a` relationship indicates classes and sub-classes. A "Card" is a "3D Object". Now, classes can have properties. For example a "Card" often has a value in points, in a card game. Relations to properties are called `has_a` relationship indicates properties. Properties are inherited. For example "Card" inherits the property of coordinates (see figure 7-4). Currently multiple inheritance is not allowed. However, multiple ontological trees can be combined. Properties can be filled with values, which are defined as classes themselves. For example, a house has_a colour, whereby colour is defined as a class itself. Its subclasses are green, blue, yellow…

The implementation of the knowledge base, which includes the ontology as its backbone was done in Prolog[7]. Prolog is a declarative language, which makes the implementation simpler. Prolog allows the addition of code during runtime. New `is_a` and `has_a` relations are created as the robot learns. See below an extract of the MIBL code:

[7] the Sicstus Prolog 3.10 engine was used and interfaced from C. Prolog is based on first-order logic. It is a declararive language such as SQL or LISP.

```
is_a('face_card','rank').
is_a('cardinal','rank').
...
is_a('JJ','face_card').
is_a('QQ','face_card').
...
has_a('object3d','X').
...
has_a('cardname','rank').
has_a('cardname','suit').
...
```

Table 7-5: extract from the implementation of the ontology in Prolog

For further reading on Prolog, see section 7.9.2 . Actual cards that are present on the playing table are instances of classes. Because of the nature of the implementation, where the name of a class is the index of the table in the database, a class name must be unique.

```
is_a_instance('S/AA','cardname').
```

Instances inherit all properties from the class they are derived from. Classes are templates for instances. Instances are not part of the tree taxonomy directly. They are copies.

This representation system allows storing of the current situation and factual knowledge about the robots world. The previously described primitives that refer to a fact are manipulating this knowledge. For example the fact that the eight has been removed from the game translates into the Prolog statement:

```
forall(
   get_property(INSTANT,'rank',08),
     (delete_instance(INSTANT))
).
```

The "value" of a property is the terminal symbol in the created ontology.

```
property_data(INSTANCE_NAME,PROPERTY,VALUE).
property_default_data(CLASS,PROPERTY,VALUE).
```

Default property values is innate knowledge of the prototype of the object. For example a playing card "2" usually has the value of 2. Or a spades is "black". The idea of prototypes to represent initial assumptions has been used many times when more

specific information is absent. For example (Minsky, 1975) or Schank's Dependency Primitives (Schank and Abelson 1977) . Winograd also defines prototypes in KRL (Bobrow and Winograd, 1977).

Table 7-1 presents the ontology implemented for MIBL not showing instances.

```
root
  object3d
    P: X=
    P: Y=
    P: Theta_Y=
    P: location_label=
    cardname
      P: value=
      P: rank=
      P: suit=
      suit
        S
        C
        D
        H
      rank
        face_card
          JJ
          QQ
          KK
          AA
        cardinal
          02
          03
          04
          05
          06
          07
          08
          09
          10
  context
    P: current-context-type=
    P: current-context-name=
    raised-issues-stack
      P: current-issue=none
  dialogue
    P: state-vector=[greeting,none,human,none]
  coordinate
    X
    Y
    Theta_Y
  location_label
      +table
        P: state-vector-index=2
      +temp1
        P: state-vector-index=5
      +temp2
        P: state-vector-index=6
      +table+stock
        P: state-vector-index=1
      +stock
        P: state-vector-index=1
      +hand1
        P: state-vector-index=3
      +hand2
        P: state-vector-index=4
yes
```

Table 7-1: MIBL Ontology, without instances

Example of a Rule Frame as it is stored in the MIBL knowledge base using the ontology functions:

```
3          Rule217
3            P: phase=playing
3            P: timestamp=3554
4             IfLoc635
4               P: instruction-type=ifloc
4               P: np=a-KK-ns
4               P: n=1
4               P: location=+hand1
4               P: object=KK
4               P: timestamp=3761
4             IfLoc763
4               P: instruction-type=ifloc
4               P: np=ddd-KK-ns
4               P: n=?
4               P: location=+table
4               P: object=KK
4               P: timestamp=3761
4             IfCond399
4               P: instruction-type=ifcond
4               P: type=equal
4               P: compare=value
4               P: lhs=[a-?-ns,1]
4               P: rhs=[[a-?-ns,1],[?-?-ns,?]]
4               P: timestamp=3761
4             move240
4               P: instruction-type=action-move
4               P: np=ddd-cardname-ns
4               P: n=1
4               P: source-loc=+table
4               P: dest-loc=+hand1
4               P: object=_829
4               P: timestamp=3795
```

Table 7-2: MIBL Rule Frame From the MIBL Corpus 03.xml / Pairing rule.
The rule frame "Rule217" with 4 instructions.

7.5 Anaphora Resolution

Antecedent anaphora are references to explicitly mentioned nouns, earlier in the discourse, see Grishman (1986). According to Grishman (1986), anaphora resolution is one of the most difficult problems in Natural Language Processing.

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 noun phrase grammar was defined to identify anaphora such as DDD, DPD. The information produced by the noun phrase grammar is shown in table 6-6 and 6-7. The grammar returns a tuple of information about the noun phrase consisting of:

```
[ Determiner , Noun , Location , Amount ]
```

This semantic information can be used to resolve the references. If a new object is introduced to the conversation, the noun phrase contains the determiner "a" or no determiner at all[8]. In this case no resolution is required. The noun phrase semantics are stored into the rule frame. If in the next utterance a DDD occurs, it is assumed that the noun phrase refers to the previously stored noun phrase. The implementation of the anaphora resolution is a series of rules like the one, just mentioned[9].

Every noun phrase tuples are stored in the rule frame, see table 7-2 property "np" for noun-phrase. The noun phrase tuple finds its way from the utterance to the rule frame as instruction primitive parameters. If the resolution algorithm is confronted with a DDD, the resolution is achieved by retrieving the previously stored noun phrase. The resolution process accesses the linguistic clauses in their formal format as rule-frame instructions like shown in table 7-2.

Here an anaphora resolution algorithm tries to identify the reference by looking at the previous instruction. It will jump over and ignore sub-dialogues and utterances which do not have instruction semantics. 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

[8] in the grammar implementation, no determiner has the symbol "ns" for not specified
[9] see code ref_res.pl in Appendix

turned over, how many or where they are. With anaphora resolution referring to a previous sentence, this information is recovered. Anaphora can be resolved by looking at two sources: gesture or to a previous utterance. In the MIBL software, gesture was not considered, future work could include gestures. References to resolve the noun-phrase from gestures is considered through multi-modal integration instead.

ANAPHORA RESOLUTION RULES APPLIED IN MIBL

Current Clause		Previous Clause		Case Description	Resolution Action
Det.	Object	Det.	Object		
a				introduction of new object and possibly new context	use object given in current clause
REF	'?'	a		reference to a previous object that is first introduced in the previous clause with "a"	use object given in previous clause
REF	not '?'	a		reference to a previous object that is first introduced in the previous clause with "a"	use object given in the current clause
REF	not '?'	ns		reference to a previous object that is first introduced in the previous clause without article	use object given in the current clause

Table 7-3: Anaphora Resolution Rules. For resolution an object must be referring back to its first instruction into the conversation. Clauses are dealt with in the formal rule-frame instruction format.
REF is ddd,dpd,them,it, or '?'

Once the previous rule-frame instruction is identified using the table above, the object of the conversation can be recovered. Further information about completing the new rule frame can be recovered as well. To come back to the example "turn them over", the question of how many cards need to be turned over can be identified by looking at the previous rule frame, if it also contains a parameter "n".

7.6 Unification

The process of multi-modal integration has been described in chapter 5. Figure 7-1 shows how unification follows after the multi-modal integration system. It is often regarded as being part of the multi-modal integration process with the last step in the integration being unification. Unification is the process of combining (unite) semantic information from different sources. The concept of unification exists in grammar (Shieber 1986) and in logic programming (Bratko 2000). In multi-modal systems, unification is performed between language and gesture primitive parameters.

The following 4 cases can occur as a result of unification:

UNIFICATION OF LANGUAGE PRIMITIVES

Case	Number of Solution	Description	variables
Completion	$n_s = 1$	A gesture and an utterance are individually incomplete, but complete each other.	all variables are resolved
Confirmation	$n_s = 1$	A gesture and an utterance are individually complete. When combining they match.	no variables exist
Contradiction	$n_s = 0$	A gesture and an utterance contradict each other, no solutions, unification fails	
Under-specification	$n_s > 1$	The gesture and language combined are still semantically underspecified. Therefore several possible candidates are returned	

Table 7-4: Unification possibilities. Where n_s is the number of solutions

In MIBL, unification is used for move, turn and pointing primitives. Parameter by parameter is compared between the language version and gesture version of the primitive. Each time the outcome is one of the 4 cases. This is known as a tautology, with action attached to all outcomes. In fact it is good practice to program algorithms in a fashion where each "if" statement needs an "else" statement so that the all cases combined are a tautology. This helps the developer to clarify its own mind and proves the algorithm covers all possible cases. This technique has also been used when designing the multi-modal integration algorithm.

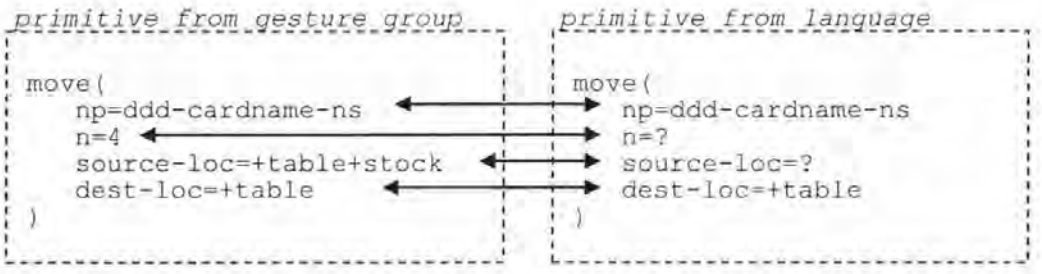


Figure 7-5: Unification process, missing information of the language primitive "deal the cards onto the table" is unified with the gesture group that was selected from multi-modal integration. Parameter by parameter is compared with the algorithm shown in figure 7-6. The implementation is in Prolog.

If the multi-modal integration algorithm firmly believes that gesture and language need to be unified, the selected candidates go through the unification process described in the flow chart below. Each primitive has parameters, and the parameters are unified individually. In the cases for under-specification or contradiction, a question can be raised to the user to obtain missing information, also see flowchart figure 7-6 with instruction "Raise question". Unification with integrated question to the user is also described by (Kruijf *et al.*, 2008). Kruijf argues that robots can only understand the world around them by deliberately planning clarification questions. In the MIBL system this is realised by the Issue stack described in section 7.10.2.

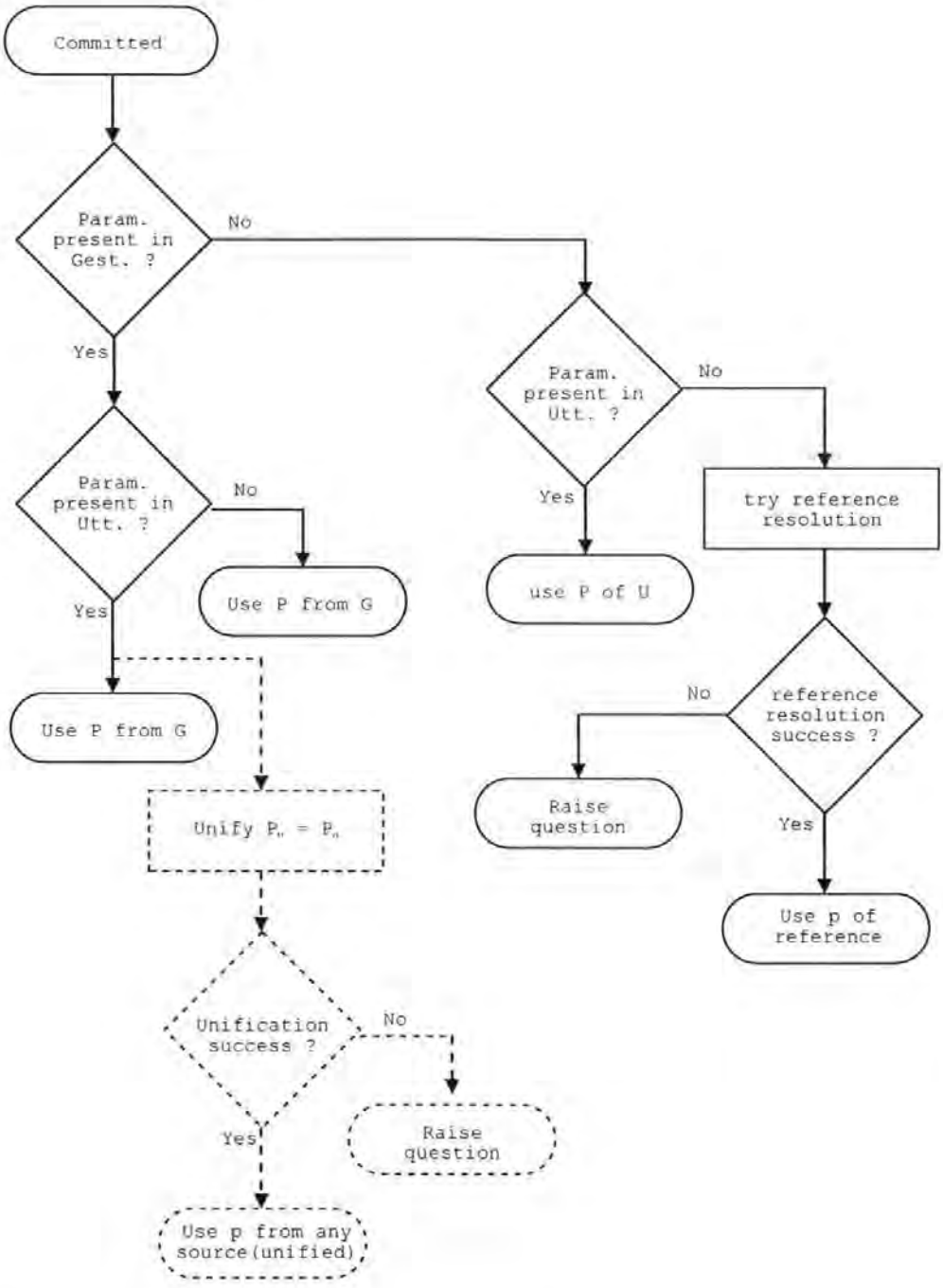


Figure 7-6: Process for unification of primitive parameters. Since speech recognition errors are more common than gesture recognition errors, the process believes gestures. An alternative method is shown in dashed lines. P=Parameter of a primitive, G=Gesture, U=Utterance

The unification algorithm has been tested through an integrated test by playing back the corpus, see experiment E3.1, E3.2 in chapter 8.5-8.6. The algorithm uses a conservative approach, it asks the user if there is uncertainty, thus its performance is trimmed to avoid false information to be stored in the knowledge base.

7.6.1 Mirrored Location References in Utterance and Gesture

A special case of primitive parameters is the location-parameter. A teacher explanation often needs to be mirrored to allow the robot to take the teacher's perspective. Or in general terms, the robot needs to imagine being the teacher. This needs to be considered when comparing location-parameters between gesture and language as well, because teachers used "you", "me", "yours"... interchangeably without paying attention. Therefore, by default, any location that is in-front-of the teacher must be mirrored to in-front-of the robot.

7.7 Initiation of Learning

7.7.1 Timing

Lower layers of a robotic system continuously receive sensor data and low level recognition processes often produce outputs at a high frequency, for example images are produced at a frame rate, in the case of MIBL, gesture data is produced with 10 Hz and low level recognition of actions can therefore output several times a second. The question is when is the best time to process this now symbolic data further for the unification, learning of rules and actions? This point in time t_0 is limited to a minimum. Too early processing take into account incomplete gestures or utterances, too late processing (>10sec) will slow down and confuse the human-robot dialogue (not anymore real-time). The following limitations constrained and defined the point in time t_0 to initiate learning:

- a gesture group has finished: $t_0 = t_{End\text{-}of\text{-}Gesture} + t_{End\text{-}of\text{-}Group\text{-}Timeout}$
- an utterance has ended, and no gestures followed:
  $t_0 = t_{End\text{-}of\text{-}Speech} + t_{SOG\text{-}to\text{-}EOS\text{-}Max} + t_{Max\text{-}Gesture\text{-}Length}$
- A gesture has ended: $t_0 = t_{End\text{-}of\text{-}Gesture}$

The criteria and timeouts shown are taken from the timing histograms on multi-modal integration (Chapter 5.2.4 figures 5-7,5-8). A scheduler algorithm manages the calling of the unification and learning algorithms at the calculated time t_0. The scheduler algorithm performance was tested as part of the multi-modal integration system in chapter 5.3.2.

7.7.2 Overview of the unification and learning algorithm

1. consider most recent two unprocessed speech events
2. establish relationship between clauses (section 6.2.1.1) to check if a primitive is modified or if a new primitive needs to be created in the knowledge base
3. process new knowledge primitives
4. process new action primitives:
   - 4.1 group gestures
   - 4.2 filter out unfinished groups, reschedule t_0 if unfinished group is discovered
   - 4.3 matching, linking and unification of action primitive with gestures
     (on success, add action primitive to rule frame)

if unification failed try learning without regarding the gestures

if a loop-indicator such as "each" is in the language multiply the new instruction

5. store information in rule-frame

The performance of the multi-modal integration (except step 4.4) was presented in chapter 5.3.2.

On a context change, i.e. a new rule is completed; the rule-frame is translated into a state-transition-rule, which completes the learning of the rule. Therefore the learning is an on-line process.

7.8 Problem Solver

7.8.1 Overview

Section 2.6.5 gave an introduction to problem solvers. The MIBL robot plans the application of a rule frame before carrying out the primitives. This is achieved by applying the rules that it learned by using a problem solver. On its turn in the card game, the robot looks at the cards and their position (`STATE_VECTOR_LIST`). This forms part of the initial state. The game rules that are made of rule frame instructions are applied as state transitions. The goal of the problem solver is to apply a learned rule successfully. The robot tries to apply the rules to different combination of available cards. The combination of possible cards and game rules form the search space of the problem solver. It is a dynamic environment, different in every game and depending on the rules. The IBL project proposed a planner to test the route through the city. In both projects, the planner helps the robot to check if the learned knowledge fits together to produce an output program that will complete the task successfully.

7.8.2 from Rule Frame to State Transition Rules

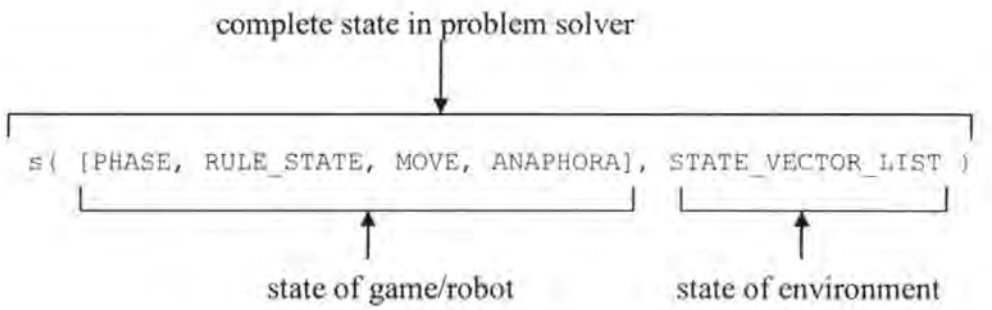
The rule frames are wrapped into state transition rules (STR) to allow the robot to predict the outcome of its actions. State transitions are the required building blocks for a problem solver. State transitions are a common tool in robot task execution; see for example (Lopes *et al.*, 2003). A state transition rule consists of the entry state, rule frame instructions (RFI), low-level robot instruction (LRI) and the exit state.

```
str( entry_state, exit_state, LRI) :- RFI
```

Since the output from state transition rules is low-level robot instructions (LRI), the STRs are a form of translation between human-level instructions and robot instructions. The complete state of the problem solver consists of two parts: the state vector list describing the environment and the game/robot state vector.

On the robots turn the robot is initially in the following state:

```
s( [PHASE,initial_state,_,[]], STATE_VECTOR_LIST )
```



when trying to apply a rule the robot goes from

```
s( [PHASE,CURRENT_RULE,LAST_INSTR,[TUPLE_LIST,MATCHED_LIST]],
STATE_VECTOR_LIST )
```

to this state

```
s( [PHASE,CURRENT_RULE,INSTR,[TUPLE_LIST,MATCHED_LIST]],
STATE_VECTOR_LIST_OUT )
```

and finally, when all instructions of the rule frame have been successfully applied, the problem solver achieved his goal. The goal is defined in the problem solver by investigating the solution path and making sure that all instructions of the chosen rule frame are found in the solution path.

For example: CURRENT_RULE could be "Rule217" and LAST_INSTR and INSTR are actions from a rule frame, for example "move240" from table 7-2.

A key point of the system design philosophy is that 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.

The "PHASE" in the state vector describes the wider context. It is used to guide the problem solver to use only the appropriate rules for the given context. For the card-game system 3 contexts have been used: "dealing", "playing" and "imitation". The context switching is handled by the dialogue manager. "imitation" is a special, in a sense because it is used to temporarily do actions imitating the teacher.

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 collects 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.

7.8.3 Micro Planner

Every state transition in the MIBL planner (problem solver) a primitive is applied. In some cases an action primitive can describe a procedure that actually requires several actions, such as utterance 06.xml/709-738:

text: `"and then we put four cards in the middle".`

primitive: `move(ns-cardname-ns,04,?,+table)`

This primitive would require 4 pick and place moves. The possibility of doing these 4 moves is tested by the micro planner. The micro planner carries out the action (move or turn) card by card. It generates low-level robot instructions (LRIs) for every move, naming the specific objects involved.

7.8.4 Generalisation and References in Rules

It was found from the corpus that complex tasks are often explained using an example, which means that the robot needs a generalization mechanism. A reason for this explaining by giving examples could be that it becomes easier to reference to the various different objects (in this case cards) involved. 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 difficult for a robot to find what the salient features of the situation are and it can therefore not easily induce a rule. This is called the "frame problem" (McCarthy and Hayes, 1969) and is one of the limitations that remain unsolved in artificial intelligence. The frame problem can only be resolved by pre-programming the right actions in the context that the robot encounters, in this case card games.

It is possible to rely on so called "hasty induction" of the rule without proves, see (Flach *et al.*, 2006). 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, which is in this example named 5". This may be how humans learn game rules, and they are often stopped later on by the teacher during test games 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.

Below is an extract from the logfile that was written when trying to apply the pairrule after learning it from transcription 15.xml :

```
do_ifloc/5 passed for: IfLoc635 with item: S/QQ
do_ifloc/5 passed for: IfLoc763 with item: C/JJ
do_ifloc/5 passed for: IfLoc763 with item: C/QQ
do_ifloc/5 passed for: IfLoc763 with item: D/05
do_ifloc/5 passed for: IfLoc763 with item: H/QQ
TUPLE_LIST = [[S/QQ,JJ,4],[C/JJ,JJ,2],[C/QQ,JJ,2],[D/05,JJ,2],[H/QQ,JJ,2]]
find_references/3 MATCHED_LIST: [[S/QQ,lhs,9,4],[S/QQ,rhs,9,4],[S/QQ,rhs,9,4]]
find_references/3 MATCHED_LIST: [[S/QQ,lhs,9,4],[S/QQ,rhs,9,4],[C/JJ,rhs,8,2]]
find_references/3 MATCHED_LIST: [[S/QQ,lhs,9,4],[S/QQ,rhs,9,4],[C/QQ,rhs,9,2]]
do_comparison/3 (equal,value): ifcond match equal,value passed S/QQ=C/QQ
do_instruction/6: TYPE=action-move INSTR=move240 OBJECT=_4577
LRI=[move(C/QQ,+table,+hand2),say(i can move this card)]
```

Table 7-6: Log file from Rule Application From the MIBL Corpus 15.xml / Pairing rule

7.8.5 Game Strategy

As stated earlier, the problem solvers goal is to successfully apply a learned rule (goal state: `s([...,done_rule,...,...],...)`). This is the end of the robots turn to play. The robot makes no attempt to plan the game until the end, because it would have to guess what cards the other players have or what is next in the stock pile. This is called a partially observable game.

The robot has no preference on which rule to apply first or which card to prefer. It there is more than one solution from the problem solver, it will just apply the first solution and discard the others, unless it failed to execute the first solution.

Adding a preference which solution to choose is a matter of game strategy. Adding a preference reduces the search space. Reduction of the search space was not required for the application of the Scopa card game rules of the MIBL project.

However, without any optimisation at all, the problem solver can easily get stuck in an endless loop. If two instruction primitives reverse their effect, putting a card on the table

and picking it up straight after, just to show your opponent what you have got, is a typical example. The problem solver may suggest that these actions have to be carried out several times which is actually not true.

Some tasks that have a large amount of rules and physical objects in them can not be solved without an optimisation. The computational expense of the micro planner is doubled with each additional object or rule.

7.8.6 Low Level Robot Instructions

LIST OF LOW LEVEL ROBOT INSTRUCTIONS

LRI	parameters	description
move	(%1,%2,%3)	robot moves a single object %1 from location %2 to location %3
turn	(%1,%2)	turn object %1 around %2 degrees
shuffle	()	shuffle all the cards and return them to the stockpile
say	(%1)	say %1 using the text-to-speech system
grammar	(%1)	swap recognition grammar to %1
system	pause	stop the time in the system (used for question answering in corpus-playback)
exit_program	()	shutdown the robot
dm	schedule_ gesture_ recognition %1	the recognition system (unify-and-learn) predicts that attention for further recognition and processing (unify-and-learn) is needed at time %1 in the future
none	()	Do nothing for the moment

Table 7-7: Low Level Robot Instructions

7.9 Advantages of Implementation in Prolog

7.9.1 Logic Programming

Prolog is based on first-order predicate logic, making it easy to describe facts and relationships. Prolog is a declarative programming language, in contrast to imperative programming languages. In declarative programming, statements are descriptions of a logical goal, rather than stating instructions on how to carry out the search in the knowledge base. One can declare the logical goal and the Prolog engine will carry out the search by trying to resolve the unknowns in the query. This resolving process is the core of the Prolog engine. It incorporates SLD[10] resolution (Linear resolution with selection function in definite clauses), see Kowalski and Kuehner (1971) for detailed explanation.

For instance, if a Prolog program consists of the single statement

animal_name('cat1', 'maumau').

and one would like to write a program to find the name of 'cat1', then this is simply done by writing

animal_name('cat1', N).

Prolog will answer

N = 'maumau', yes.

Prolog Unification. Prolog attempts to prove *animal_name('cat1', N).* by attempting to find and unify all "knowledge" in its database with this term. In this case there is only one relation in memory, which matches. This matching process uses unification, which means that the goal and the knowledge is unified, resulting in the unification of N = 'maumau'. Prolog will answer 'yes' when a logical goal has been successfully proven.

7.9.2 Storing Knowledge in Prolog

In order to represent knowledge in Prolog, one can simply state facts and their relationships. For example *colour('cat1' , black).* is a valid Prolog statement. These relationships were utilised to organize knowledge in a hierarchical tree-taxonomy. The modelling of this tree-taxonomy is inspired by Smith (2006) and Spear (2006),

[10] SLD resolution stands for **S**elective **L**inear **D**efinitive clause resolution

mentioned in section 7.4 . Governed by *has-a* and *is-a* relationships a tree of objects and their properties can be devised which constitute the robots knowledge base. These are easily expressed and can be queried in Prolog, see this example of a Program:

is_a('+table','location_label').
has_a('object3d','location_label').

7.9.3 Search Space Reduction

An inherit problem of logic programming is that the amount of computation time required can grow exponentially (depending on the algorithms used) with the number of symbols in the knowledge base.
Humans use context and attention mechanisms to filter out unimportant information, this principles was applied here. Attention mechanisms are applied to reduce the search space. In the case of MIBL, only the latest two utterances are considered for processing, which is in effect focuses attention to recent events and therefore reduces search space. Furthermore the `context` branch of the knowledge base is a pointer to the currently active rule. This reduces search space again.
The robot has a memory that remembers all gestures that have occurred. The gestures that have been used for learning already are marked. During a context change all past gestures are marked as used and won't be used again for learning.
In practice, learning rules of a single card game does not represent a computational burden in terms of search space of the knowledge base. There are the 52 cards all past language and gestures and rules in the robots knowledge base, which make up hundreds of symbols in total. It can be hypothesised that a single household task of future service robots will not need more symbols and data than in a card game.

Considering livelong learning however, the robot should delete used gesture and language events to reduce the search space. Rules should be stored on a permanent memory and loaded depending on context.

7.10 Dialogue Manager

In natural language processing a mechanism that detects and guides the discourse of the conversation is the dialogue manager. Traditionally, the dialogue manager is a state engine of some form (Spiliotopoulos *et al.*, 2001). Modern natural language tools such as Nuance NVP Builder allow editing the state engine of a dialogue manager as a flow chart diagram.

7.10.1 "Choose 1. 2. 3. or 4. for an operator"

Question-answer dialogues are used in robots and telecom natural language systems. Typically a user is given a choice of menu points. (Spiliotopoulos *et al.*, 2001) is a typical example how question driven dialogues are now being applied to robots, which interrogate their users. There is an argument that these systems work, because the human user has to adopt to the systems needs, but if this is true, why do most people choose 4 for an operator? Dean (2008) showed in a study titled "What's wrong with IVR self-service" that speech recognition in telephone applications is introduced to the benefit of the company rather than the user and that users prefer human operators. This possibly has implications for the social acceptance of speech recognition robots.

In a real human-to-human teaching scenario, however, there is no interrogation of the teacher in the corpus, there are however clarification questions after an explanation from the teacher. The teacher has the initiative in the dialogue, not the learner. The teacher chooses the topic and explains freely. Usually the student has a chance to interrogate the teacher at some point if questions arise. These student questions are usually only few and near the end of the teacher's explanation. Therefore the design principle must be:

When the teacher is teaching a task, the robot has to listen like a student.

The state engine of the dialogue is therefore a passive one, whereby states are changed by teacher utterances. The robot will only reply with "ok" to the teacher's instructions and otherwise say nothing. When after multi-modal unification a question arises from

the instruction, the robot will ask the question. After the question has been answered by the teacher it goes back into the mode with passive replies of "ok".

Even the robot usually does not have the initiative it still needs a dialogue model for understanding instructions. A dialogue state vector is used for this purpose.

Dialogue State Vector:

```
[MODE , PHASE , TURN , ISSUE]
```

7.10.2 Issue Stack

(Kruijf *et al.*, 2008) argues that robots do not fully understand the environment they are situated in. At different stages of its reasoning the robot can become aware of missing information. Frame-based reasoning provides the ideal basis for this problem, since missing information simply means unfilled slots in a rule-frame instruction. Since more than one of these unfilled slots can appear, the questions to the users are put on a stack. A "issues raised stack" has also been suggested by (Lemon *et al.*, 2001) in the WITAS dialogue system.

```
09.xml/teacher/2318-2396: "ok so if you cant go i mean if you cant capture a card then
you still have to lay one down you always have to lay a card down"
09.xml/student/2402-2413: "what from the main pack"
09.xml/teacher/2410-2439: "no from your hand of three"
09.xml/student/2443-2457: "where do i lay it to"
09.xml/teacher/2452-2474: "onto the table"
09.xml/student/2472-2481: "ok"
```

The ISSUE state in the dialogue state vector can go through the following states:

none → raise_question → wait_answer → none

As described earlier, questions can arise from missing information in rule frames. These questions are missing or ambiguous parameters in rule-frame instructions. A table below shows how these parameters are connected to questions to the teacher.

QUESTIONS TO THE USER

primitive	parameter	current state	text for text-to-speech engine (TTS)
move	dest-loc	unknown	'where do the cards go?'
move	source-loc	unknown	'where should i take the cards from?'
move	n	unknown	'how many cards would you like me to move?'
turn	n	unknown	many cards would you like me to turn over?'
turn	source-loc	unknown	'where are the cards that i should turn?'

Table 7-8: Questions to the user. Questions that the robot can ask in order to clarify information in a rule frame.

7.11 Summary on Knowledge Representation

This Chapter showed how natural language instructions, that have been converted through a grammar, can be reasoned with. The approach uses frame-based reasoning, embedding all knowledge into a structured knowledge base. Every instruction is part of a ruleframe that sets the context. These frames then become state transition rules so that the robot can plan and predict the consequences of its own actions. It was shown that missing information, that can be discovered through ambiguity or incomplete rule-frame instructions can be dealt easily discovered and requested from the user. In the next chapter, the framework will be put to the test.

8 Test and Evaluation

8.1 Test and Evaluation in Corpus-Based Robotics

The Corpus-centred approach to robot design, introduced as Corpus-Based Robotics, involves an experiment to collect a corpus at the beginning of the project, labelled in this chapter as E1. The robot is designed and implemented based on the corpus. This corpus collection experiment provides a rich source of data to carry out tests and evaluation. If the corpus-based design has been successfully applied, the robot should perform well when tested with the corpus. In fact, if human error in the task explanations is eliminated, the robot should recognise every task explanation successfully. It is an uncompromising test method of the implementation..

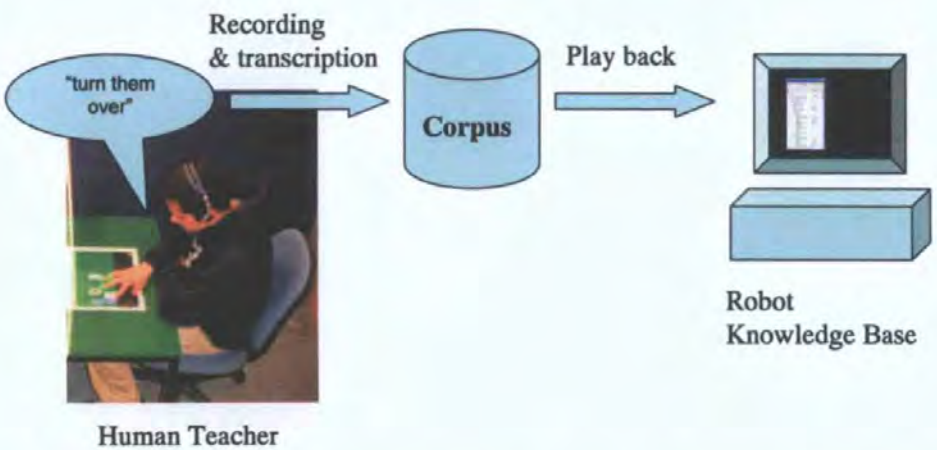


Figure 8-1: Using the corpus for testing: A form of test and evaluation is to play back the recorded teacher voice and actions to the robot, in this case a software agent.

Besides the experiments with recorded data, experiments with live (online) communication between the robot and a human teacher were conducted. These are the experiments labelled E4.1 and E4.2. . Before going into detail about the experiments, lets lay out the MIBL system and how it can be analysed.

8.2 Error Categories

There are many ways to measure a system's performance. In the case of testing MIBL it is of interest to make sure that *the robot learns and executes the instruction correctly.* This can be tested for every instruction the teacher gives. Every attempt of the teacher is indicated with <attempt> in the transcription for analysis. An attempt is defined as the teacher trying to say a game-rule to the robot in a single utterance. An attempt usually consists of a single utterance translated by a single primitive.

If there was no error and the rule was learned correctly the attempt is accompanied with the tag <error_none> in the transcription. If this is not the case, an "error" has occurred along the way. These errors can be put into categories according to the stage where they have occurred. The result shows where the system needs improvement and further research. Errors can cause a chain reaction of further errors, therefore the error of interest is the *first fatal-error* when following the information from the teacher through the system. The table 8-1 shows all error tags used for analysing experiments E4.1 and E4.2 on human-robot communication.

TABLE WITH TYPES OF ERRORS

XML-Tag	Level	Description
<error_ti>	Human	Error in the Teacher's instruction
<error_tgiveup>	Human	Teacher gives up with teaching instruction
<error_tdm>	Human	Wrong dialogue move of teacher
<error_se>	Recognition	Speech Recognition fails to recognise the utterance, eventhough the utterance is in the grammar and the HMM is therefore trained for it.
<error_ge>	Recognition	Gesture Recognition fails
<error_oosg>	Categorisation	Utterance not in the Speech Grammar (Out of Grammar Error)
<error_oogg>	Categorisation	Gesture not in Gesture Grammar, in other words, grouping of gestures is not correct (Out of Grammar Error)
<error_mmlink>	Categorisation	Wrong matching and linking between speech and gesture
<error_unif>	Categorisation	Wrong decision in unification or failed to unify primitive parameters
<error_dm>	Reasoning	Wrong dialogue move of robot
<error_ana>	Reasoning	Wrong decision in anaphora resolution or anaphora resolution failed
<error_kb>	Reasoning	Incorrectly stored instruction, for example in the wrong context
<error_exec>	Execution	Failure to execute instruction
<error_none>	-	Instruction had no error
<attempt>	-	attempt by the teacher to teach a game rule. This attempt usually consists of a single utterance containing a single primitive. Every attempt ends with an error-category or error_none. Number of attempts divided by number of error_none tags gives the success rate.

Table 8-1: List of types of Errors

In IBL, the error analysis was carried out in a similar fashion. The errors types were split into the categories Human Error, Speech Recognition, Semantic analysis, Functional Limitations and Execution problems. The fatal error per instruction was categorised. To better understand the information flow from the human speech through speech recognition, categorisation mechanisms such as multi-modal integration through to reasoning and execution, a diagram is given in figure 8-2.

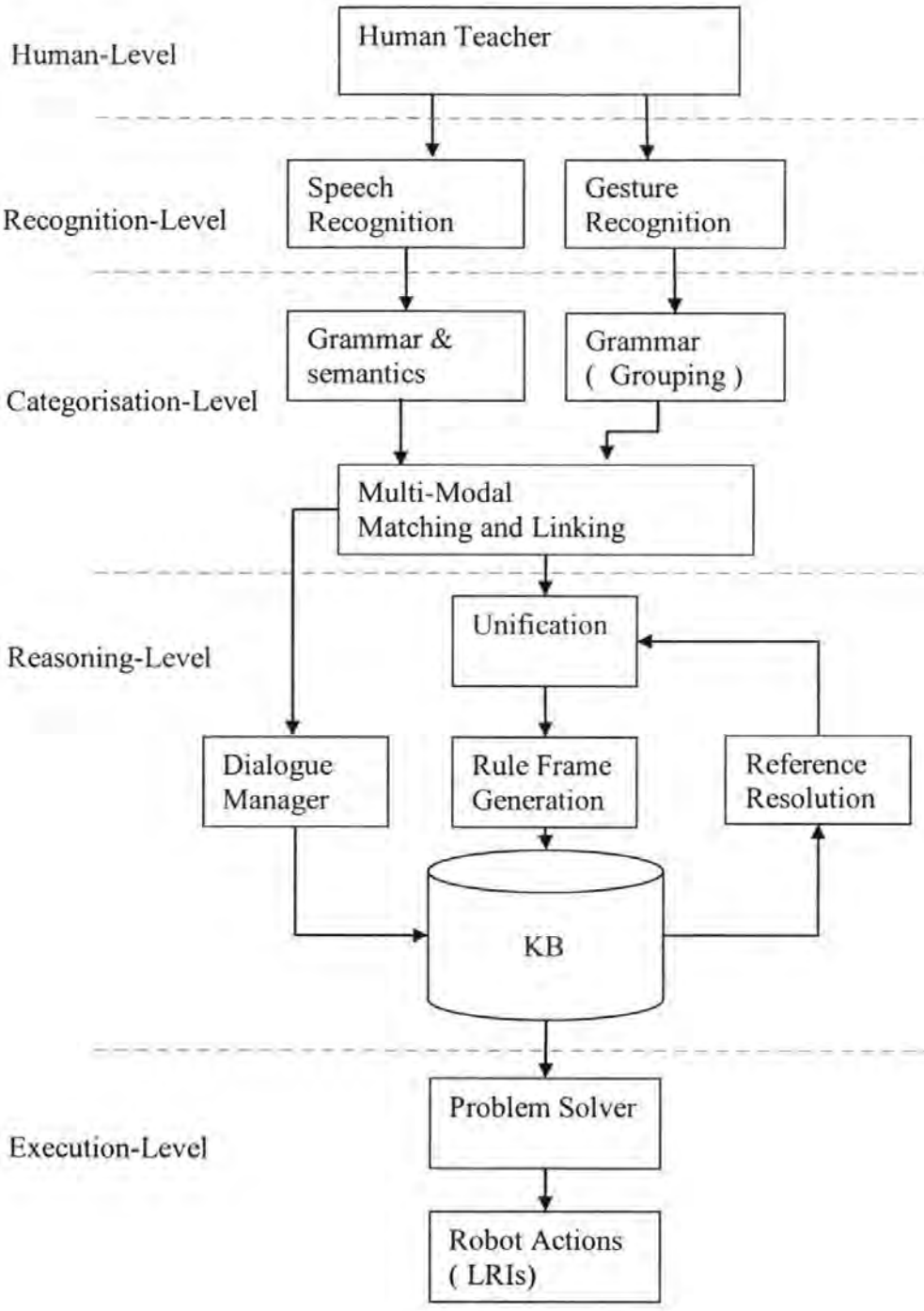


Figure 8-2: Overview of Levels in the MIBL system. These levels have been used to categorise the first fatal error of an instruction that the teacher explains. Information flows from top to bottom.

8.3 Overview of Experiments

The following shows a comprehensive list of experiments that have been carried out:

E1 Corpus collection

transcription of 35322 words and 1136 unique words

3827 utterances, 7411 gestures

(result: corpus available for research)

E2.1 Time-Based Multi-Modal Integration Algorithm test on corpus

(result: 100% assigned, 78% correct pairings , on data set-1)

E2.2 Time & Semantic Multi-Modal Integration Algorithm test on corpus

(result: 89% assigned, 100% correct pairings, on data set-1)

E3.1 System test of dealing phase by playing back from the corpus in text form

(result: learned correct instructions on set-1)

E3.2 System test of pair-rule by playing back from the corpus in text form

(result: learned 16 out 19 times the correct instructions on data set-1 + set-2)

E4.1 Deployment Pilot for Full System test with people

transcription of 745 words and 111 unique words,

190 utterances, 87 gestures

new knowledge to control experiment E4.2

E4.2 Full System test with people for Deployment

transcription of 119 new grammar rules, 3046 words added to the corpus,

300 unique words, 493 utterances, 201 gestures

Success rate per game rule:

pairrule 3 %

cardworth rule 29.5 %

cardexist 14.3 %

dealing 19.4 %

8.4 Testing Completeness in Corpus collection (E1)

The corpus collection experiment was described in detail in chapter 3. The photo figure 8-3 below shows again the setup with two people, one teaching the card game to the other.

Figure 8-3: Experimental Setup during Corpus Collection. Teacher and Student can both interact with a shared view of a virtual playing table. They are separated by a screen so that all gestures and action information goes through touch screen interaction. Servers for recording sound and touch screen are in the background.

One of the challenges in the corpus collection process is to know when to stop collecting. How many subjects need to be interviewed in order to cover the domain or at least be able to bootstrap a conversational robot for further collecting? This depends on the size of the domain. The size of the domain is a limiting factor in corpus-based robotics. Typically an artificially intelligent system can reason in a well defined world with limited concepts, but fails to function when too many new concepts are combined. Therefore the question is: how large is the domain of the MIBL project, i.e. the vocabulary and primitives of the Scopa card game. In order to get an idea of the size, a graph can be created that shows how often new, unknown words occur. See figure 8-4.

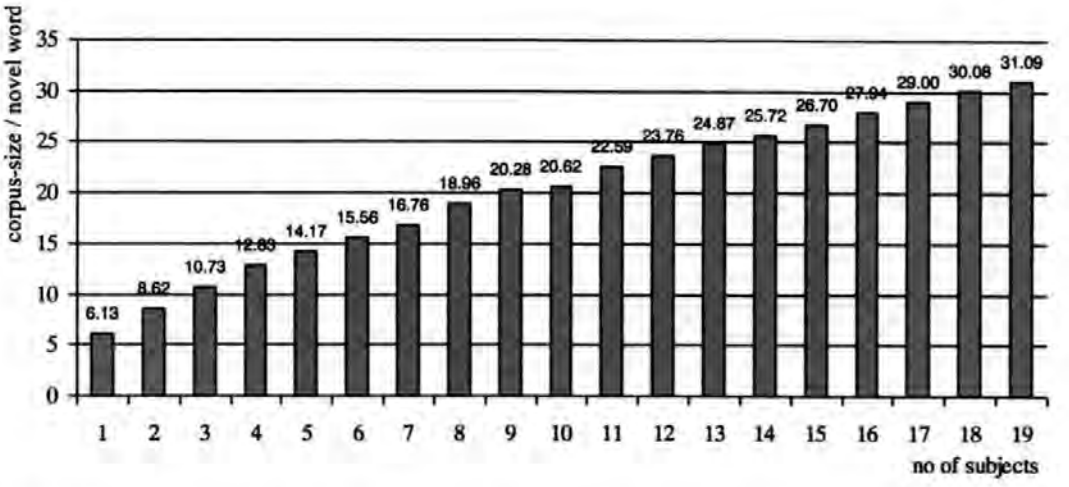


Figure 8-4: Novel words: With the growing size of the corpus, less novel words are discovered. The ratio corpus-size per novel-word is shown in the figure for **human-human corpus, experiment E1**. With an increasing number of subjects (corpus-size) more words go by until another novel word is discovered. A random subset of subjects is selected to create this graph, for example the average of any 1 of the 19 dialogues is taken create the first bar. The first bar shows approx. 6.13 which means that a novel word occurs every 6.13 words on average if the corpus consists only of the transcript of one subject.

After collecting dialogues, the period until a novel word is discovered has risen to the rate of 31 words. The inverse of the period is the frequency, whereby the axis is not time but the number of words "read" and stored in the corpus. The inverse (1/T) of the diagram above would show reduction in frequency of novel words, the more words are in the corpus.

The acceptable rate for stopping to expand (collect) the corpus is difficult to determine, however the graph 8-4 never goes to infinity since it slope almost levels towards the end of the graph. Or inverse the frequency of novel words never goes to 0. This means that there will always be new words discovered with the expansion of the corpus. This is a known phenomena in linguistics. A domain is never closed and there is always some new words discovered with further dialogues. It makes sense to keep collecting new vocabulary until the slope levels. From this point the maximum performance is reached in the domain for a given system. This process could be described as "bootstrapping".

Knowing there will be novel words which potentially cause problems to the speech recognition and understanding should be taken into account in the dialogue management so that the robot can react if a novel word occurred. In the case of MIBL the robot will reject the utterance with sentences like "I could not understand what you are saying". To further decrease the frequency of novel words, the user could be guided by the robot in what the robot understands. Therefore in MIBL, the robot repeats what the user said if the recognition probability is low, i.e. a novel word could potentially be in the utterance.

8.5 System test of dealing rule by playback from corpus in text form (E3.1)

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. This test is about testing the "learning" capability of 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 are played back. The robot has a chance to ask questions during this learning phase. Since the playback of the corpus-recording is fixed and does not contain the answers, the operator pauses the experiment to answer questions of the robot manually. That the robot asks questions in different places that the human student, shows that the reasoning is not equivalent. The robot takes a more conservative approach and asks questions as soon as there is any room for uncertainty, while humans don't.

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 data set-1 and data set-2 to reduce the workload. In the first experiment, called E3.1, 10 dialogues (of our set-1) where played back to the robot, and the robot successfully learned the dealing rule in the way the teacher explained it. Appendix A12 shows a printout of the rule frames. The robot learned from human instructions.

8.6 System test of pair-rule by playback from corpus in text form (E3.2)

Similar to experiment E3.1 the aim of the experiment E3.2 is to confirm that learning of human instructions from the corpus works. In this case not the dealing of cards but the pairing rule is investigated. The pairing of cards is a complex rule consisting of conditionals as well as a sequence of actions to take the matching cards of the table. The success of the experiments would prove at least that the instruction primitives fit together and can be found in the corpus to build up the pairing rule in the robots representation format.

In this second experiment E3.2, a total of 19 dialogues (data set-1 and set-2) from the corpus were investigated. The investigated part of the dialogues contain explanations of how to capture cards by pairing them together. This part was previously tagged (with corpus tagging from chapter 3.4). In 3 cases the explanation of the pairing rule 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. The LRIs (low-level robot instruction) are actions that the robot produces to take action. Due to the different situations that the robot was in and the individual understanding of the rule, the robot produced different LRIs in figure 8-2.

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.

Session	Success / Failure	Comment	resulting LRI
03	S		move(C/QQ,+table,+hand1)
04	S		move(S/QQ,+hand2,+table)
05	S		move(S/QQ,+hand2,+side1)
06	S		move(S/QQ,+hand2,+temp2)
07	S		move(S/QQ,+hand2,+temp2), move(C/03,+hand2,+temp2)
08	S		move(C/QQ,+table,+temp2)
09	F	missing ifloc	
10	F	teacher does not explain pairing rule	
11	S		move(C/QQ,+table,+side2)
12	S		move(S/QQ,+hand2,+temp2)
13	F	teacher does not explain the rule completely	
14	S		move(S/QQ,+hand2,+temp2)
15	S	but teacher does not mention ifcond	move(C/JJ,+table,+hand2)
16	S	but unification assumes wrong source location	move(S/QQ,+hand2,+hand2)
17	S		move(S/QQ,+hand2,+temp2), move(C/QQ,+table,+side2)
18	S		move(D/04,+temp1,+hand1), move(H/04,+temp1,+hand1)
19	S		move(C/07,+table,+side2)
20	S		
21	S		

Table 8-2: Test on corpus of pairing rule

The experiments E3.1 and E3.2 proved that the language primitives can be found in the corpus and when played back they build up the dealing and pairing rules in the robots inner representation.

8.7 Pilot for Full System test with people (E4.1)

What can we learn from tests with people? The purpose of the test of the robot with people has several reasons. First of all it gives insights to how people communicate with the robot. Furthermore it shows how a corpus-based system is deployed. The deployment will show how the system adapts as new grammar rules are added during deployment. In a typical scenario of first deployment the corpus of the system is expanded. In a commercial environment, it is common practice to deploy a natural language system with a small corpus and then expanded it, see Nuance Gram. Dev. (2005). Nuance already provides a set of call logging and tuning tools for this purpose in mind. Further rationale for adding new grammar rules during deployment is to adapt the interaction to actual field of deployment, since the differences between corpus collection and deployment of the product have an impact on the vocabulary and grammar rules. Therefore, after each subject has been invited to communicate with the robot, the dialogue will be transcribed and new grammar rules will be added to the corpus. The reasoning and multi-modal integration system however will remain unchanged.

It was decided to do a pilot to prepare for the full test with people in a partly "controlled experiment". A pilot test will show initial problems with the communications between human and robot and if any improvements on the system and setup are necessary before starting deployment experiment E4.2. A "positive control" is a procedure that is very similar to the actual experimental test, but is known from previous experience to give a positive outcome. For instance, the previous experience is the corpus, were all subjects took 6 instructions to explain the dealing of cards (no complicated rules). If these 6 instructions can be reproduced with new subjects, the experimental setup and environment is right. New primitives were not expected to occur, which makes the semantics controlled and the outcome a "positive control".

For the grammar this is already an actual experimental test, a "not controlled environment", since new subjects will use new words and grammar that the robot does not know. The 6 instruction primitives for completing the dealing are shown in figure 8-5. Often the last two instructions 5 and 6 are regarded as start of the game, because the person picks up the cards in front of the him/her to look at them. Then only the first 4 are produced by the teacher.

```
"you have to deal three cards for each player"
 1.    move=ns-cardname-ns,03,+table+stock,+templ
 2.    loopindicator=each

"place four cards in the middle of the table face up"
 3.    move ns-cardname-ns,04,+table+stock,+table
 4.    turn=ddd-cardname-+table,?,up

"drag the cards into your area and turn them over"
(missing, start of the game )
 5.    move=ddd-cardname-ns,?,?,+hand2 : imitate=now
 6.    turn=them-?-ns,?,?
```

Figure 8-5: 6 instructions primitives of the dealing phase in the format outputted from the grammar. Transcript of sub01try01.xml

8.7.1 Dialogue Management Issues and Solutions

Speech recognition is a hard problem, since speech recognition software is not achieving 100% recognition. This has implications for the dialogue between a speech recognition system and a human user. It is like speaking to person hard of hearing or a person who's English is not very good. In fact, subject 04 from the pilot experiment went closer to the microphone and spoke louder when repeating an utterance, because she believed the robot is hard of hearing. From the IBL project, it was found that people speak differently to a natural-language robot than to another person. The robot frequently requests the user to repeat what he or she said, because of bad speech recognition performance. However, the corpus was not recorded with persons that constantly ask for repeating the sentence. Therefore the dialogue structure with people talking to a robot is different to that of the corpus. Final report to the EPSRC about IBL (Bugmann 2003) suggests that the user can adapt to the robot's dialog and vocabulary, if the robot guides the user. The MIBL dialogue manager, like the IBL dialogue manager, will ask the user "did you say XYZ" if the speech recognition is unsure. This clarification process is better than just saying "can you repeat that please?". Repeating what the speech recognition understood provides feedback of what vocabulary and sentence structures the robot can understand. This feedback guides the user. Unfortunately this can be annoying to the user. Without this guidance, people would

simplify to a telegraphic style if the robot does not understand (Bugmann 2003). Communication would break down, since this style of telegraphic phrases is not in the corpus. Speech recognition should not be confused with natural language interpretation/understanding. The clarification process, which asks "did you say XYZ" comes after speech recognition, but before interpretation, which means that this dialogue has to be passed first before the semantics of the utterance get passed to the dialogue-manager and the knowledge base.

The clarification question by the robot, to some extend, interrupts the flow of the explanation of the teacher, especially in the case where the teacher has to repeat the sentence. This can be a problem simultaneously actions are carried out by the teacher for demonstration. Usually the demonstration is only carried out once, and not repeated during clarification. This has consequences for multi-modal integration, which relies on timing. As a solution, the timing of the user's first attempt to say the utterance is frozen and passed with the semantics to the dialogue-manager and unify-learn-mechanism. This way multi-modal integration timing is preserved. The user's first attempt to make his intentions known to the robot is the crucial one for multi-modal integration.

8.7.2 Procedure of pilot experiment

The instructions to the subjects can be found in appendix A2 and appendix A3.

4 subjects have been invited to take part in the pilot. Every subject had two tries to teach the robot. This small number is justified, since it is a pilot experiment. A set of paper strips with card game rules written on them is placed in front of the subjects, upside down. The subjects were instructed to take a paper strip. They studied the paper and had to give it back so they remembered the rules. In fact in the pilot all paper strips contained the same text, shown in figure 8-6. The conductor of the experiment tried to create an illusion that he genuinely didn't know the rules on these strips.

```
This is a two player game.

Dealing:

deal 3 cards to each player

then deal 4 cards, face up, into the middle

                                                              RS 203
```

Figure 8-6: Paper strip with card game rule for the pilot test

After their attention was distracted from the rules by explaining the touch screen and the experimental setup, the subjects were asked to explain the rule to the experimenter, step by step. This is to clarify in the subjects mind how to teach the rules. During the whole experiment, the person running the experiment was not allowed to say any card game instruction to the subjects, unless they did not understand what was written on the piece of paper with the rules.

After it was assured that the subjects understood the rules they were allowed to explain it to the robot. The robot first says "could you explain the game to me please ?". The subjects went on to communicate with the robot until they have explained the rules and felt the robot understood. Then the experiment was stopped.

8.7.3 Findings, problems and changes from the Pilot Test:

1. The subjects did not believe that the experimenter didn't know the rule in advance and therefore their effort when explaining the rule to me was not instruction by instruction. This was corrected by reminding the subjects.
2. The subjects did not use 6 instructions like in the corpus because it did not occur to them that they have to take the cards into their hand and look at them in order to start playing. This was because they were only instructed to deal the cards and then stop. All produced the 4 dealing instructions from figure 8-5. This will not be a problem in the next experiment (E4.2) because it will include a game phase.
3. Two subjects first thought that the robot will move the card for them, while they are explaining. This is not what was found in the corpus. The corpus had all 19 teachers explaining the dealing phase by demonstrating (teacher doing the actions). The subjects do not feel they are talking to a normal playing partner.

One hypothesis could be the lack of embodiment or they think of the robot as a servant ?

4. One subject explained the dealing without demonstrating in the first try.
5. Subjects did not wait until the robot has finished speaking, before they answered. Particularly when the answer was "Yes" or "No". A more clear busy icon was put in place, see the clock in figure 8-8. A real robot would show that its busy through its facial and body expressions.
6. The Recognition End-Point, which indicates the length of silence that is required after an utterance is regarded as finished has been increased from 0.6 to 1.4 seconds, to deal with hesitation. Users tend to hyper-articulate slightly (speak slowly with long breaks) if they think the robot does not understand.
7. Speech recognition errors and out-of-grammar errors made interaction difficult. To reduce the errors, 52 new grammar rules were added to the corpus with the use of the one-clause-one-primitive principle, see Appendix A13. In particular new grammar rules occurred in the dialogue management. More complex structures in the reply to robot questions were found. If the robot asked "how many cards would you like me to move?" the user can now say "three to the table" or related phrases, rather than just "three".
8. 745 words added to the corpus, 111 unique (novel) vocabulary appeared which the robot didn't know before.

8.8 Full System test with people (E4.2)

8.8.1 Procedure and Instructions

The pilot study and its analysis showed how the experimental setup can be improved. Taking into account the improvement and experiences, the full system test with people is conducted in a similar fashion to the Pilot test, i.e. subjects are invited. They draw from a pile of paper strips with rules printed on them. The exact instructions and procedure is in appendix A2 and appendix A3.

Role of the operator: During the explanation the subjects can get stuck because they can not find a phrase for the rule that the robot understands, then the operator (me) will encourage them to carry on with the next rule. The operator also can encourage to try that rule one more time with the robot. The operator can under no circumstances describe or use words of exactly what to say to the robot.

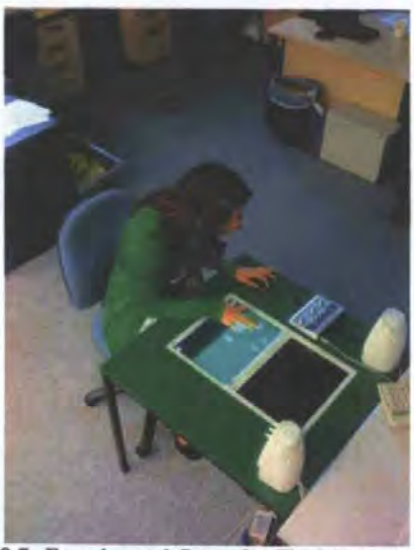


Figure 8-7: Experimental Setup for Final Test with subject sitting in front of the touch-screen, microphone and speakers. The subject is about to lean closer towards the microphone, because she is under the impression the robot does not understand.

Figure 8-8: Table Setup for Final Test. External Sound Card with Interview Microphone to increase signal-to-noise ratio.

This time they are given the following rules to explain:

This is a two player game.

Dealing:

deal 3 cards to each player

then deal 4 cards, face up, into the middle

RS 1203

Playing:

a card on the table with the same value as a card in your hand are a pair.

you can take this pair to the side as a capture

RS 1697

---

the cards have their usual value

and the jack is worth 8 , queen is worth 9 and king is worth 10

ace is low

RS 1283

8,9 and 10 have been taken out from the deck

RS 4834

Figure 8-9: Paper strips with card game rules used for the full test

8.8.2 Results Overall

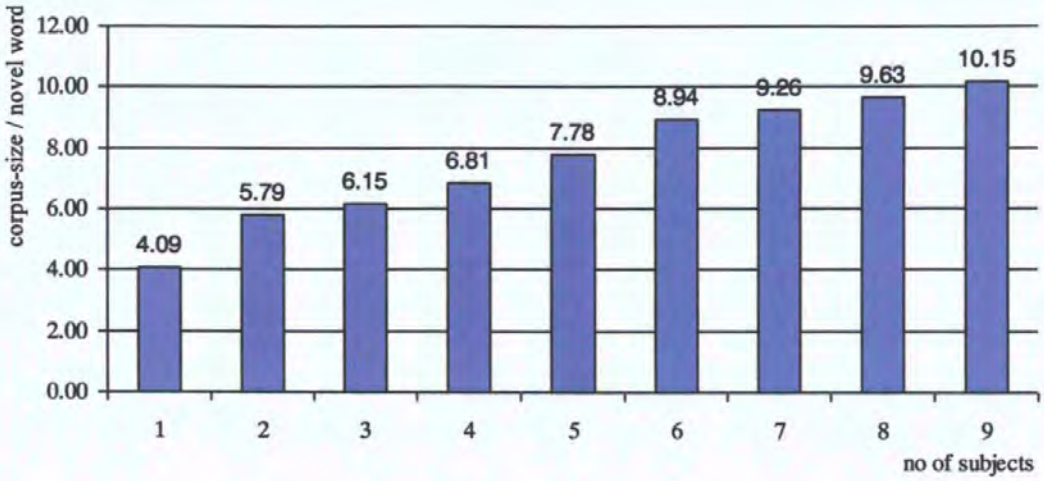


Figure 8-10: Novel words in final test:, experiment E4.2. With the growing size of the corpus, less novel words are discovered. The ratio corpus-size per novel-word is shown in the figure for **human-robot corpus, experiment E4.2**. With an increasing number of subjects (corpus-size) more words go by until another novel word is discovered. A random subset of subjects is selected to create this graph, for example the average of any 1 of the 9 dialogues is taken create the first bar. After adding 8 subjects to the corpus and comparing to the 9th, approximately every 10th word is novel (bar 9, value 10.15). The corpus from E1 was not included in this graph.

The graph 8-10 can be compared to figure 8-4, which also measured after how much words another novel word appeared. The corpus for E4.2 has 3046 words, which is considerable smaller than E1. When testing both together (not shown in graphs) the effect of adding E4.2 to E1 has hardly any impact on the corpus-size / novel-word ratio.

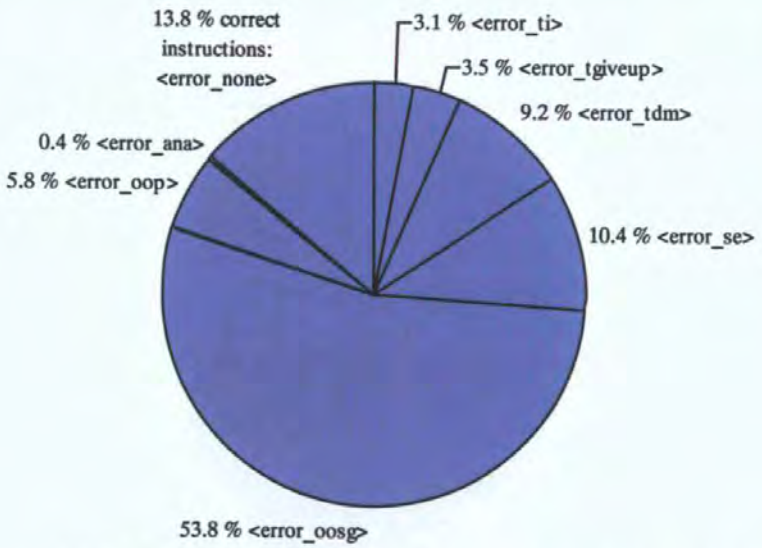


Figure 8-11: Performance of Final Experiment, no error in 13.8 % of attempts. Majority of errors 53.8 % come from category <error_oosg> (error out-of-speech grammar) which means that approximately every second utterance is not defined in the grammar and therefore the robot fails to recognise the attempt of the teacher to teach the instruction. This graphs shows the errors occurred over the whole deployment test of 9 sessions. The actual errors varied from session to session depending on the human subject and the progression in the robots knowledge of grammar rules.

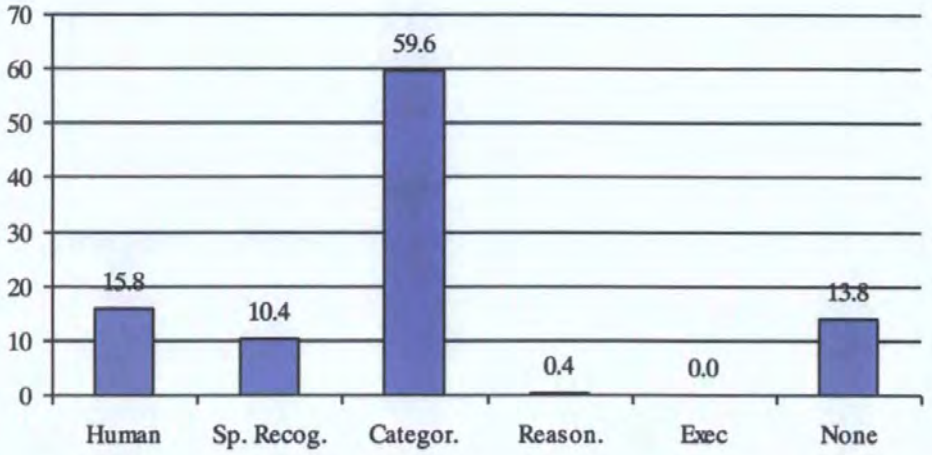


Figure 8-12: Errors split in categories: The errors shown in Figure 8-11, divided into categories from Figure 8-2. Categories from left to right: Human Error, Speech Recognition Error, Categorisation Error, Reasoning and Reference Resolution Error, Execution Error, No Error. For an explanation of the error categories, see table 8-1.

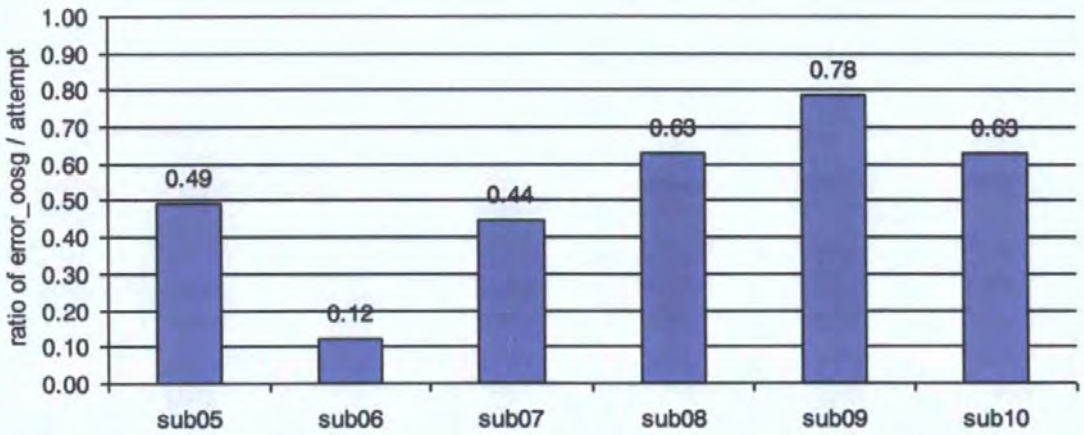


Figure 8-13: Ratio between error_oosg and no of attempts. A smaller number means less errors. After every subject new grammar rules are added that have caused the oosg-errors. There is no decrease in errors as more rules are covered. Through its randomness it seems that this graph measures the alignment of the subject with the robot rather than the influence of increasing corpus size.

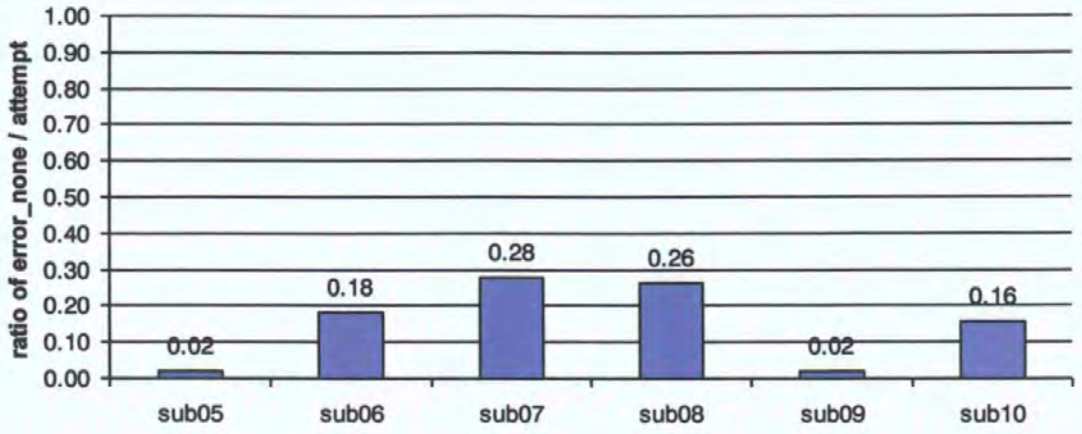


Figure 8-14: Success rate per subject:, ratio between error_none (success) and number of attempts. Sub05 1/55, Sub06 6/33 , Sub07 10/36 , Sub08 12/46 , Sub09 1/51 , Sub10 6/38 .

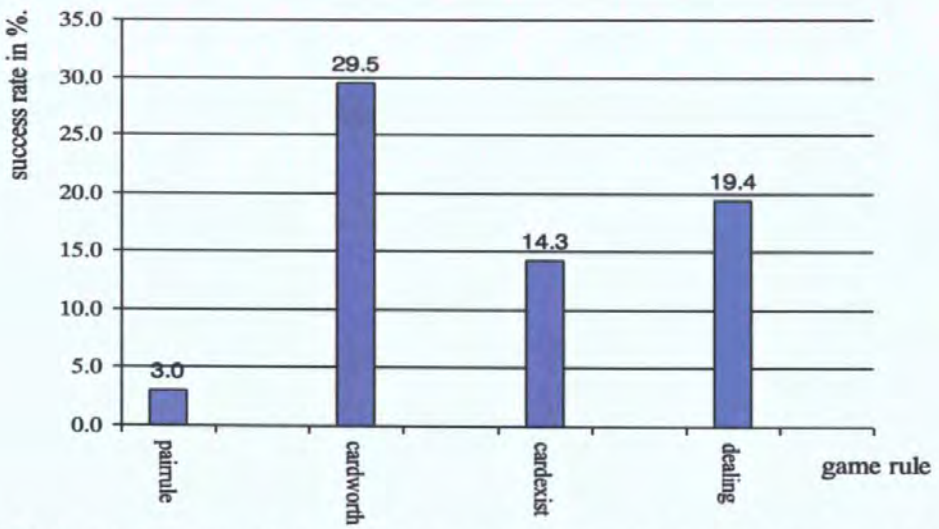


Figure 8-15: Success rate (<error_none>) per game rule. Every attempt (<attempt>) of the teacher to convey the rule is counted. If the attempt was successful, i.e. the robot understands the intention of the primitives without error, it is counted as success (shown in %). If anything goes wrong, i.e. robot error, teacher error, the attempt is counted as a failure. The rate is the total attempts divided by the successful ones. It is clear from this graph that some rules are easier to teach, for example how much points a card is worth (<cardworth>) is understood by the robot after approx. three attempts(29.5%) while the pairing of cards is a more complex rule and has many ways to be expressed, which caused unacceptable success rate of 3%.

Morphology such as singular and plural was not taken into account, and would be counted as separate words, i.e. "dog" "dogs" are counted as separate words.

8.8.3 Findings and Problems from the Final Test and Discussion

1. The experiment added 119 new grammar rules, 3046 words and 300 novel vocabulary to the corpus.
2. People did not use the touch screen as much as in the Human-to-Human corpus E1. Similarly to the pilot experiment there is a lack of flow in the dialogue because of the bad speech recognition performance and out of grammar errors 53.8 %. The repeated rejections of their utterances possibly deterred the subjects doing the actions while speaking compared to the human-to-human corpus. The reduction of Multi-modal interaction to the single modality speech was not an expected finding. Hypothetically, the modality could also have been gesture instead. However only subject 06 (sub06try1.xml) went to do more actions and shorted his sentences to "can you make a pair?". Everyone else tended to engage with clarification dialogue and reduced gestures.
3. This time the subjects completed the dealing phase, but failing to make the robot pick up its own cards, mainly because of out-of-grammar errors. The fact that the robot did not pick up its cards made the users suspicious of its capabilities another factor is that lack of embodiment. I.e. it does not have a visible arm and hand. The robot without cards in his hands leads to confusion on how to carry on to the pairing rule explanation. Picking up cards is often an implicit instruction. It is coded as such by adding the "imitate=now" primitive to the grammar if the teacher talks about picking up his cards. Teachers don't say "I pick up my cards, please pick up yours too" They expect imitation when they pick up theirs, or the other way round instruct the robot to pick up his and pick up theirs without telling.

Traditionally a performance test of the robot would have been carried out without adding grammar rules. What would hypothetically been the outcome of such a test that would not add grammar rules after each session ? Certainly the out-of-grammar errors for instance would have been at least as high as now. The logical conclusion and recommendation at the end would have been to add more grammar rules to cover the domain and therefore reduce out-of-grammar errors. This deployment test however showed that this would not have worked. Graph 8-13 does not go down with the progression over more subjects. The deployment test is already a step ahead of the usual

static analysis and shows an interesting finding, namely the out-of-grammar errors can not be reduced linearly by just adding more grammar rules.

8.8.3.1 Out-of-Grammar Errors

Out-of-Speech-Grammar errors:

As seen in graph 8-13, the rate of Out-of-Speech-Grammar errors < error_oosg> is not decreasing over the progression of more subjects. Also the success rate in graph 8-14 is not increasing. It can be hypothesised that the adding of new grammar rules does not affect the success rate at this size of the corpus. It would affect it, however, if the corpus is very small of course, since without a minimum of grammar rules there is no success at all.

This is a clear indicator that these instructions have *a large variety of being expressed in language*. Here lies a clear limitation of the corpus-based approach or indeed any natural language interface. The rate of errors must decrease to a user-bearable rate before a system can be said usable in practice. This important finding first of all limits the application of the corpus-based approach to instruction-domains that have a limited expressions/size. It will probably limit any other approach (not only corpus-based) that requires grammar-to-robot function mapping. The limit is the cost implication of mapping what hundreds of users said, rather than the concept.

Three recommendations can be made from these results specifically regarding domain size:

1. Alignment of the speaker and the robots grammar by replying with utterances that guide the human to use the right sentence structure and vocabulary.
2. A pilot study needs to be carried out on a domain. The pilot study measures the OOSG / instruction rate. This will show if the corpus-based approach is feasible for the domain. If not, a more restrictive dialogue may be necessary which is more stressful to the user on the other hand. Some domains are so large that they have to be split into sub-domains. A robot working in a department store is a typical example, each department would be a sub-domain to keep implementation feasible.

3. Further research is required into natural language understanding to explore more cost-effective ways of mapping complex language grammar to semantics and robot primitives.

It seems that corpus-based linguistics has suffered from the same problem see (Leidner 2003): "Such training corpora are typically expensive or\virtually non-existent (data resource bottleneck)...leads to unacceptable accuracy"

The IBL domain for example seems to be much smaller than the MIBL domain. Further research is needed to investigate indicators of a domain size. The out-of-grammar errors of IBL are difficult to compare since it suffered from a problem with grammar design.
Lets expand on recommendation one from above. The final report to the EPSRC about IBL (Bugmann 2003) suggests that the user can adapt to the robot's dialog and vocabulary, if the robot guides the user. (Garrod and Pickering, 2004) talk of an alignment process between the two conversation partners. In this alignment linguistic representations are aligned so that both come to an understanding. The simplest version of this alignment is implemented in MIBL as the robot repeats what it understood using the words that are in the grammar with "did you say XYZ". However there is much more potential for doing clever alignment, not only with vocabulary and grammar but also in semantics. What does the teacher understand by "on the table" or the robot may be two different things. By exploiting alignment the oosg-errors can be reduced. The Nuance Gram. Dev. (2005) mentions that 5%-20% of out-of-grammar typically occur, whereby the Nuance guide book assumes a strict interrogative dialogue management. In fact it suggest, one should start with designing the dialogue and then the grammar. This is not the way forward but it shows that the dialogue management influences the out-of-grammar errors. The problems of menu driven restricted dialogue control, as it is used in industry today are compared to the MIBL approach with free speaking in table 8-3. Is a mixture of both ways the answer ? This may be answered in future research.

Control of Dialogue	Human Teacher	vs.	Robotic Student
Properties	Free speaking of the human, the robot only asks questions sometimes. The robot lets the user change context any time.		Robot asks question, human only allowed to answer in a from the robot suggested form, i.e. "yes"," no", "one", "two". Menu driven, like automatic telephony system. Interrogative .
Advantages	- emulates the traditional dialogue between teacher and student, which is natural for teaching scenario		- alignment between robot and human is better, which means there are few speech recognition and out-of-grammar errors.
Disadvantages	- as seen in MIBL the free choice of vocabulary and sentence structure causes a large amount of out-of-grammar errors. It is hard to control them if the robot is not driving the user to alignment.		- the alignment is forced by the robot-student, effectively making the student in charge of the teaching flow and content. This leads to frustration by the human-teacher who can not continue in the way he wants.

Table 8-3: Human vs. Robot Dialogue control. Effect on Alignment

Combinatorial explosion of Grammar rules:

Novel vocabulary is not the only obstacle, the large amount of possible combination of vocabulary, i.e. the grammar is a further limitation. With novel words new possible combinations are introduced which, cause in the worst case, an exponential increase of grammar rules. Key word spotting, or SLM[11] grammars for speech recognition suffer from this problem. With the use of word classes and the one-clause-one-primitive principal the combinations are limited. This limit keeps a lid on the exponential growth problem that for example a keyword spotting grammar would suffer from. Exponential growth is bad for speech recognition. It reduces the probability of finding the correct pathway through the Hidden Markov model (in NUANCE called "confidence"). Often the primitive is recognised correctly, but parameters were erroneous because of the lower probability to get word classes right. If the number of possibilities of expressing an instruction is large, it is necessary to collect a large amount of samples for the one-clause-one-primitive method. Generally it was noted that complex instructions have a larger variety of being expressed. For instance, the pairrule in Figure 8-13 shows a

[11] Statistical Language Modeling

success rate of only 3% because the pairing rule has the largest variety of being expressed.

8.8.3.2 Speech Recognition errors

Speech-recognition errors are marked <error_se> and appear if despite an utterance being defined in the grammar, it is not recognised. 10.4% of errors where due to speech recognition, see 8-11 and 8-12. Nuance claims that the new version Dragon Naturally Speaking 9 has a 20% higher recognition rate than the previous version 8. Assuming this statement is true, the error rate can be reduced further by using the latest speech recognition software. Good audio equipment also has an influence.

8.8.3.3 Human Error

Human error shown in figure 8-12 as 15.8 % consists of 3.1 % <error_ti> (teacher instruction errors), 3.5 % <error_tigiveup> (teacher gave up explaining deliberately) and 9.2 % <error_tdm> (teacher dialogue move error).

Even though all subjects have been tested if they can explain the rules correctly, 3.1 % gave wrong instructions. They made mistakes in their verbal expressions, without noticing. Some of these <error_ti> may come through a lack of concentration or accuracy in the English.

The <error_tgiveup> was due to frustration of the teacher, often after many failed attempts of saying an expression without success.

The <error_tdm> comes from a mismatch in the dialogue, i.e. the robot asks a question, but the teacher does not answer and instead talks about something else. The very simple dialogue manager from MIBL was not able to handle this. Sometimes the error also occurred when the robot asks a question about a previous instruction on the stack, and the teacher already moved on to the next instruction. This caused confusion in both, the teacher and robot. The human teachers were instructed to say "forget it" or similar expressions if they felt the robot was confused.

9 Conclusions and Future Work

9.1 Achievements

In Chapter 1, the aim of the work has been set out as a contribution to knowledge in the field of human-robot communication. More specifically how to convert unconstrained multimodal instructions (spoken natural language + gestures/actions) into a knowledge representation usable for robot reasoning and acting from Chapter 1.1 .

The thesis has shown how to work towards this aim by introducing multi-modal integration algorithms in chapter 5. Particular attention was paid to natural instructions from human-to-human, and the attempt to make algorithms for a robot to capture and integrate these. These natural instructions can not only consist of sequences, like in an assembly task, but also of conditionals which occur in rules. It was shown that a time window for gesture and utterance overlap exists. In combination with semantic matching and nearest neighbour matching a match between the right gestures and utterances was made possible. The results have been quantified in previous chapters 5.3.2 , 8.5, 8.6 .

The work has also shown that contributions have been made in the area of converting actions into usable knowledge for a robot. The finding of primitives (Chapter 3.5, 6.1) and generating a grammar with the use of Corpus-Based Robotics (Figure 1-3) have been shown to be a useful method in order to achieve this aim. The results have been quantified in Chapter 8.

Finally a further aim was to enable a robot to reason and act upon this knowledge gained from human instructors. It was shown in Chapter 7, a combination of frame-based reasoning and an ontology of the robots world enabled reasoning. More specifically, rule-frames and later state transition rules were created from human instructions, which hold the context and actions that the robot has to carry out. These rule-frames provided the robot with a framework that allowed reasoning such as unification, reference resolution, action prediction and planning. Finally all results, shown in chapter 8 and 5.3.2 were critically analysed. These tests showed what happens in a deployment scenario. Interesting findings such as the influence of adding new vocabulary and grammar rules which did not reduce out-of-grammar errors, were demonstrated in the tests. The tests showed the points where corpus-based robotics needs improvement and further research.

9.2 Comment on the corpus-based Robotics approach

The corpus-based approach is a method to create a system that is able to understand instructions that are given at a high human-like level. This leaves the developers with the burden of trying to create very complex robot primitives. First of all it must be said that this is a good way of making researchers address hard real world problems. The corpus comes from end-users and is used for testing the final system, which is therefore rigorous and unforgiving.

However the corpus collection and transcription adds additional work compared with the traditional product development methods, where they are absent. This leads to an overall increase of labour and cost of the product. However, in theory, the product will match user requirements exactly and be able to communicate naturally which may justify the increased price. Furthermore a corpus-based robot will have a wider range of customers, since it could potentially be used by the elderly or by functionally illiterate people.

9.2.1 Human-to-Human vs. Human-to-Robot dialogues

It was observed that people speak differently to robots than to humans. This has caused problems in IBL and MIBL because the corpus has been based on human-to-human dialogues. In the future a wizard-of-oz approach should be taken during corpus collection, whereby the subject must be under the impression of talking to a robot, from the start. It might be necessary to imitate problems with speech recognition during corpus collection. The human to robot dialogues suffered from a too simple dialogue manager that could not guide the user well enough on what the robot can understand. By using alignment techniques, i.e. guiding the user in what expressions the robot understands, it may be possible to reduce out-of-grammar errors in future work.

9.3 Future Work

This paragraph shows possible future work on the MIBL project specifically.

- Comparison and integration of statistical learning algorithms for gesture recognition (from chapter 4.3 Advanced Gesture Recognition). This could improve accuracy in gesture recognition.
- The dialogue model is too rigid and cannot detect if the user decides not to answer a question. Much more can be done to make the dialogue better to keep the context between robot and teacher aligned.
- What is the role of apparent skill level in the dialogue? How to show capabilities of the robot to the user.
- Ask the subjects why they want the robot to do something that they usually would demonstrate themselves if a human would be their student.
- More clever dialogue management guides the user to use the right expressions. This needs to be exploited more and can significantly reduce the <error_oosg>.
- An interesting investigation could be to define a multi-modal grammar and modelling its recognition with HMM.
- Learning locations for gesture recognition from examples (grounding).

9.4 What is holding back user-programmable robots?

An aim set out in the introduction was to advance methods for producing a user-programmable robot for a specific domain. The work presented in this thesis has made progress by showing a method of multi-modal integration and a knowledge-representation scheme in the area of task learning.

Speech recognition is a major limitation and holding back the development of user programmable robots. The experiments have shown that 13.8% + 53.8% of errors is due to speech recognition and grammar problems. The rate is so high that it is not useful in practical applications. For the same reason we still use keyboards on our PCs. As (Lauria, 2007) correctly points out, speech interfaces still do not outperform keyboard based interfaces, for example the voice-dialling option on a mobile phone is hardly used, even it is built into many phones.

References

Abney S. (1991) "Parsing by Chunks", In Robert C. Berwick, Steven P. Abney, and Carol Tenny, Eds, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257-278. Kluwer Academic Publishers, Boston, USA , 1991

Aristotle (transl. 1989) "*Prior Analytics*" translated Robin Smith, Hackett Publishing, 1989, ISBN 0-87220-064-7

Asada M., MacDorman K.F., Ishigurob H. and Kuniyoshi Y. (2001) "Cognitive developmental robotics as a new paradigm for the design of humanoid robots", *Robotics and Autonomous Systems*,Elsevier, Volume 37, Issues 2-3, 30 November 2001, Pages 185-193

Asfour T., Regenstein K., Azad P., Schroeder J., Bierbaum A., Vahrenkamp N. and Dillmann R. (2007) "Design of ARMAR III – a new humanoid " *Proceedings of 16th Int. Workshop on Robotics* in Alpe-Adria-Danube Region - RAAD 2007 Ljubljana, Slowenia, June 7-9, 2007

Baldry A., Thibault P.J. (2006) "*Multimodal Transcription and Text Analysis*", Equinox, 2006

Biber D., Conrad S. and Reppen R. (1998) "*Corpus Linguistics – Investigating Language Structure and Use*", Cambridge University Press, Cambridge, U.K. ISBN 0-521-49957-7

Bird S. and Liberman M. (1998) "Towards a formal framework for linguistic annotations." Presented at the *ICSLP*, Sydney

Bolt R.A. (1980) "Put-that-there": Voice and Gesture at the Graphics Interface, in Proc. of the *7th Annual ACM Conf. on Computer Graphics and Interactive Techniques SIGGRAPH' 80* (Seattle, 14-18 July 1980), Computer Graphics, Vol. 14, No. 3, pp. 262-270.

Bobrow G.D. and Winograd T. (1977) "*An overview of KRL, a Knowledge Representation Language*", Volume 1, Issue 1, Pages 1-123 (January 1977)

Bos J. (2002) "Compilation of Unification Grammars with Compositional Semantics to Speech Recognition Packages." COLING 2002, *Proceedings of the 19th International Conference on Computational Linguistics*, Pages 106-112.

Brand R.J. and Tapscott S. (2007) "Acoustic Packaging of Action Sequences by Infants", *Infancy*, 11:3, 2007, pp. 321-332

Bratko I. (2000) "*Prolog: Programming for Artificial Intelligence*", Addison Wesley, 3rd edition September, 2000

Brill E. (1992) "A simple rule-based part-of-speech tagger", *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, Trento, Italy,152-155,1992

Brooks A.G. (2007) "*Coordinating Human-Robot Communication*" PhD Thesis,MIT, 2007

Brooks R.A. (1990) "Elephants Don't Play Chess", *Robotics and Autonomous Systems*, Vol. 6, No. 1&2., June 1990, pp. 3-15.

Brooks R.A. (1987) "*Planning is just a way of avoiding figuring out what to do next*", Technical report, MIT Artificial Intelligence Laboratory, USA, 1987

Brooks R.A. (1986) "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, March 1986, pp. 14–23; also MIT AI Memo 864, September 1985. (http://people.csail.mit.edu/brooks/papers/AIM-864.pdf)

Bugmann G. (2005) "The What and When of Service Robotics", *Industrial Robot: An International Journal*, Emerald , Volume 32 Issue 6 2005, p.437

Bugmann G., Klein E., Lauria S., Bos J. and Kyriacou T. (2004) "Corpus-Based Robotics: A Route Instruction Example" in *Proceedings of IAS-8*, 10-13 March 2004, Amsterdam, pp. 96-103.

Bugmann G. (2003) "Final report to the EPSRC (IGR GR/M90023)"

Buss M., Beetz M. and Wollherr D. (2007) "CoTeSys - Cognition for Technical Systems", In *Proceedings of the 4th COE Workshop on Human Adaptive Mechatronics (HAM)*, 2007.

Cadoz, C. (1994) "Les réalités virtuelles", *Dominos*, Flammarion, 1994.

Calinon S. and Billard A. (2007) "Learning of Gestures by Imitation in a Humanoid Robot" Dautenhahn and Nehaniv, editors of book: *Imitation and Social Learning in Robots,Humans and Animals: Behavioural, Social and Communicative Dimensions.* Cambridge University Press, 2007.

Cangelosi A. (2007) "*Integration and Transfer of Action and Language Knowledge in Robots: I-TALK*", Large-scale integrating project (IP) proposal, ICT Call 1, FP7-ICT-2007-1

Coutaz J. and Crowley J.L. (1995) "Interpreting Human Gesture with Computer Vision", Position paper for the workshop *Gesture at the User Interface, CHI'95*, Denver, U.S.A. 1995 (http://iihm.imag.fr/publs/1995/)

Copleston S.N. and Bugmann G. (2008) "*Personal Robot User Expectations*", MRes Thesis, University of Plymouth, U.K.

Chai J.Y., Hong P., Zhou M.X. (2003) "Combining semantic and temporal constraints for multimodal integration in conversation systems" *Proceedings of the Human*

Language Technology-NAACL 2003 workshop on Research directions in dialogue processing, Vol. 7, Edmonton, Canada, 2003

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.

Crystal D. (1997) "*A Dictionary of Linguistics and Phonetics*", 4th ed. Oxford: Blackwell Publishers, 1997

Cunningham H., Humphreys K., Wilks Y. and Gaizauskas R. (1997) "Software infrastructure for natural language processing." In *Proceedings of the Fifth Conference on Applied Natural Language Processing Washington*, DC, March 31 – April 3, 1997, pp. 237-244.

Cunningham H. (2000) "Software Architecture for Language Engineering", PhD Thesis, Department of Computer Science, University of Sheffield, June 2000

Daelemans W. and van den Bosch A. (1998) "Rapid development of NLP modules with memory-based learning.", In *Proceedings of ELSNET in Wonderland*, Utrecht: ELSNET, pp.105-113.

Dahlbäck N., Jönsson A. and Ahrenberg L. (1993) "Wizard of Oz Studies – Why and How", *Knowledge-Based Systems*, Vol. 6, No. 4, pp. 258-266.

Davis J. and Shah M. (1994) "Visual gesture recognition", *IEE Proc.-Vis. Image Signal Processing*, Vol. 141, No. 2, April 1994

De Ruiter J.P. ,Rossignol S., Vuurpijl L., Cunningham D.W. and Levelt W.J.M. (2003) "SLOT:A research platform for investigating multimodal communication." In *Proc. of Behavior Research Methods*, Instruments & Computers 2003, 35(3),408-419

Dean D.H. (2008) "What's wrong with IVR self-service", *Journal of Managing Service Quality*, Emerald, Vol 18, Issue 6, 2008, pp 594-609

Dearden A.M., Demiris Y. (2005) "Learning Forward Models for Robots" in *Proceedings of IJCAI-2005*, Edinburgh, pp. 1440-1445, July 2005.

Demiris Y. and Johnson M. (2003) "Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning", *Connection Science*, Vol. 15, No. 4, December 2003, 231–243

Demiris Y. and Khadhouri B. (2005), "Hierarchical, Attentive Multiple Models for Execution and Recognition (HAMMER)", *Proceedings of the IEEE ICRA-2005 Workshop on Robot Programming by Demonstration*, Barcelona, Spain, 2005.

DfEE (1999) "A Fresh Start - improving literacy and numeracy" , *U.K. Government Department for Education and Employment Report*, DfEE 1999, ref: CMBS 1, (known as the Moser Report)

Dillmann R., Ehrenmann M., Steinhaus P., Rogalla O., Zöllner R. (2002) "*Human Friendly Programming of Humanoid Robots - The German Collaborative Research Center*", Tsukuba Research Center, AIST, Tsukuba, Ibaraki, JAPAN, Dec.11-12, 2002

Djenidi H., Benarif S., Ramdane-Cherif A., Tadj C., Levy N. (2004) "Generic Multimedia Multimodal Agents Paradigms and Their Dynamic Reconfiguration at the Architectural Level" *in EURASIP Journal on Applied Signal Processing* 2004:11, Hindawi Publishing Corporation, pp. 1688–1707, 2004

Engelhardt K.G. and Edwards R.A. (1992) "Human-robot integration for service robotics", Chapter 16 from *Human-Robot Interaction*, Taylor & Francis Ltd. , London, 1992

Fikes R.E. and Kehler T. (1985). "The role of frame-based representation in knowledge representation and reasoning." *Communications of the ACM* 28(9) pp 904-920, 1985

Flach P.A., Antonis C.K., Lorenzo M. and Oliver R. (2006) "*Workshop on Abduction and Induction in AI and Scientific Modelling*", Proceedings, Riva del Garda, Italy, August 2006

Fong T.W., Nourbakhsh I., and Dautenhahn K., (2002), "A survey of socially interactive robots: Concepts, design, and applications", Tech. Rep. CMU-RI-TR-02-29 Robotics Institute, Carnegie Mellon University, 2002.

Garrod S. and Pickering M.J. (2004) "Why is conversation so easy?", *Trends in Cognitive Sciences*, January 2004, vol. 8, iss. 1, pp. 8-11(4)

Goldstein, I.P. and Roberts R.B. (1977) "NUDGE: a knowledge-based scheduling program" *Proc. Fifth Int. Conf. Artificial Intelligence*, 257-63

Graça J., Mamede N.J. and Pereira J.D. (2006) "A Framework for Integrating Natural Language Tools" chapter in "*Computational Processing of the Portuguese Language*",Springers Lecture Notes in Computer Science series , Vol 3960/2006 ,Springer,Berlin,ISBN 978-3-540-34045-4,pp 110-119, 2006

Green A., Huettenrauch H., Topp E.A., Severinson-Eklundh K. (2006) "Developing a Contextualized Multimodal Corpus for Human-Robot Interaction" *In Proceedings of the fifth international conference on language resources and evaluation (LREC2006)*, Genova, May 2006

Haasch A., Hohenner S., Hüwel S., Kleinehagenbrock M., Lang S., Toptsis I., Fink G. A., Fritsch J., Wrede B. and Sagerer G. (2004) "BIRON - The Bielefeld Robot Companion" (In E. Prassler, G. Lawitzky, P. Fiorini, and M. Hägele, editors*), Proc. Int. Workshop on Advances in Service Robotics*, pages 27-32, Stuttgart, Germany, May 2004. Fraunhofer IRB Verlag.

Halliday M.A.K. (1976) "A Brief Sketch of Systemic Grammar", in Kress, G.R. (ed), *System and Function in Language*, London: Oxford University Press, 1976, pp.3-6.

Hornby A.S. (2000) "Oxford Advanced Learner's Dictionary", Wehmeier S. (ed), Oxford University Press, Sixth Edition, Oxford, U.K., 2000

Huettenrauch H., Eklundh S.K., Green A., Topp E.A. (2006) "Investigating Spatial Relationships in Human-Robot Interaction", *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS 2006)*, Oct. 9–15, 2006, Beijing, China

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

Iba S., Paredis C. and Khosla P. (2002) "Interactive Multi-Modal Robot Programming", *International Journal of Robotics Research*, Vol. 24, No. 1, January, 2005, pp. 83-104. also appeared in Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington D.C., May 11-15, 2002

Johansson, H. (2001) "Understanding multimodal interaction by exploiting unification and integration rules" presented at the *13th Nordic Conference on Computational Linguistics (NoDaLiDa'01)*, Uppsala, Sweden, 2001

Johnston M., Cohen P.R., McGee D., Oviatt S.L., Pittman J.A., Smith, I. (1997) "Unification-based Multimodal Integration" in the *Proceedings of the 8th conference on European chapter of the Association for Computational Linguistics*, Madrid, Spain,pp 281-288,1997

Jung H.C., Allen J., Galescu L., Chambers S.M., Taysom W. (2007) "Utilizing Natural Language for One-Shot Task Learning", *Journal of Logic and Computation* (Advance Access), December 20, 2007

Kamp, H. and Reyle, U. (1993) "*From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*". Kluwer, Dordrecht.

Kiesler S. and Hinds P. (2004) "Introduction to this special issue on human-robot interaction", *in Human-Computer Interaction* 19 (2004) pp. 1-8.

Koide Y., Kanda T., Sumi Y., Kogure K. and Ishiguro H. (2004) "An Approach to Integrating an Interactive Guide Robot with Ubiquitous Sensors." In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004. (IROS 2004), volume 3, pages 2500–2505, 28 Sept/ 2 Oct.

Kowalski R. and Kuehner D. (1971) "Linear Resolution with Selection Function" *Artificial Intelligence*, Vol. 2, 1971, pp. 227-60.

Kruijff J.-G., Brenner M., Haws N. (2008) "Continual Planning for Cross-Modal Situated Clarification in Human-Robot Interaction", *International Symposium on Robot and Human Interactive Communication, RO-MAN 2008*, Munich, Germany, Aug, 2008, pp. 592-597

Kuniyoshi Y., Inaba M., Inoue H. (1994) "Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance", *IEEE Trans. Robotics and Automation* Vol 10, No 6, Dec 1994

Kyriacou T. (2004) "*Vision-Based Urban Navigation Procedures for verbally instructed robots*", PhD Thesis, University of Plymouth, U.K.

Kyriakopoulos K. and Siciliano B. (2004) "EURON Report", *IEEE Robotics & Automation Magazine*, Vol. 11, No. 4, December 2004, ISSN 1070-9932, pg 128

Lauria S. (2007) "Human Robot Interactions: Towards the Implementation of Adaptive Strategies for Robust Communication", F. Mele et al. (Eds): *BVAI 2007*, LNCS 4729, pp. 555-565, Springer Verlag, Heidelberg 2007

Lauria S., Kyriacou T., Bugmann G., Bos J. and Klein E. (2002) "Converting Natural Language Route Instructions into Robot Executable Procedures" *in proc. of the 2002 IEEE International Workshop on Robot and Human Interactive Communication (Roman'02)*, Berlin, Germany, pp. 223-228.

Leidner J.L. (2003) "Current Issues in Software Engineering for Natural Language Processing" *Proceedings of the Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)* held at the Joint Conference for Human Language Technology and the Annual Meeting of the Noth American Chapter of the Association for Computational Linguistics 2003 (HLT/NAACL'03), Edmonton, Alberta, Canada, pp. 45-50.

Lemon O., Bracy A., Gruenstein A. and Peters S. (2001) "The WITAS Multi-Modal Dialogue System I", *in proc. EuroSpeech 2001*, Aalborg, Denmark, Sept. 2001

Loesch M, Schmidt-Rohr S.R., Dillmann R. (2008) "Making Feature Selection for Human Motion Recognition More Interactive Through the Use of Taxonomies", *in proc International Symposium on Robot and Human Interactive Communication, RO-MAN 2008*, Munich, Germany

Lopes L.S., Teixeira, A.J.S., Rodrigues M., Gomes D., Girao, J., Teixiera C., Senica N., Ferreira L. and Soares P. (2003) "A robot with natural interaction capabilities" Emerging Technologies and Factory Automation, 2003. *in proc. ETFA '03. IEEE Conference*, Volume 1, 16-19 Sept. 2003 Page(s):605 - 612 vol.1

Lungarella M., Metta G., Pfeifer R., Sandini G. (2003) "Developmental Robotics: A Survey." *Connection Science.* 15(4), pp. 151-190. 2003.

Maas J.F. and Wrede B. (2006) "BITT: A Corpus for Topic Tracking Evaluation on Multimodal Human-Robot-Interaction." *in proc. of the Fifth international conference on Language Resources and Evaluation LREC2006.*

Mann G. (1996) "*Control of a Navigating Rational Agent by Natural Language*", PhD thesis, Department of Artificial Intelligence, School of Computer Science & Engineering, The University of New South Wales, Sydney, Australia

Martin J.C. (2005) "Analysis and synthesis of cooperation between modalities.", *HCI International 2005*. 11th International Conference on Human-Computer Interaction. Las Vegas, Nevada USA

Matsui T., Asoh H., Fry J., Motomura Y., Asano F., Kurita T., Hara I. and Otsu N. (1999) "Integrated Natural Spoken Dialogue System" of Jijo-2 Mobile Robot for Office Services, *proc. AAAI/IAAI*, pp. 621-627.

Marciniak T. and Strube M. (2005) "Beyond the pipeline: Discrete optimization in NLP.", *in proc. of the 9th CoNLL*, Ann Arbor, MI, pages 136--143. 2005

Maybeck P.S. (1979) "*Stochastic models, estimation, and control*", Academic Press, New York, USA, Vol 1, 1979

McCarthy J. and Hayes P.J. (1969) "Some philosophical problems from the standpoint of artificial intelligence", *Machine Intelligence*, Meltzer and Michie (Eds.) , Edinburgh University Press, Vol 4, pp 463-502. , 1969

McKerrow P.J. (1991) "*Introduction to Robotics*", Addison-Wesley Publishing Co., Electronic Systems Engineering Series, Wokingham,Australia, ISBN 0 201 18240 8,

Mellish C.S. (1985) "*Computer interpretation of natural Language descriptions*", Ellis Horwood Limited, Chichester, ISBN 0-85312-828-6

Miller G.A. (1985) "WordNet: a dictionary browser.", *Proc. of the 1st International Conference on Information in Data*, University of Waterloo, Waterloo, 1985.

Minsky M. (1975) "A framework for representing Knowledge", In P. Winston (Ed.), *The psychology of computer vision*, McGraw-Hill,New York,U.S.A., 1975 (originally a MIT-AI Laboratory Memo 306, June, 1974.)

Miura J., Iwase K. and Shirai. Y. (2005) "Interactive Teaching of a Mobile Robot," *Proc. 2005 IEEE Int. Conf. on Robotics and Automation*, pp. 3389-3394, Barcelona, Spain, April 2005

Monaco P.C. (2002) "*Creating and Editing Grammars for Speech Recognition Graphically*", United States Patent No 6434523 B1, Nuance Communications, Menol Park U.S.A 13/08/2002.

Newell A. and Simon H.A. (1956) "*The logic Theory Machine, a complex information processing system*", The RAND Corporation, Santa Monica, CA, USA, June 15, 1956, P-868

Newell A. and Simon H.A. (1972) "*Human Problem Solving*", Englewood Cliffs, Prentice Hall, New Jersey, 1972

Nigay L. and Coutaz J. (1995) "A Generic Platform for Addressing the Multimodal Challenge" *in the Proceedings of CHI'95*, 1995, p. 98-105

Nuance Gram. Dev. (2005) "*Nuance Speech Recognition System, Version 8.5, Grammar Developer's Guide*", Merriam-Webster ,Menlo Park, California, U.S.A.

Nuance App. Dev. (2005) "*Nuance Speech Recognition System, Version 8.5, Application Developer's Guide*", Merriam-Webster ,Menlo Park, California, U.S.A.

Ogale A.S., Karapurkar A., Aloimonos Y. (2005) "View-invariant modeling and recognition of human actions using grammars", *in proc International Conference on Computer Vision*, Workshop on Dynamical Vision (ICCV-WDM), October 2005

Otero N., Knoop S., Nehaniv C.L., Syrdal D., Dautenhahn K. and Dillmann R. (2006) "Distribution and recognition of gestures in human-robot interaction," in *15th IEEE International Symposium on Robot and Human Interactive Communication*, Hatfield, U.K., 2006

Oviatt S.L. (1999) "Ten myths of multimodal interaction" *Communications of the ACM*, Vol. 42, No. 11, November, 1999, pp. 74-81

Oviatt S., Coulston R., and Lunsford R. (2004) "When Do We Interact Multimodally? Cognitive Load and Multimodal Communication Patterns." *in Proc. of 6th International Conference on Multimodal Interfaces (ICMI 2004)*, Pennsylvania, USA, October 14-15, 2004

Oviatt S., Cohen P., Vergo J., Suhm B., Holzman T., Winograd T., Landay J., Larson J., Ferro D. (2000) "Designing the User Interface for Multi-modal Speech and Pen-based Gesture Application", *Human-Computer Interaction Journal*, Vol. 15, Issue 04/02/2000 , pg. 263 - 322

Oviatt S., DeAngeli A. and Kuhn K. (1997) "Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction." In: Pemberton, Steven (ed.) *Proceedings of the ACM CHI 97* Human Factors in Computing Systems Conference March 22-27, 1997, Atlanta, Georgia. pp. 415-422.

Panchev C. and Wermter S. (2006) "Temporal Sequence Detection with Spiking Neurons: Towards Recognizing Robot Language Instruction." *Connection Science*, Vol 18,1, pp. 1-22.

Parlett D. (2004) "the Oxford A-Z of Card Games", Oxford University Press, Second Ed.

Perzanowski D., Shultz A.C. and Adams W. (1998) "Integrating Natural Language and Gesture in a Robotics Domain" *in Proceedings of the IEEE International Symposium on Intelligent Control*: ISIC/CIRA/ISAS Joint Conference, Gaithersburg, MD: National Institute of Standards and Technology, 247-252, September 1998.

Perzanowski D., Adams W., Shultz A.C. and Marsh E. (2000) "Towards Seamless Integration in a Multi-modal Interface." in *Proceedings of the Workshop on Interactive Robotics and Entertainment*, Carnegie Mellon University: AAAI Press, 3-9, April 2000.

Perzanowski D., Schultz A.C., Adams W., Marsh E. and Bugajska M. (2001) "Building a Multimodal Human-Robot Interface", *IEEE Intelligent Systems*, 16 (1), IEEE Computer Society, 16-21.

Rabiner L.R. (1989) "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE*, 77(2), pp 257-286, 1989

Riesbeck C.K. (1986) "From Conceptual Analyzer to Direct Memory Access Parsing:An Overview", *Advances in Cognitive Science*, Chapter 8 (http://www.cogsci.northwestern.edu/courses/cg207/Readings/Riesbeck_From_CA_to_DMAP.pdf)

Robinson J.A. "A Machine oriented Logic Based on the resolution principle" *Journal of the ACM*, 12:23--41, 1965.

Rohlfing K., Loehr D., Duncan S., Brown A., Franklin A., Kimbara I., Milde J.-T., Parrill F., Rose T., Schmidt T., Sloetjes H., Thies A. and Wellinghof S. (2006) "Comparison of multimodal annotation tools", *Gesprächforschung* - Online-Zeitschrift zur Verbalen Interaktion, Vol.7, 99-123, ISSN 1617-1837

Rosner M., Johnson R. eds. (1992) "*Computational Linguistics and Formal Semantics*", Series: "Studies in Natural Language Processing", Cambridge University Press, October 1992. ISBN 0521429889

Roy D. (2005) "Semiotic Schemas: A framework for Grounding Language in Action and Perception", Elsevier, *Artificial Intelligence Journal*, Volume 167, Issues 1-2, Pages 170-205 (http://web.media.mit.edu/~dkroy/papers/pdf/aij_current.pdf)

Roy D. and Mukherjee N. (2005) "Towards situated speech understanding: visual context priming of language models", Elsevier: *Computer Speech & Language*, Vol 19, Issue 2 pg 227-248, April

Roy D., Hsiao K., Mavridis N. (2004) "Mental imagery for a conversational robot", *IEEE Transactions of Systems,Man and Cybernetics*, Part B, 34(3):1374-1383, 2004

Rumelhart D.E. (1976) "Understanding and summarizing brief stories" in D. LaBerge and S.J. Samuels (eds.). "*Basic processes in reading: Perception and comprehension*" Lawrence Erlbaum Associates, Hillsdale,NJ,U.S.A.

Russell S. and Norvig P. (2003) "*Artificial Intelligence – A modern Approach*", Pearson Education, 2nd Ed., New Jersey, U.S.A. 2003

Rybski P.E. and Voyles R.M. (1999) "Interactive Task Training of a Mobile Robot through Human Gesture Recognition", *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, Detroit, Michigan, U.S.A, May 1999, pg 664-669

Saunders J., Nehaniv C.L., Dautenhahn K. and Alissandrakis A. (2007) "Self-Imitation and Environmental Scaffolding for Robot Teaching", *International Journal of Advanced Robotics Systems*, Vol. 4, Issue 1, pp. 109-124, ISSN 1729-8806

Schaal S. (1999) "Is imitation learning the route to humanoid robots?" Journal: *Trends in Cognitive Sciences*, Volume 3, Issue 6, 1 June 1999, Pages 233-242

Schank R. (1975) "*Conceptual Information Processing*", North Holland, Amsterdam.

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

Sekine S., Grishman R. (1995) "A Corpus-based Probabilistic Grammar with Only Two Non-terminals", *Fourth International Workshop on Parsing Technology*

Shieber S. (1986) "An introduction to Unification-Based Approaches to Grammar", *CSLI Lecture Notes*, Vol 4, University of Chicago Press, Chicago, Il,USA, 1986

Smith B. (2006) "*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)

Sowa J.F. (2005), website "http://www.jfsowa.com/cg/cgexamp.htm", accessed, March 2005.

Spear A.D. (2006) "*Ontology for the Twenty First Century: An Introduction with Recommendations*", Manual for Basic Formal Ontology from the Institute for Formal Ontology and Medical Information Science, Saarland University, 2006

Spiliotopoulos D., Androutsopoulos I. and Spyropoulos C.D. (2001) "Human-Robot Interaction Based on Spoken Natural Language Dialogue". Presented at the *European Workshop on Service and Humanoid Robots (Servicerob 2001)*, Santorini, Greece, 2001.

Steil J.J., Röthling F., Haschke R. and Ritter H. (2004) "Situated robot learning for multi-modal instruction and imitation of grasping" *Robotics and Autonomous Systems*, Special Issue on "Robot Learning by Demonstration", (47), 129-141, 2004

Torrance M.C. (1994) "*Natural Communication with Robots*", MSc Thesis submitted to MIT Dept of Electrical Engineering and Computer Science.

UNECE/IFR (2005a), "*2005 World Robotics Survey – Summary*", United Nations Economic Commission for Europe/International Federation of Robotics (Statistical-Department), UNCE Information Service, Geneva .

UNECE/IFR (2005b), "*World Robotics 2005: Statistics, Market Analysis, Forecasts, Case Studies and Profitability of Robot Investment*", United Nations Economic Commission for Europe and International Federation of Robotics, Geneva and Frankfurt.

Weizenbaum, J. (1966) "ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine," *Communications of the ACM* 9: 36-45.

Weizenbaum, J. (1976) "*Computer power and human reason*" San Francisco, CA: W.H. Freeman

Wang Y. and Acero A. (2001) "Grammar Learning for Spoken Language Understanding", *in proc. of ASRU Workshop*. Madonna di Campigilo, Italy, Dec 2001

Wang Y. and Acero A. (2006) "Rapid Development of Speech Recognition Grammars", *Speech Communication*, Vol. 48, No. 3-4. , 2006

Wermter S., Elshaw M., Weber C., Panchev C., Erwin H. (2003) "Towards Integrating Learning by Demonstration and Learning by Instruction in a Multimodal Robotics" *Proceedings of the IROS-2003 Workshop on Robot Learning by Demonstration*, pp. 72-79, October 2003.

Wertheimer M. (1923) "Untersuchungen zur Lehre von der Gestalt II", published in *Psychologische Forschung*, 4, 301-350, Year 1923

Wilks Y. (1973) "An artificial intelligence approach to machine translation" article in "*Computer Models of Thought and Language*", W.H. Freeman, San Francisco, CA, U.S.A.

Wimmer M, Schuller B., Arsic D., Radig B., and Rigoll G., (2008) "Low-Level Fusion of Audio and Video Features For Multi-Modal Emotion Recognition". In: Alpesh Ranchordas and Helder Araújo, editors, *Proc. 3rd Int. Conf. on Computer Vision Theory and Applications VISAPP*, Funchal, Madeira, Portugal, volume 2, pp. 145–151., 2008.

Winograd T. (1971) "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", published in three forms:
MIT AI Technical Report 235, February 1971 (http://hdl.handle.net/1721.1/7095)
Journal, Cognitive Psychology Vol. 3 No 1, 1972
Understanding Natural Language (Academic Press, 1972).

Witt A. (2002) "*Multiple Informationsstrukturierung mit Auszeichnungssprachen. XML-basierte Methoden und deren Nutzen für die Sprachtechnologie*", PhD Thesis, University of Bielefeld, Germany

Wolf J.C. and Bugmann G. (2005) "Multimodal Corpus Collection for the Design of User-Programmable Robots." *Proc. Taros 2005*, London, pp. 251-255.
MuTra link: (http://www.swrtec.de/swrtec/mibl/mutra/index.php)

Wolf J.C. and Bugmann G. (2006) "Integration of visual and spoken input in robot instructions" *in the Proceedings of the European Robotics Symposium*, Italy, Palermo

Wolf J.C. and Bugmann G. (2006) "Linking Speech and Gesture in Multimodal Instruction Systems" *in the Proceedings of RO-MAN 06: The 15th IEEE International Symposium on Robot and Human Interactive Communication*, Hatfield, U.K., pg. 141-144

Wolf J.C. and Bugmann G. (2007) "Understanding Rules in Human-Robot Instructions" Special Session paper *in the Proceedings of RO-MAN 07: The 16th IEEE International Symposium on Robot and Human Interactive Communication*, Jeju Island, South Korea, Paper# WA2-4, pg. 714-719, Sept 2007.

Wolf J.C. and Bugmann, G. (2008) "Converting Multi-Modal Task Instructions to Rule-Based Robot Instructions" *in the Proceedings of RO-MAN 08: The 17th IEEE International Symposium on Robot and Human Interactive Communication*, Munich, Germany, Aug 2008.

Wolf, J.C. (2008) "*The MIBL Manual*", Technical Report, University of Plymouth

Wrede B., Haasch A., Hofemann N., Hohenner S., Hüwel S., Kleinehagenbrock M., Lang S., Li S., Toptsis I., Fink G.A., Fritsch J. and Sagerer G. (2004) "Research issues for designing robot companions: BIRON as a case study", *In Proc. IEEE Conf. Mechatronics & Robotics*, 2004.

Yanco H.A. and Drury J.L. (2004) "Classifying human-robot interaction: an updated taxonomy", *in IEEE International Conference on Systems, Man & Cybernetics* (2004) pp. 2841-2846.

Table of Appendices

Appendix A1: Word Frequency in Corpus

1880	the	156	point	66	many	43	mm	23	number	15	press	9	quick
1450	you	155	game	65	start	42	shuffle	22	obviously	14	next	9	pretty
934	and	151	oh	63	area	41	move	22	done	14	until	9	winning
891	so	149	ha	63	only	39	sure	22	captured	14	hee	9	around
862	ok	145	turn	62	pair	38	both	22	mine	14	total	9	through
792	a	141	at	61	count	38	putting	22	since	14	every	9	adding
756	i	140	diamonds	61	when	37	an	22	means	14	our	9	best
617	cards	138	are	60	ten	37	pile	21	value	14	new	9	wasnt
592	yeh	135	youve	60	by	37	should	21	eh	14	perfect	9	maybe
569	to	134	up	59	use	37	were	21	gotta	14	some	9	noone
517	that	133	five	59	left	36	would	21	back	14	quite	9	already
515	of	129	capture	58	each	34	yes	21	summing	14	match	9	yet
498	one	129	middle	58	ehm	33	good	21	part	14	sum	9	please
461	er	122	more	58	off	32	show	20	explain	13	rule	9	lot
386	card	120	six	58	know	32	actually	20	mean	13	makes	9	isnt
380	in	117	last	58	won	32	person	20	where	13	anyway	9	instead
378	have	114	sweep	58	screen	31	who	20	youll	13	hm	8	ways
373	three	113	table	57	these	31	oo	19	equal	13	allright	8	equals
365	take	111	with	57	basically	30	used	19	once	13	youd	8	anymore
361	it	109	down	57	any	30	has	19	away	13	far	8	thirteen
351	is	109	out	56	because	30	keep	19	finger	13	look	8	fourteen
337	your	109	which	56	had	30	hmm	19	hands	13	together	8	quickly
328	can	107	seven	56	way	30	bit	19	hang	13	lets	8	numbers
322	if	106	over	55	dump	30	really	18	bring	12	case	8	jacks
321	just	103	what	55	youre	29	pairing	18	ah	12	let	8	centre
314	go	103	queen	54	black	29	called	18	weve	12	idea	8	completely
314	got	99	as	54	alright	29	top	18	scores	12	cool	8	havent
300	im	98	again	54	flip	29	try	18	little	12	able	8	showing
287	two	95	or	53	me	29	taking	18	ones	12	trying	8	room
282	now	92	coz	52	deal	28	pack	18	getting	12	queens	8	hit
251	ill	90	first	52	wanna	28	give	18	also	12	onto	8	counts
244	its	89	most	52	will	28	gets	17	been	11	id	8	twos
241	on	88	those	51	not	28	need	17	forty	11	course	8	playing
230	right	87	hand	51	diamond	27	same	17	made	11	seventeen	8	stick
228	there	86	here	51	lay	27	score	17	want	11	kings	8	fours
221	erm	85	jack	50	points	27	um	17	corner	11	final	8	threes
219	put	85	well	49	time	26	about	17	extra	11	second	8	taken
216	thats	85	like	49	thing	26	sweeps	17	whoever	11	dealer	8	why
213	do	84	end	48	green	26	going	16	always	10	yourself	7	possible
211	we	84	but	46	either	26	add	16	gone	10	probably	7	call
209	no	83	think	46	could	26	worth	16	didnt	10	eleven	7	plus
203	then	81	eight	45	drag	26	example	16	clear	10	come	7	different
203	this	80	how	45	make	25	doing	16	better	10	took	7	worry
197	cant	80	king	45	aswell	25	win	16	goes	10	easier	7	twelve
197	four	80	be	45	touch	25	than	16	too	10	before	7	fifteen
190	get	76	dont	45	sort	24	still	16	doesnt	10	shall	7	problem
188	yep	75	side	44	into	23	face	16	round	10	leave	7	definitely
182	all	75	sorry	44	play	23	nothing	16	went	10	minute	7	button
180	them	74	theres	44	they	23	remember	16	us	10	neither	7	added
172	gonna	74	another	44	anything	23	very	15	upside	10	third	7	thought
168	for	71	nine	44	bottom	23	run	15	twenty	10	whats	7	wow
167	ive	71	from	43	other	23	having	15	cannot	10	section	7	works
165	my	70	was	43	say	23	deck	15	matter	9	capturing	7	much
162	see	69	ace	43	did	23	yours	15	rules	9	sixteen	7	whatever

7	front	5	bar	4	work	3	high	3	notice	2	against	2	sec
7	circle	5	zero	4	communal	3	dm	3	areas	2	facing	2	altogether
7	operation	5	does	4	across	3	practically	3	damn	2	lose	2	swoops
7	seen	5	bad	4	arrow	3	multiple	3	pairs	2	told	2	words
7	whole	5	happens	4	set	3	view	3	yey	2	quicker	2	forgot
7	therefore	5	close	4	demonstration	3	couldnt	3	difficult	2	shut	2	tied
7	fine	5	games	4	corners	3	continue	3	related	2	itself	2	screens
7	forget	5	am	4	simply	3	collect	3	small	2	moving	2	blue
7	place	5	looking	4	itll	3	luck	3	disappear	2	heart	2	main
7	dark	5	excellent	4	wont	3	simple	3	tummy	2	general	2	accumulate
6	removed	5	drop	4	line	3	believe	3	bend	2	til	2	priority
6	aim	5	something	4	ready	3	unless	3	vale	2	clubs	2	separately
6	must	5	brilliant	4	chance	3	reshuffle	2	theyve	2	miles	2	convincingly
6	sense	5	wanted	4	even	3	fact	2	clean	2	hearts	2	dear
6	empty	5	word	4	rest	3	grab	2	fives	2	smarter	2	wait
6	d	5	alongside	4	find	3	apart	2	chuck	2	bringing	2	talk
6	takes	5	god	4	earlier	3	jeorg	2	stack	2	counting	2	figure
6	nil	5	difference	4	suits	3	might	2	apply	2	shape	2	addiction
6	push	5	suit	4	arent	3	else	2	single	2	bigger	2	piled
6	sevens	5	neaten	4	draw	3	seeing	2	guessing	2	near	2	allocate
6	understand	5	mind	4	several	3	however	2	pointer	2	coming	2	excuse
6	represents	5	later	4	turns	3	fingers	2	werent	2	teach	2	majority
6	icon	5	whoevers	4	joerg	3	dealing	2	dunno	2	touching	2	access
6	flipped	5	display	4	object	3	help	2	less	2	slow	2	ask
6	dealt	5	nineteen	4	scoring	3	own	2	lost	2	shuffles	2	dragging
6	things	5	swept	4	restarts	3	certain	2	disappears	2	name	2	comparison
6	wins	5	basic	4	menu	3	winnings	2	exactly	2	complicated	2	active
6	low	5	wouldnt	4	track	3	few	2	falling	2	least	2	catch
6	remaining	5	begin	4	picture	3	turning	2	asleep	2	others	2	ass
6	moment	5	etc	4	thatd	3	valuable	2	ooo	2	suppose	1	possibility
6	using	4	uh	4	allocated	3	choice	2	fill	2	known	1	apologies
6	mistake	4	entire	4	swap	3	looks	2	row	2	figured	1	claimed
6	none	4	finish	3	their	3	lucky	2	itd	2	shouldnt	1	usable
6	said	4	eighteen	3	forward	3	bear	2	making	2	adds	1	winningly
6	aces	4	choose	3	opponent	3	happen	2	thinking	2	remind	1	expected
6	correct	4	cs	3	beginning	3	counted	2	puts	2	huh	1	ourselves
6	method	4	reset	3	couple	3	reason	2	identical	2	matters	1	mainly
6	amount	4	eights	3	ahh	3	important	2	bonus	2	confusing	1	ve
6	scored	4	scooper	3	finally	3	proper	2	wrong	2	overall	1	between
6	base	4	he	3	captures	3	sight	2	attempt	2	action	1	cleaning
6	rid	4	flips	3	automatically	3	though	2	running	2	explaining	1	includes
5	player	4	nope	3	random	3	during	2	remove	2	valued	1	his
5	finished	4	wed	3	mouse	3	forgotten	2	wants	2	normal	1	ideal
5	throw	4	working	3	fault	3	started	2	change	2	practise	1	ugh
5	order	4	confused	3	paired	3	tuh	2	higher	2	replace	1	dumped
5	based	4	happened	3	pull	3	tie	2	guaranteed	2	separate	1	screwed
5	soon	4	hold	3	given	3	symbol	2	ff	2	guess	1	early
5	nines	4	c	3	decks	3	placed	2	usual	2	along	1	beat
5	tens	4	values	3	per	3	whereby	2	private	2	thatll	1	load
5	whos	4	turned	3	gain	3	kind	2	flipping	2	sign	1	instance
5	tell	4	pick	3	represent	3	laying	2	myself	2	g	1	redo
5	may	4	played	3	allowed	3	being	2	carry	2	honest	1	basi
5	easy	4	dispose	3	wonderful	3	keeping	2	tricky	2	cleared	1	catching
5	everything	4	piles	3	managed	3	pressing	2	problems	2	patch	1	psychology

1	experiments	1	closer	1	manoeuvre	1	oop	1	reuse	1	below
1	rounds	1	breaks	1	easiest	1	often	1	after	1	agghhhh
1	board	1	rats	1	found	1	numericals	1	originally	1	disasterous
1	voila	1	lower	1	quickest	1	realise	1	ran	1	unlucky
1	puttin	1	placing	1	accidently	1	differently	1	eventually	1	great
1	keeps	1	pure	1	such	1	neat	1	piece	1	happening
1	saves	1	ohohoh	1	buddy	1	upfront	1	split	1	enough
1	matched	1	dealed	1	building	1	responsible	1	sections	1	thirty
1	burning	1	disposing	1	practice	1	news	1	stripe	1	interesting
1	baby	1	evil	1	indicate	1	official	1	longer	1	hoping
1	generous	1	ee	1	ever	1	ahead	1	shaped	1	seem
1	free	1	truly	1	aware	1	handle	1	tap	1	beaten
1	ohhh	1	likely	1	loads	1	stuff	1	corresponds	1	twice
1	gives	1	claim	1	whichever	1	advantage	1	bearing	1	representative
1	love	1	obvioustly	1	heres	1	taught	1	wicked	1	fingertip
1	sums	1	circles	1	professional	1	cover	1	rubbish	1	dividing
1	head	1	inside	1	saying	1	queestions	1	slip	1	sitting
1	hopefuls	1	demonstrate	1	long	1	straight	1	ago	1	further
1	knew	1	reverts	1	themselves	1	faced	1	reveal	1	perfectly
1	needed	1	original	1	awarded	1	types	1	carries	1	suggest
1	flick	1	demo	1	understood	1	special	1	shuffled	1	principle
1	stakes	1	pushed	1	showed	1	stays	1	previous	1	acknowledge
1	accept	1	reshuffles	1	starts	1	future	1	awell	1	methods
1	afterwards	1	range	1	trouble	1	dumping	1	carried	1	judge
1	almost	1	onwards	1	everytime	1	capturings	1	restart	1	relating
1	spacious	1	relate	1	noones	1	luckily	1	sweeped	1	competitor
1	kinda	1	respectively	1	regarding	1	him	1	swooped	1	factor
1	somethings	1	power	1	watch	1	afraid	1	says	1	mat
1	technically	1	removes	1	clever	1	reverse	1	ignore	1	careful
1	gains	1	hides	1	tech	1	felt	1	sixes	1	store
1	scoopers	1	follow	1	devise	1	assume	1	mentioned	1	otherwise
1	depends	1	actual	1	starting	1	pretend	1	fit	1	underneath
1	thousands	1	classed	1	feel	1	ordinary	1	missed	1	showin
1	watching	1	remain	1	hint	1	directly	1	cursor	1	mate
1	tutorial	1	plonk	1	ages	1	wondering	1	mention	1	belong
1	commence	1	totally	1	learning	1	knows	1	relevant	1	divide
1	operated	1	thrown	1	whatll	1	telling	1	agh	1	hey
1	task	1	supposed	1	resets	1	truth	1	TRUE	1	talking
1	written	1	clarify	1	accident	1	yovue	1	curser	1	guy
1	changed	1	subtract	1	walk	1	completed	1	laid	1	hes
1	spectacular	1	learnt	1	slowly	1	lap	1	ordinarily	1	plays
1	personal	1	complete	1	players	1	anybody	1	secret	1	misfortune
1	legit	1	confident	1	brought	1	anyone	1	remembering	1	big
1	chucked	1	shout	1	glad	1	clears	1	past	1	strange
1	oops	1	hidden	1	asked	1	never	1	presume	1	large
1	em	1	gap	1	aiming	1	shuffling	1	runs	1	space
1	helps	1	jumped	1	trial	1	busy	1	selects	1	systems
1	indicates	1	rocking	1	supposing	1	concentrating	1	controls	1	joker
1	technical	1	wierd	1	rare	1	innit	1	select	1	male
1	term	1	fixed	1	cases	1	become	1	meant	1	refresh
1	allow	1	waiting	1	smart	1	seperate	1	n	1	system
1	saw	1	somewhere	1	tiny	1	indeed	1	persons	1	unusual
1	allows	1	chi	1	piling	1	tables	1	errrr	1	opposite
1	continues	1	touchscreen	1	clumsy	1	begins	1	errr	1	greek

1	italian	1	learn
1	distinction	1	nobody
1	required		
1	fucking		
1	align		
1	competition		
1	guards		
1	gave		
1	consists		
1	matches		
1	manage		
1	gaining		
1	tactically		
1	hints		
1	generally		
1	multiples		
1	poor		
1	slighty		
1	slightly		
1	especially		
1	hopefully		
1	favour		
1	changes		
1	fire		
1	queenie		
1	useful		
1	spoiled		
1	plan		
1	while		
1	kept		
1	messing		
1	agghhh		
1	cheat		
1	seems		
1	lots		
1	speak		
1	happy		
1	decisive		
1	entirely		
1	check		
1	selection		
1	ohhhhh		
1	hurt		
1	shadow		
1	mode		
1	shad		
1	f		
1	pairings		
1	hi		
1	thens		
1	public		
1	mmm		
1	oof		
1	pass		

Appendix A2: Instructions to Subjects in Pilot

Human-Robot Interaction Research

Centre for Robotics and Intelligent Systems

University of Plymouth

Briefing:

You are going to explain parts of a card game to a robot.

1. Pick a random set of rules as written text now.
   Please read this rules and make sure you understand them.

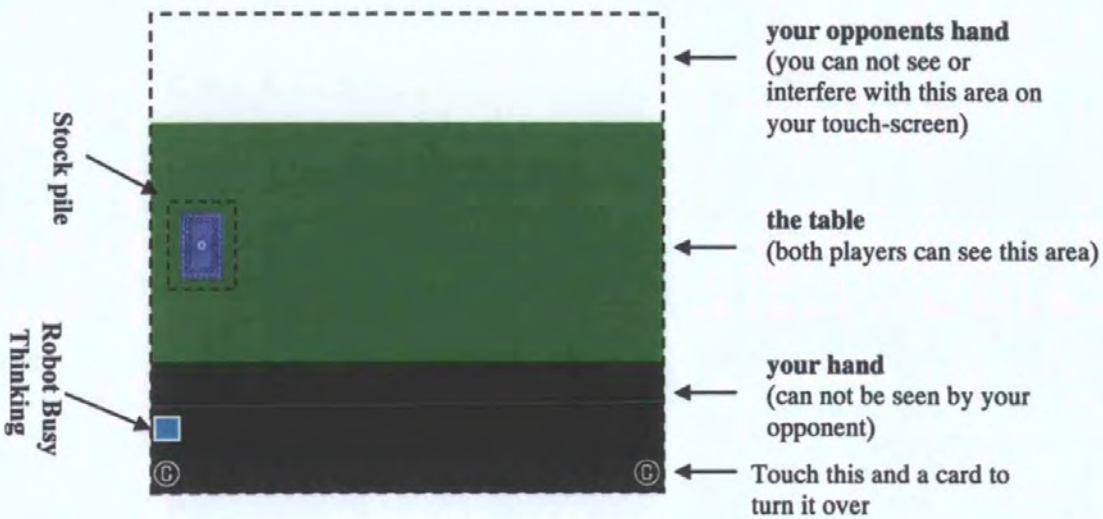
2. You will be given the chance to explain the rule to me on a dry-run.
   Make sure you break down the explanation into small steps.
   You can, if you wish, use the touch screen to support our explanation.

2. We are going to use a touch screen, with playing cards on it.

3. You will then go on to explain the rule to the robot named "Claire".
- "Claire" recognises your actions on the touch screen.
- "Claire" understands speech
- If you made a mistake you can say "*ignore what i just said*"
- Because Claire's hearing is bad it will often ask you "*did you say .....?*".
  Please reply with *yes* or *no* and do not ignore the robots questions.
- While the robot is speaking it doesn't listen, please don't interrupt it

4. After the session you can give some feedback by filling in a one page form.
   You will get an information sheet about this research and we are happy to answer an questions about this research.

Appendix A3: Touch Screen Instructions to Subjects in Pilot



Appendix A4: Examples of Errors

<error_tdm> wrong dialogue move of teacher

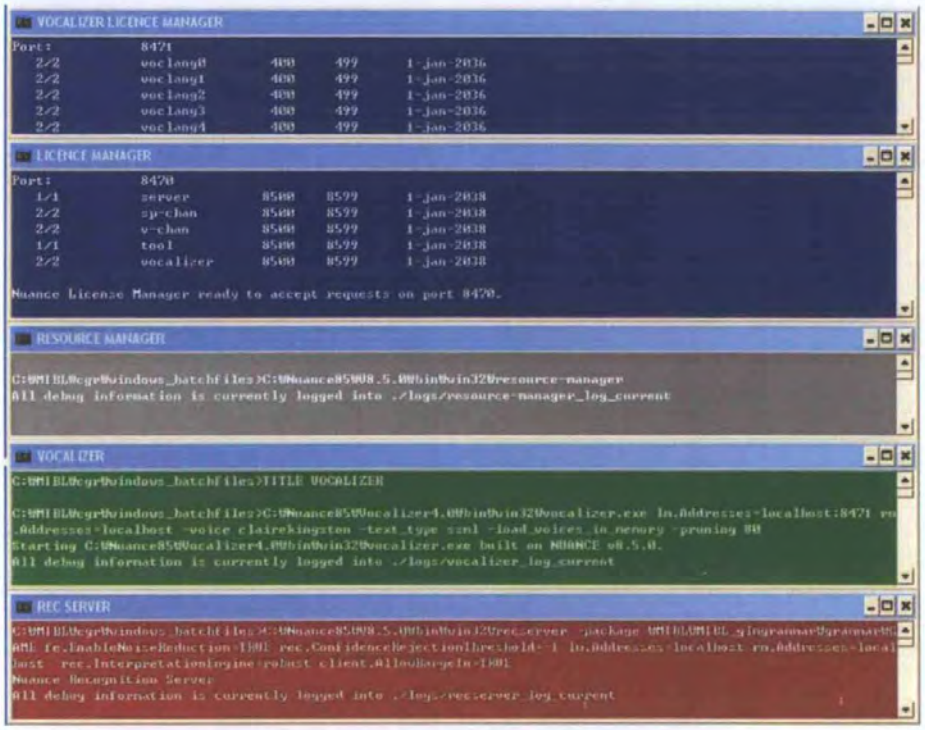
```
<tv t="8659" until="8687">
the jack is worth ten
<recognised>the jack is worth ten</recognised>
<attempt/>
</tv>

<sv t="8805" until="8815">
<speak>did you say the jack is worth ten? </speak>
</sv>

<tv t="8837" until="8927">
no <operator>for once it undestands me and i am telling it the wrong thing</operator>
<recognised>no and take one yours one tens in to me those any zero to me this jacks your place
to me</recognised>
<error_tdm/>
</tv>
```

from sub06try1.xml

Appendix A5: Screenshots of the MIBL Software

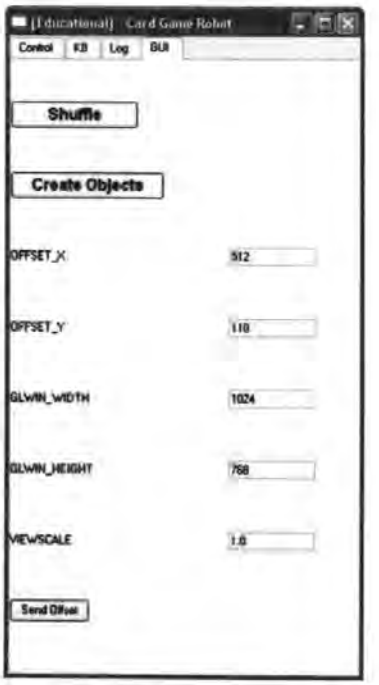


Nuance Speech Recognition background services

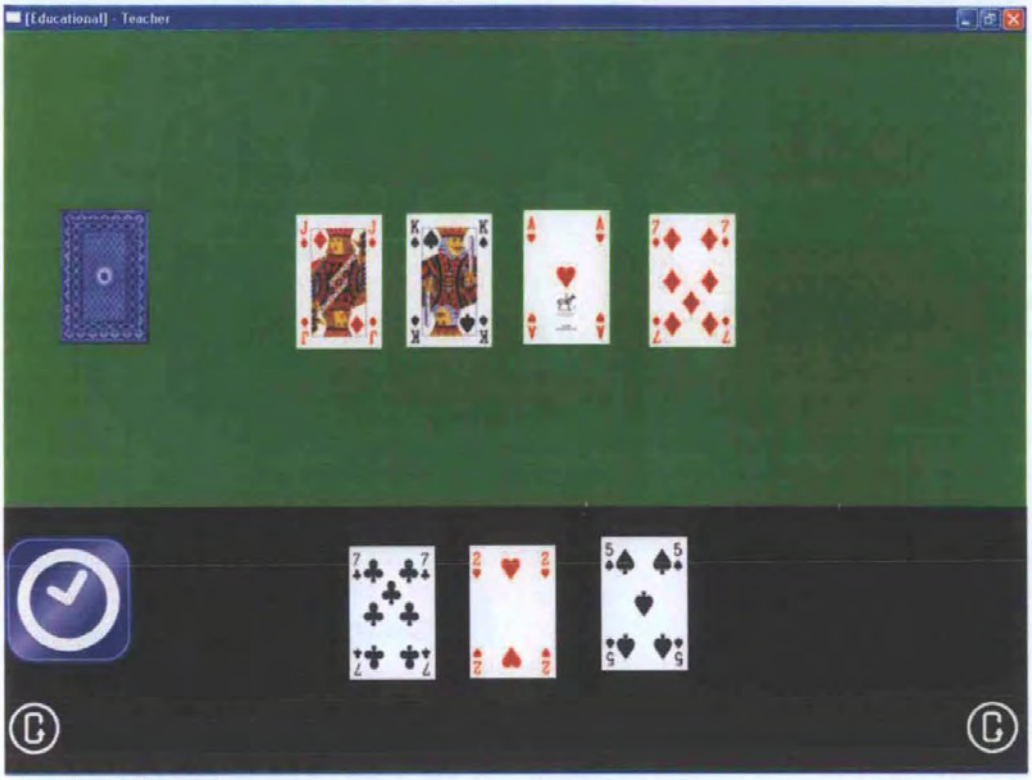
Control Menu of the Card Game Robot Software

Knowledge Base Browser of the Card Game Robot Software

Log Browser of the Card Game Robot Software

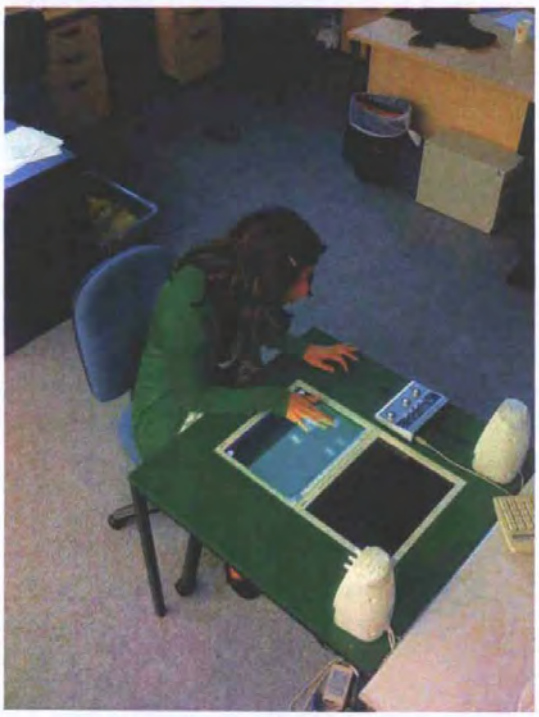


GUI Control of the Card Game Robot Software



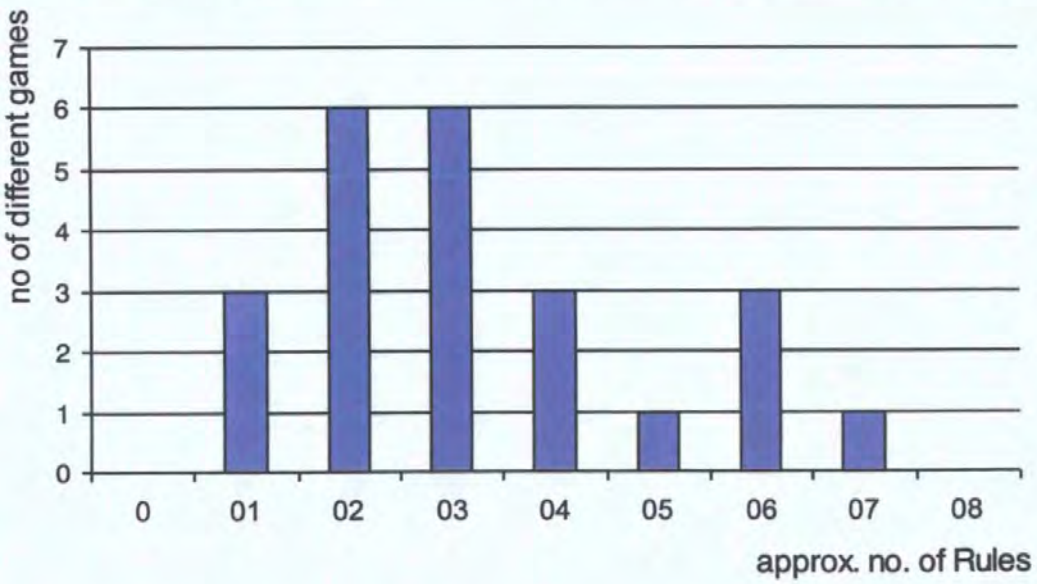
Teachers Screen

Appendix A6: Photos of Final Experiment



Appendix A7: Num. of Primitives in Card Games

Name	Type	Trumphs	point system	Approx. No. of Primitives
German Whist	point trick game	x	x	11
Put	plain trick game			14
Challenge	going-out game	x	?	17
Durak	going-out game	x	?	23
Bohemian Schneider	plain trick game			24
Basra	Fishing game		?	24
Penneech	point trick game	x	x	25
Sedma	point trick game	x		26
Bondtolva	point trick game	x	x	29
Truc	plain trick game			35
Scopa	Fishing game		x	35
Marjolet	point trick game	x	x	37
Ecarte				38
Gin	Rummy game		?	39
Spit	Patience game		?	39
Tute	point trick game	x	x	41
Trappola	point trick game		x	45
Cassino	Fishing game		?	48
Bezique	point trick game	x	x	52
Piquet	plain trick game		x	64
Klaberjass	point trick game	x	x	67
Cribbage	matching game		?	68
Spite / Malice	Patience game		?	72



std. deviation 17.46
37.96

Appendix A8: Grammar

```
;-------------------------------------------------------------
;----------------  Clause Link Level -------------------------
;-------------------------------------------------------------
.CLAUSE_LINK_LEVEL
[

              (CORPUS_LEVEL:sem1 CORPUS_LEVEL:sem2 CORPUS_LEVEL:sem3)  (<sem strcat($sem1 strcat(":" strcat($sem2 strcat(":" $sem3))))>   )
              (CORPUS_LEVEL:sem1 CORPUS_LEVEL:sem2)                    (<sem strcat($sem1 strcat(":" $sem2))>)
              (CORPUS_LEVEL:sem1)                                      (<sem $sem1>)
]

;-------------------------------------------------------------
;-----------------  Corpus Level -----------------------------
;-------------------------------------------------------------

CORPUS_LEVEL
[
       ;Corpus must go here
       ;(EXACT_GAME:s){return($s)}
]

;-------------------------------------------------------------
;----------------- Clause-Grammar Level ----------------------
;-------------------------------------------------------------
;All Constructs listed here must also be listed in the file: GAME.clause-level-taglist
.Tagging
[
       CardWorth_Construct
       CardExist_notExist_Construct
       CaptureRule_Construct
       PairRule_Construct
       Move_Construct
       Turn_Construct
       Explain_Construct
       Context_Change_Construct
       Demo_Construct
       RunOutOfHand_Construct
       RunOutOfEnd_Construct
       CardEachTurn_Construct
       TakeTurn_Construct
       Pilot_Construct
       Final_Construct
```

```
]

;-------------------------------------------------------------
;-----------------  Phrase Level -----------------------------
;-------------------------------------------------------------

ReturnNumberRecursive [
       ReturnNumber:n (return($n))
       (ReturnNumber:n ?and ReturnNumberRecursive:nlist) (return(strcat($n strcat("&" $nlist))  ) )
]

ReturnNumber [
       (one)    (return("01"))
       (two)    (return("02"))
       (three)  (return("03"))
       (four)   (return("04"))
       (five)   (return("05"))
       (six)    (return("06"))
       (seven)  (return("07"))
       (eight)  (return("08"))
       (nine)   (return("09"))
       (ten)    (return("010"))
       (eleven) (return("011"))
       (twelve) (return("012"))
       (thirteen) (return("013"))
]

; Right recursive list of CardCat
CardCatRecursive [
                     (CardCat:c1)   (return($c1))
                     (CardCat:c ?and CardCatRecursive:cclist)     (return(strcat($c strcat("&" $cclist))  ))
]

; Noun Phrase refering to a card or card-category
; ns = not specified location
CardCatNP [
       (ReturnNumber:n CardCat:c)    (return( strcat( strcat( strcat( strcat( strcat("ns-"   $c ) "-") "ns") ",") $n) ))  ; three cards
       (DDD CardCat:c)               (return( strcat( strcat( strcat( strcat( strcat("ddd-"  $c ) "-") "ns") ",") "?") ))  ; the seven, these
sevens
       (DDD)                         (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") "ns") ",") "?") ))  ; those
;      (CardCat:c)                   (return( strcat( strcat( strcat( strcat( strcat("?-"    $c ) "-") "ns") ",") "?") ))

       (card)                        (return( strcat( strcat( strcat( strcat( strcat("a-"    "cardname" ) "-") "ns") ",") "1") ))
       (cards)                       (return( strcat( strcat( strcat( strcat( strcat("ns-"   "cardname" ) "-") "ns") ",") "?") ))

       (NumberCat:n)                 (return( strcat( strcat( strcat( strcat( strcat("?-"    "?") "-") "ns") ",") $n)  ))  ; take three
(missing article suggests number)
       (DPD CardCat:c)               (return( strcat( strcat( strcat( strcat( strcat("dpd-"  $c ) "-") "ns") ",") "?") )); your card
       (them)                        (return( strcat( strcat( strcat( strcat( strcat("them-" "?") "-") "ns") ",") "?") ))
```

```
        (them all)                      (return( strcat( strcat( strcat( strcat( strcat("them-" "?") "-") "ns") ",") "all") ))
        (?the DetAmountOf:n them)       (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") "ns") ",") $n ) ))  ; the three of them
(found in the corpus)
        (it)                            (return( strcat( strcat( strcat( strcat( strcat("it-"   "?") "-") "ns") ",") "01") ))

        (a CardCat:c)                   (return( strcat( strcat( strcat( strcat( strcat("a-"    $c ) "-") "ns") ",") "1") ))  ; a seven
        (an CardCat:c)                  (return( strcat( strcat( strcat( strcat( strcat("a-"    $c ) "-") "ns") ",") "1") ))  ; am eight

        (no CardCat:c)                  (return( strcat( strcat( strcat( strcat( strcat("ns-"   $c ) "-") "ns") ",") "00") ))
        (none of DDD)                   (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") "ns") ",") "00") ))

        (DetAmountOf:n DDD CardCat:c) (return( strcat( strcat( strcat( strcat( strcat("ddd-"  $c ) "-") "ns") ",")  $n ) )) ; all of these cards
        (DetAmountOf:n DPD:p CardCat:c)(return( strcat( strcat( strcat( strcat( strcat("dpd-"  $c ) "-") $p) ",")  $n ) )) ; all of my cards ,
any of my cards
        (DetAmountOf:n DDD NumberCat CardCat:c)  (return( strcat( strcat( strcat( strcat( strcat("ddd-"  $c) "-") "ns") ",") $n) )) ; all of
those two diamonds
        (ReturnNumber:n more CardCat:c)(return( strcat( strcat( strcat( strcat( strcat("ddd-"  $c ) "-") "ns") ",")  $n ) )) ; three more cards
        (DetAmountOf:n others)          (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") "ns") ",") $n ) )) ; two others
        (DDD DetAmount:n CardCat:c)     (return( strcat( strcat( strcat( strcat( strcat("ddd-"  $c ) "-") "ns") ",") $n ) )); those two diamonds
        (DDD DetAmount:n)~.9            (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") "ns") ",") $n ) ))  ; those three (cards)

; for a test, noun phrases with possesives
        (the ones Location:1)           (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") $1) ",") "?") ))
        (DetAmountOf:n the ones Location:1)   (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") $1) ",")  $n) ))
        (DetAmountOf:n DDD CardCat:c Location:1)     (return( strcat( strcat( strcat( strcat( strcat("ddd-"  $c ) "-") $1) ",")  $n) )) ; any of
the cards on the table
        (DDD CardCat:c Location:1)      (return( strcat( strcat( strcat( strcat( strcat("ddd-"  $c ) "-") $1) ",") "?") ))
        (DetAmountOf:n DPD:1 DetAmount CardCat:c)   (return( strcat( strcat( strcat( strcat( strcat("ddd-"  $c ) "-") $1) ",")  $n) ))  ;one of
your three cards
        (DPD:1)                         (return( strcat( strcat( strcat( strcat( strcat("dpd-"  "?") "-") $1) ",") "?") )) ; yours , mine
]

PassiveCardCatNP [
        (they are all)                  (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") "ns") ",") "all" ) ))
        (these are the DetAmountOf:n)   (return( strcat( strcat( strcat( strcat( strcat("ddd-"  "?") "-") "ns") ",") $n ) ))
        (it is)                         (return( strcat( strcat( strcat( strcat( strcat("it-"  "?") "-") "ns") ",") "01" ) ))
]

CardCatRef [
        (it)                            (return("it" ))
        (they)                          (return("ddd"))
        (CardCat:c)                     (return( $c  ))
]

DetAmountOf
[
        (DetAmount:n ?of)               (return( $n  ))
]
```

```
DetAmount
[
        (both)                          (return( "2" ))
        (all)                           (return( "all" )) ;Universal
        (any)                           (return( "any" )) ;Existential
        (ReturnNumber:n)                (return( $n  ))
]

DDD [
;       determinative demonstrative deictic references: this,these,that,those,the
        (this)
        (these)
        (that)
        (those)
        (the)
]

DPD [
; determinative possessive deictic references: my,your,our,his,her,its,their,ones
        (my)                            (return("+hand1"))
        (your)                          (return("+hand2"))
        (yours)                         (return("+hand2"))
        (mine)                          (return("+hand1"))
        (our)                           (return("ns"))
        (his)                           (return("ns"))
        (her)                           (return("ns"))
        ; its causes frequent problems with recognition
        ; (its)                         (return("ns"))
        (their)                         (return("ns"))
]

CardCat [
; because suits are not so important in Scopa, and therefpre not so much in the corpus, change the probabilities
        (cards)~.4                              (return("cardname"))
        (NumCatSP:n)~.35                        (return($n))
        (card)~.2                               (return("cardname"))
        (SuitCat:s)~.0166                       (return($s))
        (SuitCat:s of NumberCat:n)~.0166        (return(strcat($s strcat("/" $n)) ))
        (NumberCat:n of SuitCat:s)~.0166        (return(strcat($s strcat("/" $n)) ))
]

SuitCat[
        spades  (return("S"))
        clubs   (return("C"))
        hearts  (return("H"))
        diamonds        (return("D"))
        spade   (return("S"))
        club    (return("C"))
        heart   (return("H"))
```

```
        diamond (return("D"))
]

NumCatSPRecursive[
        (NumCatSP:n)            ( return($n) )
        (NumCatSP:n [?and ?or] NumCatSPRecursive:nlist)(     return(strcat($n strcat("&" $nlist))  ))

]

NumCatSP[                                           ; Numerical name for card category which accepts singluar and plural
        (?number NumberCat:n) (return($n))
        pair    (return("02"))
        twos    (return("02"))
        threes  (return("03"))
        fours   (return("04"))
        fives   (return("05"))
        sixes   (return("06"))
        sevens  (return("07"))
        eights  (return("08"))
        nines   (return("09"))
        tens    (return("10"))
        jacks   (return("JJ"))
        queens  (return("QQ"))
        kings   (return("KK"))
        aces    (return("AA"))
]

NumberCat[
        zero    (return("00"))
        two     (return("02"))
        three   (return("03"))
        four    (return("04"))
        five    (return("05"))
        six             (return("06"))
        seven   (return("07"))
        eight   (return("08"))
        nine    (return("09"))
        ten             (return("10"))
        jack    (return("JJ"))
        queen   (return("QQ"))
        king    (return("KK"))
        ace             (return("AA"))
]

Location [
        ([on to in into onto] LocationN:ln)( return($ln) )
        ( LocationN:ln)( return($ln) )
]
```

```
; from the human teachers point of view: hand1/temp1 = "my"
;                                        hand2/temp2 = "your"

LocationN [
                    (your ?little black bit)      {return("+hand2")}
                    (your black area)             {return("+hand2")}
                    (?the black area)             {return("+hand2")}
                    (your black bar)              {return("+hand2")}
                    (your place)                  {return("+hand2")}
                    (the black area ?(at the bottom of your screen)){return("+hand2")}
                    (your hand)                   {return("+hand2")}
                    (my hand)                     {return("+hand1")}
                    (my place)                    {return("+hand1")}
                    (?the black bar)              {return("+hand")}
                    ( the middle)                 {return("+table")}
                    ( the middle of [your the] screen)    {return("+table")}
                    ( the [centre center] of the screen ) {return("+table")}
                    ( in the middle of the table) {return("+table")}
                    ( the center)                 {return("+table")}
                    ( the centre)                 {return("+table")}
                    (?the table)                  {return("+table")}
                    (the bottom of your screen)   {return("+table")} ; is that right ?
                    ( ?the public area)           {return("+table")}
                    (your area)                   {return("+hand2")}
                    (?the stockpile)              {return("+table+stock")}
                    (?the pile)                   {return("+table+stock")}
                    (?the deck)                   {return("+table+stock")}
                    ( the table)                  {return("+table")}
                    ( the green area)             {return("+table")}
                    ( my area)                    {return("+hand1")}
                    (your side into the dark area){return("+hand2")}
                    (my dark side)                {return("+hand1")}

                    (to you)                      {return("+hand2")}
                    (for me)                      {return("+hand1")}
                    (to me)                       {return("+hand1")}

                    (the side)                    {return("+side1")}
                    (in front of you)             {return("+temp1")}
                    (they are in front of you)    {return("+temp1")}


                    (there)                       {return("+there")}
                    (here)                        {return("+here")}

                    ; to one side
                    ; to the side
                    ; ?here in my little pile
]
```

```
I
[
        ?BE
]

BE
[
        (i)
        (ill)
        (i will)
        (i am going to)
        (you)
        (we)
        ([i you] would ?(like to))
        (youd)
]

MOVE [
                                                ; amount in the corpus
                (take)~0.631                    ; 631
                (put)~0.377                             ; 377
                (deal)~0.122                    ; 122
                (move)~0.047                    ;  47
                (drag)~0.056                    ;  56
                (pull)~0.003                    ;   3
]

FACE
[
        (up)            {return("up")}
        (down)          {return("down")}
]
```

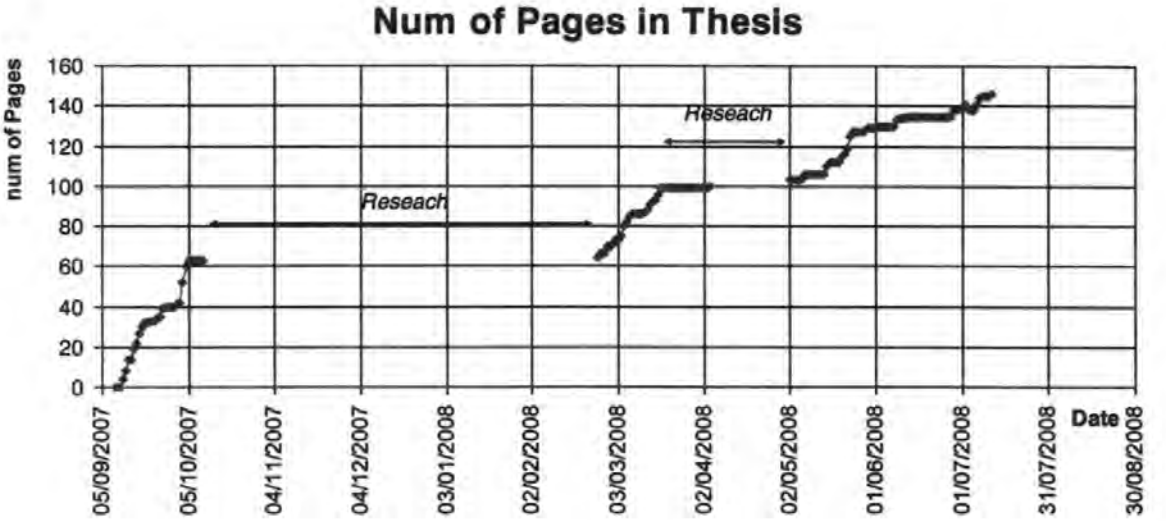
Appendix A9: Part of Prolog Code: unify_and_learn

```
unify_and_learn(TNow,RULENAME_ACTION,RULENAME_KNOWLEDGE,LRI):-
        (current_dialogue_state([learning,_,_,_]);current_dialogue_state( [clear_issue,_,_,_])),
        once(latest_event(SoS,unused)),
        once(prev_event(SoS,SoS_Prev,unused)),
        clause_linker_pre(TNow,SoS,SoS_Prev),
        (unify_and_learn_knowledge(TNow,SoS,SoS_Prev,RULENAME_KNOWLEDGE);RULENAME_KNOWLEDGE='none',true),
        (unify_and_learn_action(TNow,RULENAME_ACTION,LRI_Learn);(RULENAME_ACTION='none',true)),
        clause_linker_post(TNow,SoS,SoS_Prev),
        once(past_instruction(SoS,SoS_Prev_Instr)),
        check_for_imitate_now(TNow,SoS,SoS_Prev_Instr,LRI_Now),
        append(LRI_Learn,LRI_Now,LRI),
        write(TNow),write(' LRI (unify_and_learn/4) at '),write(TNow),write(' = '),write(LRI),write('\n').

% ------ knowledge base primitves ----
unify_and_learn_knowledge(TNow,SoS,SoS_Prev,_):-
        log('unify_an_learn_knowledge (SoS_Prev) was called at '),log(TNow),log('\n'),
        ((
forall(language_event(UI_Prev,1,KBPRIM_Prev,PARAM_Prev,SoS_Prev,UNTIL_Prev,unused),(log('trying:'),log(KBPRIM_Prev),log('\n'),(kb_primitive(KBPRI
M_Prev,UI_Prev,SoS_Prev,PARAM_Prev);true)))
        );(true)),
        log('unify_an_learn_knowledge (SoS) was called at '),log(TNow),log('\n'),
        ((forall(language_event(UI,1,KBPRIM,PARAM,SoS,UNTIL,unused),(log('trying:'),log(KBPRIM),log('\n'),(kb_primitive(KBPRIM,UI,SoS,PARAM);true
))));(true)).

% ------ action primitives -----------
unify_and_learn_action(TNow,RULENAME,LRI) :-
        latest_event(SoS,unused),
        prev_event(SoS,SoS_Prev,unused),
        log('\n\n--- unify_an_learn_action was called at '),log(TNow),log(' ---\n'),
        % make a list of gestures
        ((
                % setof returns an ordered list of unused SoGs (SoG = Start of Gesture)
                setof( SoG ,TYPE ^ CARD ^ SRCLOC ^DSTLOC ^EoG ^ X ^ Y ^ gesture_event(2,TYPE,CARD,SRCLOC,DSTLOC,SoG,EoG,X,Y,unused), List)
        );(
                % no gestures in the knowledge base
                List = []
        )),
        % grouping of gestures, linking to speech events
        grouping(TNow,SoS, SoS_Prev, List, GestureGroups, SoS_Selected, SoG_Selected,LRI ),!,
        reverse(GestureGroups,GestureGroupsRev),            % reverse the order so the latest information is considered first
        log('GestureGroupsRev = '),log(GestureGroupsRev),log('\n'),
        filter_unfinished_groups(TNow,GestureGroupsRev,GestureGroupsRevFiltered, LRI),
        ((var(LRI),LRI=['none']);(true)),
        % gesture group finished, no LRI for further recognition required    % otherwise, LRI contains schedule_gesture_recognition
        try_unification(SoS,SoS_Prev,GestureGroupsRevFiltered,SoG_Selected,SoS_Selected,LRI,RULENAME,INSTRUCTION_ID).
```

Appendix A10: Thesis Log



Progress monitoring was carried out with a spreadsheet during my write-up year. The number of pages written in the thesis was logged. It was a good indicator for predicting submission and progress, although the number of pages is not the soul indicator of progress, it was a good tool for motivating myself on a daily basis. At two points I decided to stop writing and complete the research. It proved to be hard to write and do research at the same time.

Appendix A11: Noun Phrase Anaphora (ref-res.pl)

```
% ------------------------------------------------------
% ---   MIBL Project
% ---   AI software for MIBL card-game-robot
% ---   Centre for Robotics and Intelligent Systems
% ---   University of Plymouth, U.K.
% ---   Joerg Wolf (joerg.wolf@plymouth.ac.uk)
% ---   2006, 2007, 2008
% ref_res.pl : Noun Phrase Anaphora
% ------------------------------------------------------


% ref_past/1 are grammar keywords that indicate a reference to gesture or past utterance

% ref_past('ns'). ns was taken out. for example "deal three cards" has no article (ns) because
of the introduction of a new object

ref_past('ddd').
ref_past('?').
ref_past('dpd').
ref_past('them').
ref_past('it').

refers_to_physical_object(OBJECT) :-
        ontology(OBJECT,'object3d').

refers_to_physical_object(OBJECT) :-
        is_a_instance(OBJECT,CLASS),
        ontology(CLASS,'object3d').

%refers_to_physical_object('cards').


reference_resolution_object(SoS, PRIM, NP, OBJECT_RET ) :-
        once(prev_instruction(SoS,SoS_Prev)),
        split_np(NP,REFERENCE,OBJECT,NP_LOC),
        refers_to_physical_object(OBJECT),
        write('reference_resolution_object/4: Object given\n'),
        OBJECT_RET = OBJECT.

reference_resolution_object(SoS, PRIM, NP, OBJ ) :-
        once(prev_instruction(SoS,SoS_Prev)),
        split_np(NP,REFERENCE,OBJECT,NP_LOC),
        write('\nreference_resolution_object/4: request object resolution for
'),write(OBJECT),write(' at SoS='),write(SoS),write('\n'),
        % XXX: is there a mistake in ref_res_rule making OBJECT_RET return the wrong answer?
        ref_res_rule(SoS,SoS_Prev,PRIM,REFERENCE,OBJECT,OBJECT_RET,InstructionID),
        get_property(InstructionID,'object',OBJ),
        write('\nreference_resolution_object/4: for InstID='),write(InstructionID),write('
obj='),write(OBJ),write('\n'),
        refers_to_physical_object(OBJ).

reference_resolution_n(SoS, PRIM, NP,N ) :-
        once(prev_instruction(SoS,SoS_Prev)),
        split_np(NP,REFERENCE,OBJECT,NP_LOC),
        ref_res_rule(SoS,SoS_Prev,PRIM,REFERENCE,OBJECT,_,InstructionID),
        get_property(InstructionID,'n',N).

reference_resolution_last_loc(SoS, PRIM, NP,LOC_REQUESTED,LOC ) :-
        once(prev_instruction(SoS,SoS_Prev)),
        split_np(NP,REFERENCE,OBJECT,NP_LOC),
        ref_res_rule(SoS,SoS_Prev,PRIM,REFERENCE,OBJECT,_,InstructionID),
        get_property(InstructionID,'instruction-type',REF_INST_TYPE),
        ((      REF_INST_TYPE == 'action-move',
                get_property(InstructionID,'dest-loc',LOC)
        );(
                REF_INST_TYPE == 'ifloc',
                LOC_REQUESTED \== 'dest-loc',
```

```
                get_property(InstructionID,'location',LOC)
        );(
                REF_INST_TYPE == 'action-turn',
                get_property(InstructionID,'source-loc',LOC)
        )).

% -----------------------------------------------------
% -----------------------------------------------------
% -----------------------------------------------------



% "SoS: a-X"
% "SoS_Prev _"
ref_res_rule(_,_,_,'a',OBJECT,OBJECT_RET):-          % comment: using 'a' and before 'ddd'
could be a context change indicator
%       write('ref_res_rule/7 try Rule 1\n'),
        OBJECT_RET = OBJECT,
        write('ref_res_rule/7 (SoS_Prev: _  ... SoS: a-X) \n').



% -----------------------------------------------------
% XXX: EXPERIMENTAL REFERENCE RESOLUTION , EVEN IF PREVIOUS UTTERANCE IS ALSO A REFERENCE
(them)
% antecedent anaphora ( reference to prior mention )
% "SoS_Prev: ddd-X ... SoS: them-X"
% "SoS_Prev: them-X ... SoS: ddd-X"
ref_res_rule(SoS,SoS_Prev,_,REF,UNKNOWN_OBJECT,OBJECT_RET,InstructionID) :-
        ((UNKNOWN_OBJECT == '?');(UNKNOWN_OBJECT == 'ns');(UNKNOWN_OBJECT == 'cardname')),
        ref_past(REF),
        language_event(UI,_,PRIM,PARAM,SoS_Prev,_,InstructionID),
        prefixcomma_once(NP,PARAM,44),
        split_np(NP,REFERENCE,OBJECT_EARLIER,NP_LOC_EARLIER),
        ref_past(REFERENCE),
%       write('ref_res_rule/7 try Rule 2\n'),
        OBJECT_RET = OBJECT_EARLIER,
        write('ref_res_rule/7 (SoS_Prev: ddd-X ... SoS: them-X or them-X ... SoS: ddd-X):
reference to SoS='),write(SoS),write('primitive='),write(InstructionID),write('\n').

% -----------------------------------------------------


% -----------------------------------------------------
% antecedent anaphora ( reference to prior mention )
% ddd = determinative demonstrative deictic reference
ref_res_rule(SoS,SoS_Prev,_,REF,'?',OBJECT_RET,InstructionID) :-
        ref_past(REF),
        language_event(UI,_,PRIM,PARAM,SoS,_,InstructionID),
        prefixcomma_once(NP,PARAM,44),
        split_np(NP,REFERENCE,OBJECT_EARLIER,NP_LOC_EARLIER),
%       write('ref_res_rule/7 try Rule 3\n'),
        REFERENCE='a',
        OBJECT_RET = OBJECT_EARLIER,
        write('ref_res_rule/7 (SoS: a-X     ? OBJECT_EARLIER): reference to
SoS='),write(SoS),write('primitive='),write(InstructionID),write('\n').

% -----------------------------------------------------
% "SoS: a-X     ... SoS: ddd-X"
ref_res_rule(SoS,SoS_Prev,_,REF,OBJECT_NOW,OBJECT_RET,InstructionID) :-
        ref_past(REF),
        language_event(UI,_,PRIM,PARAM,SoS,_,InstructionID),
        prefixcomma_once(NP,PARAM,44),
        split_np(NP,REFERENCE,OBJECT_EARLIER,NP_LOC_EARLIER),
%       write('ref_res_rule/7 try Rule 4\n'),
        REFERENCE='a',
        ((OBJECT_NOW \== '?', OBJECT_EARLIER = OBJECT_NOW);(true)), % proves the link to the
SoS
        OBJECT_RET = OBJECT_NOW,
        write('ref_res_rule/7 (SoS: a-X     ... SoS: ddd-X): reference to
SoS='),write(SoS),write('primitive='),write(InstructionID),write('\n').

% -----------------------------------------------------
% "SoS_Prev: a-X ... SoS: ddd-X"
ref_res_rule(SoS,SoS_Prev,_,REF,OBJECT_NOW,OBJECT_RET,InstructionID) :-
```

```
        ref_past(REF),
        language_event(UI,_,PRIM,PARAM,SoS_Prev,_,InstructionID),
        prefixcomma_once(NP,PARAM,44),
        split_np(NP,REFERENCE,OBJECT_EARLIER,NP_LOC_EARLIER),
%       write('ref_res_rule/7 try Rule 5\n'),
        REFERENCE='a',
        ((OBJECT_NOW \== '?', OBJECT_EARLIER = OBJECT_NOW);(true)), % proves the link to the
SoS
        OBJECT_RET = OBJECT_NOW,
        write('ref_res_rule/7 (SoS_Prev: a-X ... SoS: ddd-X): reference to
SoS_Prev='),write(SoS_Prev),write('primitive='),write(InstructionID),write('\n').

% ref_res_rule(491,104,turn,them,?,_49233,_53955) ?

% ------------------------------------------------------
% "SoS_Prev: ns-X ... SoS: ddd-X"
% "ns-" means no article, which usually indicates an introduction of a new object
ref_res_rule(SoS,SoS_Prev,_,REF,OBJECT_NOW,OBJECT_RET,InstructionID) :-
        ref_past(REF),
        language_event(UI,_,PRIM,PARAM,SoS_Prev,_,InstructionID),
        prefixcomma_once(NP,PARAM,44),
        split_np(NP,REFERENCE,OBJECT_EARLIER,NP_LOC_EARLIER),
%       write('ref_res_rule/7 try Rule 6\n'),
        REFERENCE='ns',
        ((OBJECT_NOW \== '?', OBJECT_EARLIER = OBJECT_NOW);(true)), % proves the link to the
SoS
        OBJECT_RET = OBJECT_NOW,
        write('ref_res_rule/7 (SoS_Prev: ns-X ... SoS: ddd-X): reference to
SoS_Prev='),write(SoS_Prev),write('primitive='),write(InstructionID),write('\n').



% for debugging:
test_ref_res(SoS) :-
        language_event(UI,_,PRIM,PARAM,SoS,_,_),
        prefixcomma_once(NP,PARAM,44),
        reference_resolution_object(SoS, PRIM, NP, OBJECT_RET ),
        write('test_ref_res/1: '),write(PRIM),write('('),write(PARAM),write(')\n refers to:
'),
        write(OBJECT_RET),write('\n').


split_np(NP,REFERENCE,OBJECT,NP_LOC) :-
        bagof(J,splitcomma(J,NP,45),NP_ITEMS),
        nth(1,NP_ITEMS,REFERENCE,_),
        nth(2,NP_ITEMS,OBJECT,_),
        nth(3,NP_ITEMS,NP_LOC,_).
```

Appendix A12: Dealing Playback Exp. E3.1

Rule Frames of Dealing Phase after playback of data set-1: For explanation see chapter 7.

03.xml	Rule217	playing	2366				
move635	action-move	ns-cardname-ns	3	+table+stock	+temp2	cardname	2431
turn763	action-turn	ddd-?-ns	4	+table	11730	cardname	2821
move399	action-move	ns-cardname-ns	4	+table+stock	+table	cardname	2708
turn848	action-turn	them-?-ns	1	+hand1	11730	cardname	2606
move686	action-move	them-?-ns	1	+temp2	+hand2	11730	2564
move830	action-move	ddd-?-ns	3	+temp1	+hand1	cardname	2540
move266	action-move	?-?-ns	3	+table+stock	+temp1	cardname	2482
06.xml	Rule217	playing	474				
move635	action-move	ddd-cardname-ns	3	+table+stock	+temp1	cardname	558
move763	action-move	ns-cardname-ns	3	+table+stock	+temp1	10933	504
move399	action-move	ns-cardname-ns	3	+table+stock	+temp2	10933	504
07.xml	Rule217	playing	741				
move635	action-move	ns-cardname-ns	3	+table+stock	+temp2	cardname	771
loop763	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	771
	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	771
turn399	action-turn	ddd-?-ns	?	+table	10933	cardname	1050
move331	action-move	ns-cardname-ns	4	+table+stock	+table	cardname	1007
11.xml	Rule217	playing	1161				
move635	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	1240
loop763	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	1240
	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	1240
move399	action-move	ddd-cardname-ns	2	+temp1	+hand1	cardname	1300
turn240	action-turn	them-?-ns	4	+table	10933	cardname	1553
move331	action-move	ns-cardname-ns	4	+table+stock	+table	cardname	1490
12.xml	Rule217	playing	50				
move635	action-move	ddd-cardname-ns	2	+temp1	+hand1	cardname	1116
move763	action-move	ns-cardname-ns	3	+table+stock	+temp1	775	1083
move240	action-move	ns-cardname-ns	3	+table+stock	+temp2	775	1069
turn331	action-turn	them-?-ns	4	+table	775	cardname	1351
move848	action-move	ns-cardname-ns	4	?	+table	775	1269
turn271	action-turn	ddd-?-ns	3	+hand1	775	cardname	1194
14.xml	Rule217	playing	0				
turn635	action-turn	ddd-cardname-ns	3	+hand1	10957	cardname	7177
turn763	action-turn	them-?-ns	3	+hand2	10957	cardname	7160
move331	action-move	ddd-?-ns	1	4	+hand2	10957	7061
move686	action-move	?-?-ns	3	+temp1	+hand1	cardname	7043
move271	action-move	?-?-ns	3	+table+stock	+temp1	cardname	7001
move830	action-move	?-?-ns	3	+table+stock	+temp2	cardname	6982
19.xml	Rule217	playing	1085				
move635	action-move	ns-cardname-ns	3	+table+stock	+temp1	10933	1115
turn240	action-turn	ddd-?-ns	2	+table	10933	cardname	1354
move331	action-move	ddd-cardname-ns	4	+table+stock	+table	cardname	1314
turn848	action-turn	them-?-ns	3	+hand1	10933	cardname	1272
move686	action-move	ddd-?-ns	3	+temp1	+hand1	cardname	1207
20.xml	Rule217	playing	1074				
move635	action-move	ns-cardname-ns	3	+table+stock	+temp2	cardname	1104
loop763	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	1104
	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	1104
move331	action-move	ddd-cardname-ns	4	+table+stock	+table	cardname	1347
turn848	action-turn	them-?-ns	3	+hand1	10933	cardname	1303
move686	action-move	them-?-ns	3	+temp1	+hand1	cardname	1225
turn271	action-turn	them-?-ns	2	+table	10933	cardname	1433
21.xml	Rule217	playing	885				
move635	action-move	ns-cardname-ns	3	+table+stock	+temp2	cardname	915
loop763	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	915
	action-move	ns-cardname-ns	3	+table+stock	+temp1	cardname	915
move399	action-move	ddd-?-ns	2	+temp1	+hand1	cardname	987
turn240	action-turn	ddd-?-ns	2	+hand1	775	cardname	1020

turn331	action-turn	ddd-?-ns	4	+table	775	cardname	1173
move848	action-move	ns-cardname-ns	4	+table+stock	+table	cardname	1123

Appendix A13: Pilot Grammar

```
Pilot_Construct[

;--- pilot_written-rules.xml.txt dealing ---
        (game one)
        (CardCatNP:cn1 on the table with the same value as CardCatNP:cn2 in your hand are a pair ) (return(
strcat(strcat(strcat(strcat(strcat(strcat("ifloc=a-cardname-ns,1,+table:ifloc=a-cardname-ns,1,+hand2:ifcond=equal,value," $cn1) ",") $cn2) ",")
"?-?-ns,?") ":") ))
        (you can take CardCatNP:cn to the side as a capture) (return( strcat(strcat("move=" $cn)  ",?,+side2:") )  )
        (?all NumCatSPRecursive:cclist have been [(taken out) removed] from the [deck pack game])(return(strcat(strcat("not_exist=" $cclist)
":")))

;--- pilot_sub01try1.xml.txt dealing ---
        (you deal CardCatNP:cn) (return( strcat(strcat("move=" $cn)  ",+table+stock,?:") )  )
        (you have to deal CardCatNP:cn for each player)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+temp1") ":")
"loopindicator=each:") )  )
        (?no deal CardCatNP:cn [for to] each player)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+temp1") ":")
"loopindicator=each:") )  )
        (now you have to deal CardCatNP:cn in the middle of the table) (return( strcat(strcat("move=" $cn)  ",+table+stock,+table:") )  )
        (deal CardCatNP:cn) (return( strcat(strcat("move=" $cn)  ",+table+stock,?:") )  )
        (deal CardCatNP:cn) (return( strcat(strcat("clause_relation=move,move,modify:move=" $cn)  ",?,+table:") )  )
        (place CardCatNP:cn in the centre of the table face up) (return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+table:turn=")
$cn) ",up:") )  )
        (take the cards from the deck) (return(":reply=+table+stock:")  )
        (and place CardCatNP:cn in the middle of the table) (return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+table:turn=") $cn)
",up:") )  )
        (the middle of the table) (return(":reply=+table:")  )
        (CardCatNP:cn face up)(return( strcat(strcat(strcat(strcat("turn=" $cn)          ",") "?" ) ":") )  )
        (CardCatNP:cn must face up)(return( strcat(strcat(strcat(strcat("turn=" $cn)          ",") "?" ) ":") )  )
        (CardCatNP:cn should face up)(return( strcat(strcat(strcat(strcat("turn=" $cn)          ",") "?" ) ":") )  )
        (CardCatNP:cn face up)(return( strcat(strcat(strcat(strcat("turn=" $cn)          ",") "?" ) ":") )  )
        ([put place] CardCatNP:cn in the middle of the table)(return( strcat(strcat("move=" $cn)  ",+table+stock,+table:") )  )

;--- pilot_sub01try2.xml.txt dealing ---
        (first you deal CardCatNP:cn for each player)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+temp1") ":")
"loopindicator=each:") )  )
        (?(i said) deal CardCatNP:cn for each player)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+temp1") ":")
"loopindicator=each:") )  )
        (ignore what i just said )(return( ":dm=erase:" ))
        (now you have to deal CardCatNP:cn) (return( strcat(strcat("move=" $cn)  ",+table+stock,?:") )  )
        (place CardCatNP:cn in the centre of the table facing up) (return( strcat(strcat(strcat(strcat("move=" $cn)
",+table+stock,+table:turn=") $cn) ",up:") )  )
        ([put place] CardCatNP:cn in the middle of the table)(return( strcat(strcat("move=" $cn)  ",+table+stock,+table:") )  )
```

```
;--- pilot_sub02try1.xml.txt dealing ---

;--- pilot_sub03try1.xml.txt dealing ---
        (ReturnNumber:n cards to me ReturnNumber:n cards to you)( return(strcat( strcat(":reply=" $n) ":")))
        (to each player)(return(":reply=+hand2:"))
        (to each player)(return(":reply=+hand2:"))
        (turn CardCatNP:cn face up)(return( strcat(strcat(strcat(strcat("turn=" $cn)          ",") "?")  "up") ) )
        (turn face up CardCatNP:cn)(return( strcat(strcat(strcat(strcat("turn=" $cn)          ",") "?")  "up") ) )

;--- pilot_sub03try2.xml.txt dealing ---
        (deal CardCatNP:cn [for to] each player)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+temp1") ":")
"loopindicator=each:") )  )
        ([on onto] my hand and [on onto] your hand)(return(":reply=+hand1:reply=+hand2:")  )
        (NumberCat:N [to into from] LocationN:L)(return( strcat(strcat(strcat(strcat(":reply=" $N) ":reply=") $L) ":"))  )
        (turn CardCatNP:cn face up)(return( strcat(strcat(strcat(strcat("turn=" $cn)          ",") "?")  "up") ) )
        (turn face up CardCatNP:cn)(return( strcat(strcat(strcat(strcat("turn=" $cn)          ",") "?")  "up") ) )

;--- pilot_sub04try1.xml.txt dealing ---
        (i am going to deal ?another CardCatNP:cn for you and me)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+temp1") ":")
"loopindicator=each:") )  )
        (?(i said) i am going to deal ?another CardCatNP:cn)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,?") ":") "") )  )
        (i am going to deal ?another CardCatNP:cn for me)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+temp1") ":") "") )  )
        (i am going to deal ?another CardCatNP:cn [on to into] my place)(return( strcat(strcat(strcat(strcat("move=" $cn)
",+table+stock,+hand1") ":") "") )  )
        (the cards go LocationN:il)(return(strcat(strcat(":reply=" $il) ":")))
        (the card goes LocationN:il)(return(strcat(strcat(":reply=" $il) ":")))
        (?(i said) ?now i am going to deal ?another CardCatNP:cn [on onto] the table)(return( strcat(strcat(strcat(strcat("move=" $cn)
",+table+stock,+table") ":") "") )  )
        (?please deal ?another CardCatNP:cn [on onto] the table)(return( strcat(strcat(strcat(strcat("move=" $cn)  ",+table+stock,+table") ":")
"") )  )
        (?please turn CardCatNP:cn up ?please)(return( strcat(strcat(strcat(strcat("turn=" $cn)  ",up") ":") "") )  )
        (?please turn CardCatNP:cn over ?please)(return( strcat(strcat(strcat(strcat("turn=" $cn)  ",?") ":") "") )  )

;--- pilot_sub04try2.xml.txt dealing ---

)
```

Appendix A14: MIBL Manual

Adding a new grammar construct (Version 04/2008 with <grammar>):

1. Edit the `mmtransdef.dtd` file to add the tag
2. In C:\MIBL\MIBL_glmgrammar\elements.h
   - add a new `#define EL_TYPE_...` at the end of the list
   - increase `#define EL_AMOUNT_OF_TYPES` by one
   - add the tag to `static const string ElementName[...` at the end
3. Recompile xml-filter.exe and cgr.exe
4. A grammar can be created that includes all <grammar> tags found inside the newly created tags by this batch file: (in the example the xml tag is called pairrule)

```
call config.bat

set FILE=pairrule

set XMLFILELIST=%XMLMASTERFOLDER%/completeset.txt

del /Q grammar/GAME.%FILE%.grammar


FOR /F %%i IN (%XMLFILELIST%) DO xml-filter\Debug\xml-filter.exe
%XMLMASTERFOLDER%/%%i %FILE%_%%i.txt %FILE% -grammar


echo PairRule_Construct[ > grammar/GAME.%FILE%.grammar

FOR /F %%i IN (%XMLFILELIST%) DO type %FILE%_%%i.txt>> grammar/GAME.%FILE%.grammar

echo ] >> grammar/GAME.%FILE%.grammar


FOR /F %%i IN (%XMLFILELIST%) DO del /Q %FILE%_%%i.txt

pause
```

5. The example batchfile contains the name of the new grammar rule (PairRule_Construct). This must be added to the to the `.Tagging` section in the main-grammar (`GAME.grammar`)
6. The beginning of the main-grammar (`GAME.grammar`) must include the new grammar file (`#include "grammar/GAME.pairrule.grammar"`)
7. Add the name of the grammar rule to the file `%GRAMMAR%.clause-level-taglist`

Adding a new grammar construct (Old Version):

- tag the utterances with a new XML tag
- copy the xml2capture-rule-set.bat and modify the word "capturerule" to whatever the XML tag is
- Give the output file a name by modifiying RULECORPUS=capture-rule-set.txt
- run the batchfile
- a file lut_split.%RULECORPUS% should have appeared which is a list of utterances ready to be pasted into the grammar
- Create a new grammar-construct for example:

RuleX_Construct [

Paste your lut_split-file into here

]

- Add RuleX_Construct to the .Tagging section in the main-grammar
- Add RuleX_Construct to the file % GRAMMAR %.clause-level-taglist
- Add the semantics to each line in the grammar
- Run COMPILE_ALL.bat
- Finished

Software requirements

The system has been tested with Microsoft Windows 2000 and XP. Parts of it have been tested in Redhat Linux 7.3.

Software	Ver.	Directory	System Env. Variables
Nuance	8.5	\Nuance85\V8.5.0	NUANCE
Nuance Vocalizer	4.0	\Nuance85\Vocalizer4.0	
Trolltech Qt	3.3.3	\Qt\3.3.3Educational	QTDIR,QMAKESPEC
SICStus Prolog	3.10.0	\Program Files\SICStus Prolog 3.10.0	(SP_PATH)
MS Visual C++ (with SP6)	6.0 SP6	\Program Files\Microsoft Visual Studio	* LIB, INCLUDE, MSDevDir
MIBL folder		\MIBL	
NL Tools	-	\MIBL\nltools	
Programmers Notepad (optional)	2		
XEmacs (optional)	21.4.21	C:\Program Files\XEmacs\XEmacs-21.4.21\	

* At the end of the installation of Visual C++ 6 there is an opportunity to tick a box to "register environment variables". If this was not done, the batch file VCVARS32.BAT has to be executed every time before compiling code from the command line.

(\Program Files\Microsoft Visual Studio\VC98\Bin\VCVARS32.BAT)

Environment Varables required to run cgr.exe without calling config.bat:

(this is necessary for debugging with Visual Studio):

NUANCE=\Nuance85\V8.5.0

QMAKESPEC=win32-msvc

QTDIR=C:\Qt\3.3.3Educational

SP_PATH=\PROGRA~1\SICSTU~1.0\bin\

PATH variable must include ;%NUANCE%\bin\win32;%SP_PATH%;%QTDIR%\bin;

MS Visual Studio 6 project settings:

create_vc-project.bat contains a command that creates a Visual Studio project from a Qt project with "qmake -t vcapp server.pro".

After opening this automatically created Visual Studio project confirm the following project settings:

Debug | General | Working directory: \MIBL\cgr

Debug | General | Program arguments:

-package \MIBL\MIBL_glmgrammar\grammar\GAME lm.Addresses=localhost rm.Addresses=localhost client.TTSAddresses=localhost audio.InputVolume=255 fe.EnableNoiseReduction=TRUE lm.Addresses=10.0.0.7 rm.Addresses=10.0.0.7 client.TTSAddresses=10.0.0.7

C++ | General | Preprocessor definitions:

WIN32,DEBUG,_WINDOWS,UNICODE,XML_STATIC,QT_DLL,QT_THREAD_SUPPORT

C++ | Preprocessor | Additional include directories:

C:\Nuance85\V8.5.0\include,C:\PROGRA~1\SICSTU~1.0\include,$(QTDIR)\include,\MIBL\cgr\cardgamerobot,C:\Qt\3.3.3Educational\mkspecs\win32-msvc

Link | General | Object/Library modules:

"qt-mtedu333.lib" "qtmain.lib" "opengl32.lib" "glu32.lib" "delayimp.lib" "kernel32.lib" "user32.lib" "gdi32.lib" "comdlg32.lib" "advapi32.lib" "shell32.lib" "ole32.lib" "oleaut32.lib" "uuid.lib" "imm32.lib" "winmm.lib" "wsock32.lib" "winspool.lib" **"C:\Nuance85\V8.5.0\lib\win32\rcapi.lib" "\PROGRA~1\SICSTU~1.0\bin\spaux.obj" "\PROGRA~1\SICSTU~1.0\bin\sprt310.lib" "nli\libexpatMT.lib"** "kernel32.lib"

"user32.lib" "gdi32.lib" "comdlg32.lib" "advapi32.lib" "shell32.lib" "ole32.lib" "oleaut32.lib" "uuid.lib" "imm32.lib" "winmm.lib" "wsock32.lib" "winspool.lib"

Link \ Input \ Additional library path: $(QTDIR)\lib

MS Visual Studio 6 Environment Variables:

(the path is actually written into the variable in 8.3 DOS style pathname. see VCVARS32.BAT)

include=C:\Program Files\Microsoft Visual Studio\VC98\atl\include;C:\Program Files\Microsoft Visual Studio\VC98\mfc\include;C:\Program Files\Microsoft Visual Studio\VC98\include

lib=C:\Program Files\Microsoft Visual Studio\VC98\mfc\lib;C:\Program Files\Microsoft Visual Studio\VC98\lib

MSDevDir=C:\Program Files\Microsoft Visual Studio\Common\MSDev98

Path includes:

C:\Program Files\Microsoft Visual Studio\Common\Tools\WinNT;C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin;C:\Program Files\Microsoft Visual Studio\Common\Tools;C:\Program Files\Microsoft Visual Studio\VC98\bin

Redhat Linux 7.3:

Redhat Linux 7.3 was the platform chosen for the IBL project, and therefore all the tools used in the MIBL project have been chosen to be compatible with this operating system.

For a Linux installation, Nuance 8.0 requires Redhat Linux 7.3 since it is only compatible with the glibc library of that version. Attempts to make Nuance 8 run on a newer linux (Fedora Core 2,5,9) failed, because of glibc incompatibilities Nuance crashes. Sicstus Prolog 3.10.0 also works with Redhat Linux 7.3. The status of Qt 3.3.3 in RH 7.3 is unknown, but likely to be working (XXX), alternatively the Qt commands can be modified to be compatible with a earlier version of Qt 3 which works on Redhat Linux 7.3

Hardware Requirements

Minimum:

1 GByte of RAM

1.6 GHz Pentium 4

Recommended:

2 PCs with 2.4 GHz Dual Core, each

1 GByte of RAM, each

For Optimum Performance a cluster of 3-4 CPUs or PCs is necessary, optimum performance on a single PC can probably achieved with an Xeon QuadCore.

If split into several single core PCs:

For optimum performance of the robot use 1 PC for Nuance, 1 PC for CGR, 1 PC for GUI.

For optimum performance while recording, 1 PC for Teacher GUI, 1 PC for Student GUI, 1 PC for CGR-recording, (Sound recording could be done on the GUI PCs)

(with less than 1GByte RAM the swapping to harddrive causes the system to slow down and eventually crash)

Running Nuance from another machine

1. Make a copy of the \Nuance85 folder to the Nuance PC
2. Share the C drive and map it at the Nuance PC
3. open C:\MIBL\cgr\windows_batchfiles\config.cfg
4. change "set NUANCE_HOST=192.168.0.2" to the IP of the Nuance computer
5. start Nuance by double-click on: X:\MIBL\cgr\windows_batchfiles\ start_all.bat

Prolog Commands:

To start Prolog: ("C:\Program Files\SICStus Prolog 3.10.0\bin\spwin.exe")

Initially always load the MIBL knowledge base:

```
consult('C:/MIBL/cgr/cardgamerobot/AI/kb.pl').
```

To repeat the interaction that was last recorded and filled the knowledge base with game-rules load (consult) the logged data.

```
consult('/mibl/cgr/logfiles/datalog.pl').
```

Command	Description
`show_language_events.`	show all speech events
`show_state_vector.`	Shows where all cards and their location (in the robots mind)
`show_transcript.`	Show the transcript of what user and robot said
`dumpkb('Rule217').`	To save the rule frames to a CSV file
`showkb.`	Show the tree in the knowledge base that contains the rule frames
`nextstep(LRI, _ ,_ ).`	Run Problem Solver to see what the robot would do next
`show_str.`	Show State-Transition-Rules
trace.	Build in Prolog command that activates the debbuger. (When using XEmacs with the sicstus_emacs_init.el extension for Prolog, it is possible to do in-source-debbugging, which means the line of the source code that is currently traced is highlighted)
`current_dialogue_state(X).`	Show dialogue state vector.
`dumpxml.`	Save all language and gesture events from the database as XML transcription. (useful for error analysis and assisted transcription)

Modes of Use :

There are two principle ways to use the CGR server software. In both cases the robot will try to learn a task:

1. playback a corpus recording (Corpus Playback Mode)
2. online robot-teacher interaction (Online Mode)

Corpus Playback Mode:

A corpus consisting of CSV and XML file can be played back to the CGR software. The software will switch the robot into learning mode and a "datalog.pl" is produced for future refence. During this playback, the microphone can also be used at any time for online interaction. Online gestures, however should be avoided (situation untested), since they could interfere with the already recorded CSV data.

To play back a part of the corpus to the robot follow this procedure:

1. load CSV file
2. load XML file
   You will be prompted to choose a tag that marks the part of the XML file that should be played back. By default the dialogue has "pairrule no=1" which means that the tag <pairrule no="1"> will be looked for in the XML file and all teacher utterances <tv> will be scheduled for playback.
3. press play and leave it play for 10 seconds until the 3D objects appear.
   (The CSV recordings contain the objects that have to be created at

   the beginning of the file.)
4. look at the Log-output and search for when the first Speech Event is scheduled
   For Example: "`Scheduled MultiModal Speech Event ... 155.9 ...`"
5. forward the timestamp cursor to shortly before that Speech Event to skip the uninteresting part of the recordings (For example 155.9 means timestamp 1559) The first speech event is actually created by the CGR software to switch the robot into learning mode.
6. Press play and see how the robot learns.
   It is a good idea to monitor if all necessary semantics appear.

   It is allowed to pause at any time.

7. If the robot has a question, the dialogue_manager can be configured to pause the conversation, so that the system can continue to use the timestamps from the recording when resuming after the question. Use the LRI command `system('pause')` in the dialogue_manager which will toggle the pause button.

8. After all the scheduled Speech Events that you are interested in have passed, the playback is finished. The CGR software can be closed for offline analysis.

Online Mode:

1. Press "REC"
2. Press "Create Objects" from the GUI tab
3. say "game one" to bring the robot into learning mode
4. teaching the robot something :)
   The usual log files are created from the recording and interaction.

the assert_transcript in datalog.pl could be used as a transcription of the dialog.

Data and Log Files:

Filename	Description
cgr/logfiles/temp.csv	CSV file with the gestures
cgr/logfiles/datalog.pl	contains everything that goes into the Prolog Engine It is very useful to consult this file offline for debugging !
cgr/logfiles/log.txt	Output log from the CSV server
cgr/logfiles/prologlogfile.txt	Output from the prolog command `log()` outputs with `write()` onto the prolog-console are not logged
cgr/logfiles/rec_utt_file.csv	Contains an index of the recorded wav-files from the recserver Format: SoS,EoS,EoR,uttfilename,recognizedstring filename can be changed from the CGR Control Application Menu
default-prolog-logfile.txt	Prolog logfile replacement for prologlogfile.txt that is created when kb.pl is consulted manually rather than from the CSV server
cgr/logfiles/dumpxml.xml	a multi-modal XML transcription of the utterances and gestures stored in the robots database

Data Analysis after interaction:

(Also see section Prolog Commands)

1. start Prolog
2. consult kb.pl
3. consult datalog.pl
   (truncate the datalog.pl file to the point in time that is interesting for analysis)

4. use the prolog command (such as showkb.) described earlier to find out what the robot is doing.
5. Common is to activate the debugger with "trace." and then continue with a line out of the datalog.pl manually, for example : unify_and_learn(117,RA,RK,LRI).

Adding to the corupus after (live) interaction:

1. Follow steps 1-3 of Data Analysis after interaction to load the data.
2. type dumpxml. into the prolog console which creates cgr/logfiles/dumpxml.xml.
3. move and rename dumpxml.xml to the XML-Masterfolder
4. load an audio recording that has been done during the interaction with an accurate wave-editor such as Magix Samplitute.
5. Confirm or correct the teachers voice "<tv>", since the speech recognition is not perfect.
6. add grammar tags <grammar> wherever necessary. If the grammar already covers the utterance can be tested with pasting the utterance into \MIBL\MIBL_glmgrammar\grammartest.bat
7. add the xml-file to the file-list in the XML-Masterfolder. For example to "pilotset.txt" or "devset.txt"
8. run \MIBL\MIBL_glmgrammar\xml-errors-show.bat to show syntax errors in the xml-file (and fix them)
9. recompile the grammar
10. test the new grammar rule with \MIBL\MIBL_glmgrammar\grammartest.bat.
    A more comprehensive test is to follow the Corpus-Playback Mode procedure

11. If new semantics have appeared that need implementation and testing, then this should be done now, since the grammar is complete and step 10 has created a new datalog.pl with the new semantics to test the implementation.
    (Pilot tests revealed that new semantics did not appear with the 4 test persons, so adding new grammar was sufficient.)

Done.

Trouble Shooting:

Symptoms:

- log message "`TTS (Vocaliser not available)`"

- No TTS (i.e. robot does not say "online" on startup)

1. check that the vocalizer is running

2. Check that parameter of the application "client.TTSAddresses= some-ip-address" is set and points to the server with the vocalizer

3. The CGR server ignores the fact that the TTS is not working. To continue without the Vocaliser you have to read the LRI instruction `say('xxxx')` in the log messages to see what the robot is trying to saying.

Symptoms:

- log message:

```
WorldModel::MoveObject Could not find object while trying to move

No object in State Memory!
```

- program crashes after interaction starts

1. you pressed "Create Objects" before "Record", do it the other way round

---

Symptoms:

- log message: `Out of Memory for World Model`

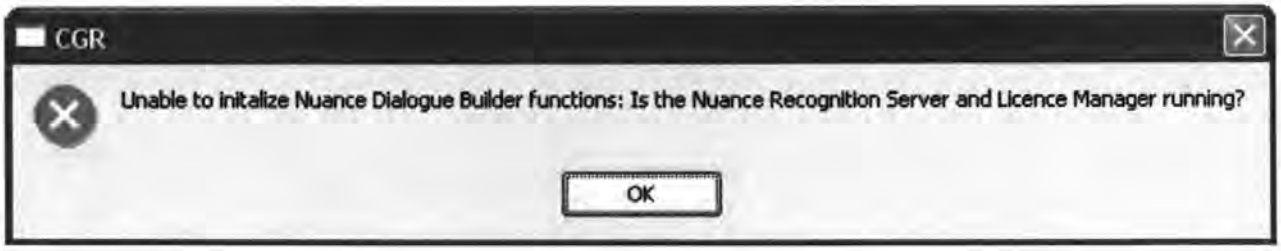
1. the number of maximum number of States in the Simulator have been reached. Unless you need to work near the end of a CSV file you can ignore this message.

2. The number of States can be incread in recorder.cpp function Recorder::Recorder

```
int EstimatedNumOfStates = 35000;
```

10000 States needs approximately 100 MByte of RAM !

Symptoms:



1. RecServer is not running. Check the red windows with the recserver. is it there ? did the recserver quit but the red window is still there ?

2. you specified the wrong lm.Addresses or rm.Addresses

3. you didnt wait long enough between starting RecServer and CGR server (give it 30 seconds)

Symptoms:

- log message: ASR(100)=<recognizer_too_slow>

1. your computer is too slow or has not enough CPUs or not enough RAM.

   See hardware requirements

Symptoms:

- log message: ASR(100)=<timeout>

  dialogue_manager...

  Goal dialogue_manager failed

1. This is not an error.
   The robot has regular recognition timeouts to initiate the dialogue_manager. Because questions to the user or other events might have occured in the meantime and the dialogue_manager has to act. It is better to have regular timeouts for a quick and natural response.

Research article

The impact of spoken interfaces on the design of service robots

G. Bugmann, J.C. Wolf and P. Robinson
School of Computing Communication and Electronics, University of Plymouth, Devon, Plymouth, UK

Abstract
Purpose – Service robots need to be programmable by their users who are in general unskilled in the art of robot programming. We have explored the use of spoken language for programming robots.
Design/methodology/approach – Two applications domains were studied: that of route instructions and that of game instructions. The latter is work in progress. In both cases work started by recording verbal instructions representative of how human users would naturally address their robot.
Findings – The analysis of these instructions reveals references to high-level functions natural to humans but challenging for designers of robots. The instruction structure reflects assumptions about the cognitive abilities of the listener and it is likely that some human capabilities for rational thinking will be required in service robots.
Research limitations/implications – Some of the high-level functions called for by natural communication stretch current capabilities and there is a clear case for more effort being devoted in some areas. Instruction analysis provides pointers to such research topics.
Practical implications – It is proposed that service robot design should start with investigating the way end-users will communicate with the robot. This is encapsulated in the "corpus-based" approach to robot design illustrated in this paper. This results in more functional service robots.
Originality/value – The paper stresses the importance of considering human-robot communication early in the robot design process.

Keywords Robotics, Design, Service delivery systems

Paper type Research paper

Introduction

The development of human-robot communication (HRC) systems is an important factor in the development of the service robot market. Effective communication between user and robot is essential to enable access to the full functional capabilities of such robots. User-friendly communication will in itself add to the perceived value and social acceptability of the robot.

Service robotics, e.g. personal assistants, poses specific and new problems of HRC that are very different from those in industrial robotics. In industrial robotics, human-robot interface are designed to enable skilled workers to generate execution programs for the robot and to command the activation of such programs. In service robotics, robots are likely to be frequently re-assigned new tasks by users who are mostly unskilled in the art of robot programming. Thus the problem becomes very much one of providing the robot with communication capabilities compatible human with natural communication channels. Typically, these channels will include both verbal and gesture inputs.

Two experiments in the design of natural language interfaces (NLIs) are summarized here. Both experiments show that requirements of natural communication constrain the design of a robot in its hardware as well as software aspects. As a consequence, the systematic use of a "corpus-based" design method is suggested. The corpus is a sample of communication acts between the user and the planned robot. Its analysis is the first step in the corpus-based robot design process.

Programming service robots using natural language

Service robots will need to be programmable by their users in new ways in order to have any chance of commercial success. This is especially true for personal assistant robots which will have as many different tasks as users. These tasks can vary in the details of their execution or constitute totally new sequences of actions. Thus, assistant robots can clearly not be fully pre-programmed by the manufacturer. Instead, they need capabilities to acquire new tasks or task specifications from their users, e.g. which pieces of furniture can be moved during cleaning, the place where items are stored, how to prepare a given variety of tea, etc.

An important constraint in the design of user-programmable service-robots is that few of the future users are likely to have the ability or inclination to learn formal programming languages. They are also likely to lack basic

The Emerald Research Register for this journal is available at
www.emeraldinsight.com/researchregister
The current issue and full text archive of this journal is available at
www.emeraldinsight.com/0143-991X.htm

Industrial Robot: An International Journal
32/6 (2005) 499–504

[ISSN 0143-991X]
[DOI 10.1108/01439910510629235]

Industrial Robot: An International Journal
Volume 32 · Number 6 · 2005 · 499–504

knowledge in robot kinematics, sensor data processing, control, etc. However, they will be experienced in instructing other humans using natural language. For that reason, we have explored the use of unconstrained spoken natural language for robot programming. There has been comparatively little research in the area of programming robotic helpers by using spoken-language. Early work includes Crangle and Suppes (1994), Torrance (1994), Huffman and Laird (1995) and Matsui *et al.* (1999). All these previous approaches used a constrained language that the user had to learn. The research described in this paper deals with issues arising from the use of unconstrained utterances generated by naïve users, i.e. those lacking technical expertise. The focus of this paper, however, is not on natural language processing or program generation, but on the implication for robot design of the use of unconstrained language.

Speech recognition has made great progress in recent years and does not constitute the bottleneck it once did. Instead, the challenge is dealing with the tight interlocking of human language and human mental and physical capabilities. This is illustrated by the findings of two projects described below. A first project focused on giving route instructions to a mobile robot. A second project, which is ongoing, focuses on game instructions. Both projects reveal that the particularities of human language have a major impact on robot design.

Instruction-based learning (IBL) for a mobile robot

In the IBL project, a robot is instructed on how to travel from one place to another in a miniature town. On the basis of the user's instructions, a computer program script is created which is then used to navigate between the two places. Route instructions were expected to contain all the main structures found in computer programs: selection, sequence and repetition. Thus, results obtained were expected to generalize to other domains.

A miniature urban environment was built on a 170 × 120 cm area suitable for navigation by a miniature 8 × 8 cm robot. Subjects were placed in front of the town and were asked to instruct the robot (Plate 1). The sessions were filmed and the verbal instructions were recorded digitally with a good quality headset microphone.

Plate 1 A subject instructing the robot during corpus collection. Inset: remote-brained miniature robot

Subjects were told that the recordings would be used later by a human operator driving the robot by remote control using only the images acquired from the on-board camera. This was aimed at generating robot-centric spatial references that would be appropriate for use by a computer program controlling the robot using only information from the on-board camera. By specifying that the user of the instructions would be a human, the need for users to second-guess what a robot might be able to understand was avoided. This eliminated any distortion of the subjects natural expressions.

The project also investigated how subjects would refer to routes previously instructed to the robot. This is because, in principle, once a given procedure has been explained to a robot, a user would normally re-use it and refer to it when explaining more complex procedures. Therefore, routes given to subjects were organized so that certain routes would be extensions of previous routes with a few more turns or intersections. Subjects were instructed that, when appropriate, they could save time by referring to previously explained routes, instead of re-explaining all of the steps. Twenty-four subjects were used, instructing six routes each. The recordings of these 144 instructions constitute the IBL corpus.

Corpus analysis

The analysis of the corpus had two purposes:

1 to customise the speech recognition and natural language processing system for the domain of the application; and
2 to define the functional primitives of the robot.

On the natural language side, the recorded instructions were first divided automatically into shorter sections by detecting naturally occurring silences. These silences tend to fall naturally between functional chunks. This method produced meaningful chunks in most of the cases. The sound files were then transcribed by hand into text files. The transcripts were used to automatically generate a restricted grammar comprising the subset of all the rules of a wide-coverage grammar hit by the corpus. This restricted grammar is then used to train a speech recognition system[1] for this route instructions domain (Bos, 2002; Bos *et al.*, 2003).

On the robot primitive side, the corpus transcripts were annotated by hand, as there is no off-the-shelf tool for doing this automatically. The annotation process starts with identifying actions classes in the user instructions and then deciding upon an annotation format (Table I).

This annotation of instructions is a somewhat subjective process guided by the knowledge that formal programs would have to be written for each action. Thus there was a bias towards grouping actions into a small number of "primitive" procedures. The consequence was that each primitive accepts a number of parameter combinations, e.g. "turn left, turn

Table I Example of functional annotation of a transcription

[take your first right]
→ TURN (first, right)
[continue down the street past Derry's past Safeway]
→ FOLLOW_ROAD_UNTIL (past, Derry's)
→ FOLLOW_ROAD_UNTIL (past, Safeway)
[the car park will be on your right]
→ DESTINATION_IS (car park, right)

right, take the second right, turn left after the church" are all grouped as one primitive "turn ($p1$, $p2$, $p3$, $p4$, ...)", where $p1$, $p2$,... are parameters such as direction, ordinal, etc. It should be noted that during the project, the specifications of primitives slightly changed as other constraints appeared, e.g. when details of robot behaviour or speech interpretation were considered. Another bias came from the need for a termination condition for each primitive. For instance, when subjects say "keep going", this is a non-terminated action that would set the robot into an infinite loop. In practice, such instructions can be ignored. The reason is that each of the other terminating functions has a "keep going" function already built into it. For instance, "turn left" functionally means "keep moving until you reach the left turn, and then take the turn".

The 13 primitives found in the corpus and details about their implementation are given in Kyriacou *et al.* (2005) (Table II).

Robot design constraints

The corpus analysis raises a number of points relevant to service robot design:

1 action primitives are determined by natural language;
2 primitives are complex functions;
3 primitives must be very robust;
4 error handling is an open problem;
5 human instructions contain many errors; and
6 spoken input affects the computational architecture of the controller.

- *Action primitives are determined by natural language.* For a roboticist, the standard action primitives that come to mind are actions such as "rotate left", "rotate right", "move forward" and "move backward". In contrast, users refer to functions such as "park near to ...", "turn right after...", "cross the road to...". Human language is economical with words. Often, few simple words are used to describe complex procedures. A service robot must be able to execute such procedures to "understand" natural-language instructions. Corpus collection and analysis is a useful method for acquiring such information.
- *Primitives are complex functions.* Take the example of a typical final command in route instructions. This is often a statement like "and you will see it there on your left". This command is highly under-specified and requires significant autonomy from the robot which must visually locate the destination and then plan a path towards it. In a real urban environment the final command would pose vision and control challenges that are at, or beyond, the limits of current technical capabilities.
- *Robustness.* Primitives must be very robust in the sense of coping with a variety of environmental variations. When a robot is sent out to "take the next left" its user needs to be confident that the robot will be able to find the turn and take it. It is a major challenge for a programmer to design such a robust function. There may also be effects from one primitive to the next in a sequence. Here again, the corpus is useful in

Table II List of primitives

Primitive procedure	Description
1 **follow_road** (relation_1, ordinal_1, object_1, relation_2, object_2)	Commands the robot to move forward on the road until a specified location
2 **turn** (ordinal_1, relation_1, object_1, relation_2, object_2)	Commands the robot to take a turn from the current road
3 **location** (object_1, relation_1, ordinal_1, object_2 = 'road', object_3, destination_1)	Specifies the location of an object. If the object is the destination the robot moves to it otherwise the robot stops as soon as it locates the object
4 **exit_roundabout** (ordinal_1, relation_1, object_1)	Commands the robot to take an exit off the roundabout. If the robot has not entered the roundabout then it follows the road until it meets the roundabout, enters it turning left (clockwise around the roundabout) and takes the designated exit
5 **Go** (relation_1, object_1)	Commands the robot to execute a previously explained route
6 **Go_until** (object_1, relation_1, object_2)	Commands the robot to use part of a previously explained route
7 **enter_roundabout** (direction_1, relation_1, object_1)	Commands the robot to enter the roundabout in a specific direction
8 **cross** (object_1, relation_1, object_2)	Commands the robot to cross the road to an object (usually the car park) ahead or to just cross to the opposite road at a crossroads, for example
9 **rotate** (relation_1, object_1)	Commands the robot to rotate about itself
10 **take_road** (relation_1, object_1)	Commands the robot to take a road in view. Usually used when the robot is at an intersection and needs to get on an opposite road
11 **exit_object** (object_1)	Commands the robot to exit from a place. Usually used for exiting the car park
12 **park** (relation_1, object_1)	Commands the robot to park either on/by a specific location
13 **bear** (relation_1, object_1)	Commands the robot to take one of the two directions at a *y*-junction

that it defines a number of situations on which to test sequences of primitives. In the IBL project, one half of the corpus was used for system development and one half was used to test its robustness (Kyriacou *et al.*, 2005).

- *Error handling.* Practically, one can only test developed primitives in a limited number of conditions and there will always be other situations for which the robot is unprepared. How to detect and handle such cases is still an open question. Similarly, the robust handling of speech recognition errors remains an important research area.
- *Human errors.* Human instructions contain many errors. In the IBL corpus, 29 per cent of route instructions contained errors that prevented the robot from reaching its destination. In contrast, when human listeners followed the same instructions, they failed in only 17 per cent of routes. This shows that human listeners can correct errors in instructions. It is unclear at present how they do it. It can only be assumed that to build such error-correction mechanisms in a robot would require significant artificial intelligence (AI) work. There are also safety and communication issues, as many errors are detected and corrected by human listeners while travelling the route. Generally, robotic assistant should not be able to decide in the middle of a task that something else should be done instead without prior approval from its user. This requires from the robot the ability to generate problem description to feedback to its user.
- *Computational architecture.* Almost all references to previous routes found in the corpus required only a partial use of the instruction sequence, e.g. "take the route to the station, but after the bridge turn left". One of the problems is that the bridge may not even be mentioned in the instruction of the route to the station. How to generate an execution program from such instruction? A possible solution is to implement a multi-threaded concurrent processing scheme where the robot would "follow the road to the station" and at the same time try to "take the left turn after the bridge". The latter process would remain the sole active as soon as the turn is found (Lauria *et al.*, 2002). It remains to be seen if this solution is general enough, but it is interesting to note that the way users express themselves could end up dictating the computational architecture of the robot controller.

It was also found that the route-instruction domain is not generic in the computation sense, as almost all instructions only comprised sequences. There were many cases of implicit decision and loops. For instance, a function such as "turn left" can be decomposed into "keep moving until a left turn is seen, then take the turn". However, there were no explicit references to IF-THEN-ELSE structures in the corpus. Therefore, instructions in another domain were investigated where more conditions were expected. This work is described below.

Game instruction

When humans give instructions on how to play a card game, these instructions are structured differently from other instructions such as how to follow a route. Game instructions such as card games or board games usually comprise, in the main, rules to apply with fewer sequences of action. They are also expected to be multi-modal processes where gestures such as pointing play an important part. To collect a corpus of game instructions, an environment was designed where an instructor uses verbal instructions and a touch-screen to point to a card or game piece (Plate 2).

An Italian card game was selected to ensure that the subjects of the experiment have no prior knowledge of the game. Each subject-learner became subject-instructor a few days later. The first instructors learnt the game from written instructions provided in random order, to avoid any initial bias in the structure of the instructions (Wolf and Bugmann, 2005). As in the IBL project, the instruction dialogues were filmed and recorded. The analysis of the corpus resulting from these experiments is presently underway.

Preliminary findings of the corpus analysis are:

- *Instruction type.* Instructions include action rules (e.g. "when it is your turn, put down a card with the same value"), informative statements (e.g. "the king is worth 10 points") or imaginary situations (e.g. "suppose this card is a 4, then you can take..."). Such instructions do not lend themselves easily to the construction of classical imperative programs. Declarative programming techniques may be need to be considered here, as they are well adapted to deal with knowledge represented as a set of rules. How to deal with the sequences found in route instructions and other tasks requires further analysis.
- *Functional primitives.* In the IBL project, the functional primitives were the action commands found in instructions. Here we find physical actions such as "deal cards", "capture cards" or "turn over cards". However, some instructions contain no physical action commands. Instead, they invoke knowledge-management functions (e.g. "associate a value with a card", "create a rule", "start an imaginary situation"). These knowledge-management primitives (or "mental actions") will require appropriate designs of knowledge representation methods.

Plate 2 Setup for collecting a corpus of card game instructions.

Note: The instructor on the right demonstrates a move on his touch screen. The learner sees a copy of the move on her screen. The separation panel between instructor and learner forces the gesture component of the communication to take place via the screen and can easily be recorded

- *Implicit functionality.* The structure of game instructions reflects the teacher's assumption that the student has capabilities for rational thinking and will be able to use acquired knowledge for generating a purposeful sequence of game moves. For instance, in the corpus, the instructor often reinforces a rule statement with examples. This assumes some generalization powers from the learner. The designer of a learning robot will need to build such capabilities in the robot. It is unclear at present how much details on these capabilities can be inferred from the corpus and if these will be consistent with what we know about human rationality.

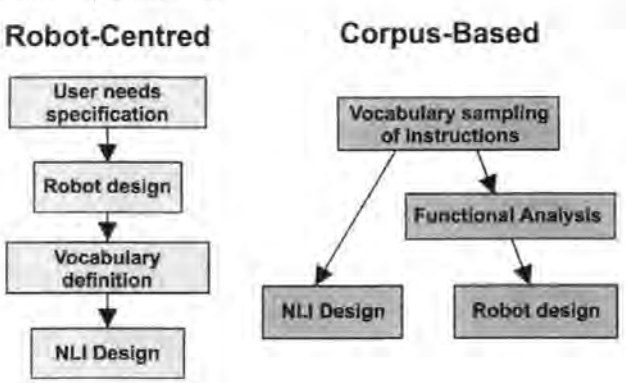
Corpus-based robot design

In the area of computer software development, it is a recognized practice to specify the user interface early in the design process and then to design the software around the interface. In robotics, this is a new concept, as spoken interfaces were traditionally seen as the last component to be added to a robot.

In the traditional approach to robot human-robot interface design, termed "robot-centred" in Figure 1, the design team starts with as specification of the user's needs, then designs a robot as functional as possible, then defines the vocabulary to access the functionality. Finally, a NLI is built to deal with that vocabulary. The traditional approach requires the user to learn the specific language and keywords prepared by the robot designer, i.e. a "constrained language". However, if one expects the robot to understand unconstrained spoken language, then the interface question needs to be considered prior to robot design. This is because of the functional implication of spoken interfaces illustrated above.

In the proposed "corpus-based" approach, the design process starts with sampling sentences representative of how users intend to address the robot (Bugmann *et al.*, 2004). Once the words that the user likes to use are known, the design of a NLI that deals with them can start. At the same time, analysing user utterances also informs on the functions that users like to refer to. This then specifies the functionality to be built into the robot. The end product is a system well tuned to the user's language and needs.

Figure 1 Robot-centred vs corpus-based robot design. In the robot-centred approach, the functionality is defined first, then the access vocabulary, then the NLI. In the corpus-centred approach, the content of samples of dialogues between humans defines at the same time the vocabulary to be dealt with by the speech interface and the required functionality of the robot



In general, analysing samples of dialogues between humans representative of future human-robot interactions (such as instructions dialogues in the projects described above), provides key information to the designer of a service robot. This was illustrated above mainly on the software side, however, there are also hardware implications. Let us assume that a user of a domestic robot-cook needs to give an instruction involving the expression "a pinch of salt". This will clearly exert constraints on how the robot's manipulators are to be designed. Similarly, if a mobile robot needs to understand the command "turn right at the blue sign", it will need to be provided with colour vision.

Whether this method can be of use in industrial robotics needs exploring. For instance, is there a saving by designing interfaces requiring less training from the operators? Are there aspects of scheduling and planning which could benefit? While this is an open question, there is no doubt that corpus-based design has a clear place in service robotics.

Conclusion

A number of experiments have demonstrated the close relationship between the way humans speak and think and the impact this has on service robot design. Therefore, it is suggested that an analysis of HRC should be the first step in the design process. This is encapsulated in the proposed corpus-based design method.

Overall, speech interfaces require a high level of functional competence from the robot, as humans commonly refer to high-level functions in their everyday language. It, therefore, follows that, in order to understand human language, robots need to mimic some functional capabilities of human listeners. This includes aspects of rational thinking and robust perception. The corpus-based approach, described here, provides a method to specify such functional competences.

Note

1 The system was built around the Nuance 7.0 toolkit (www.nuance.com) running on a network of Unix workstations and Linux PCs linked through the open agent architecture (www.ai.sri.com/ ~ oaa/).

References

Bos, J. (2002), "Compilation of unification grammars with compositional semantics to speech recognition packages. COLING 2002", *Proceedings of the 19th International Conference on Computational Linguistics*, pp. 106-12.

Bos, J., Klein, E., Lemon, O. and Oka, T. (2003), "DIPPER: description and formalisation of an information-state update dialogue system architecture", paper presented at Fourth SIGdial Workshop on Discourse and Dialogue, Sapporo.

Bugmann, G., Klein, E., Lauria, S. and Kyriacou, T. (2004), "Corpus-based robotics: a route instruction example", *Proceedings of IAS-8*, 10-13 March, pp. 96-103, Amsterdam.

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.

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

Kyriacou, T., Bugmann, G. and Lauria, S. (2005), "Vision-based urban navigation procedures for verbally instructed robots", *Robots and Autonomous Systems*, Vol. 51, pp. 69-80.

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 International Workshop on Robot and Human Interactive Communication (Roman'02)*, pp. 223-8, Berlin.

Matsui, T., Asoh, H., Fry, J., Motomura, Y., Asano, F., Kurita, T., Hara, I. and Otsu, N. (1999), "Integrated natural spoken dialogue system of Jijo-2 mobile robot for office services", *Proceedings of AAAI/IAAI*, pp. 621-7.

Torrance, M.C. (1994), "Natural communication with robots", MSc thesis, MIT Department of Electrical Engineering and Computer Science.

Wolf, J.C. and Bugmann, G. (2005), "Multimodal corpus collection for the design of user programmable robots", *Proceedings of the Taros'05*, London.

Multimodal Corpus Collection for the Design of User-Programmable Robots

Joerg C. Wolf **Guido Bugmann**
Robotic Intelligence Laboratory
School of Computing Communications and Electronics
University of Plymouth
Drake Circus
Plymouth PL4 8AA, U.K.
{joerg.wolf, guido.bugmann}@plymouth.ac.uk

Abstract

In order to design a robot able to learn from its user, the example of card game instruction is investigated. A detailed description of a setup for the collection of a human-to-human multimodal instruction corpus is given. This setup uses touch screens and will also provide a base for the human-robot instruction interface to be designed. Preliminary results on learning dialogues are given and issues of corpus transcription, annotation and analysis are discussed.

1 Introduction

In the future, intelligent robots may become part of daily life. Robots are already entering our environment as interactive toys. Robots will manage the household or an office environment as autonomous agents. Future service robots can not be completely pre-programmed by the manufacturer. There are far too many possible tasks. In order for these robots to successfully learn and interact with people from the general public, they must be *programmable by anybody* (naive users / without training) and not just by engineers, roboticists and computer scientists. The user does not even have to be IT-literate. The design of a "user programmable" system is the subject of this research.

A user-programmable robot must use an interface that is natural to the user. The best way to design a truly easy-to-use interface is by examining the interaction between people. By observing instructions from human teachers to human students, guidance is sought here for the design of a robot acting as the student.

Previous work carried out by our laboratory on the Instruction Based Learning project (IBL) (Kyriacou, 2004; Bugmann *et. al.* 2004) has shown that it is possible to extract information from a representative sample of the teacher's utterances (the "corpus") in order to:

- Identify primitive procedures that the robot has to be able to carry out (the robot's "prior knowledge")
- Write and tune a speech-recognition software to call and combine these primitive procedures.

This approach to the definition of the robot's functionality and natural-language interface (NLI) has been described as "corpus-based robotics" (Bugmann *et. al.* 2004) and is outlined in figure 1.

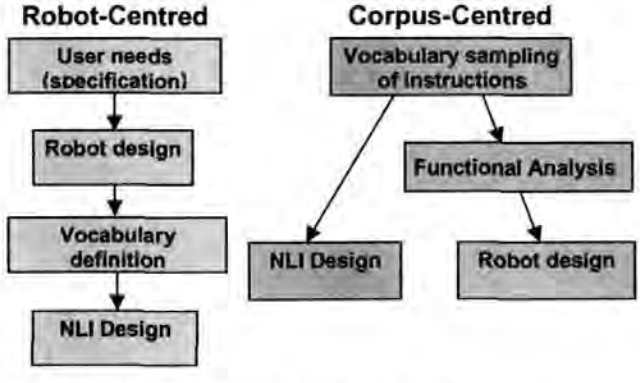


Figure 1: Robot vs. Corpus-Centred Natural Language Interface (NLI) design. In the Corpus-centred approach, the content of samples of instructions between humans defines at the same time the vocabulary to be dealt with by the speech interface and the required functionality of the robot. In the robot-centred approach, the functionality is defined first, then the access vocabulary, then the NLI.

The IBL project focused on route instructions given to robots. A dialogue such as the following was possible between the user and the robot:

User: "Go to the University."
Robot: "How do I go there?"
User: "Take the third turning to the left..."
Robot: "Next instruction please."
User: "...take the third exit off the roundabout..."
Robot: "Next instruction please."
User: "The University will be on our right."
Robot: "OK, it's done."

Since the IBL project was using route instructions, the resulting system was developed to deal with sequential instructions, and could not handle other forms of instructions, such as general rules, which apply at any time during the task, such as "Stop at the petrol station if you run low on petrol". The system could not deal with conditionals, such as the one above, that were not found explicitly in the corpus (Lauria *et al.*, 2002). In route instructions, sentences starting with "if" instructions are generally just a colloquial way of expressing a sequential instruction, as in the following example from the IBL corpus: "...okay if you carry on straight along this road and if you take the third left you will go over a bridge..."

Therefore, to develop a more general instruction system, there is a need for looking at a different application, where instructions not only include sequences, but also other instruction structures. In imperative programs these would be decisions and repetitions. However, in the declarative paradigm, programs consist of lists of goals and a set of rules (see e.g. PROLOG). It is unclear which paradigm is a more useful representation of human instructions. This is one of the questions that need to be addressed by analysing a new corpus of instructions in a different domain (see section 2).

Furthermore, previous research in our group focused purely on verbal instructions which are well suited to communicate rules and sequences of actions, but are less well suited for other aspects of instructions. In practice, many tasks are explained using a mixture of verbal instructions, gestures and demonstrations. Thus, a truly natural interface between human and robots must be multimodal. This is one of the features to include in this project.

In the following section, we describe the selected instruction domain and the particular setup designed to facilitate the capture the human subject's multimodal behaviour and their reproduction by a robot.
In section 3 we describe the corpus collection protocol and in section 4 (very) preliminary results are given[1] along with consideration on its semantic annotation. Section 5 is the conclusion.

[1] *The corpus is currently being transcribed and annotated*

2 Experiment Design

2.1 Instruction domain

The criteria for the selection of a new application/task are as follows. i) The task should contain a wide range of instruction types. ii) The task should be scalable from simple to complex. iii) The vocabulary should be restricted to a domain.

Given these constraints, game instruction seems to be a good choice. In particular, card games come in a great variety of type and complexity, yet their vocabulary is restricted.

We investigated all two player games listed in "the Oxford A-Z of Card Games" (Parlett, 2004), and defined a simple measure of complexity as being the sum of the number of rules stated in Parlett's game descriptions. The more rules a game had the more complex it was assumed to be. We selected the national Italian card game "Scopa", since it has intermediate complexity that is typical for card games and is not commonly known in the U.K. Scopa is a fishing-type card game. A fishing game means that there are several cards face-up on the table, and the players have to match cards in their hand with the cards on the table. Matching cards on the table can be captured by the player in order to score.

2.2 Multimodal interface

In future robots, multimodal interfaces will require complex sensory processing, such as gesture and face recognition. As this project focuses on the problem of learning, we decided to devise a simplified interface that would still allow natural communication with human users.

Our solution to the problem is to use a touch screen that allows at the same time to acquire human gesture information by the robot (without complex sensory processing) and execution of game moves (without complex actuators). The screen represents the world as the robot would see it through it's vision system. The user is able to point at and manipulate objects on the screen as a demonstration of how to do the task. At the same time the user gives verbal instructions. Touch-screens have been used in multimodal human-robot interfaces for different applications, for example by (Perzanowski *et al.*, 2001), or for investigations in human communication (De Ruiter *et al.*, 2003)

A great advantage of using a screen representing the robot's world is that the robot can be simulated, while the interaction and interface to the robot does not change. It also allows focusing research on human-robot interfaces without having to build a robot.

The software used to display the playing cards is based on the Qt (Trolltech®[2] 2005) and the OpenGL®

[2] *Qt is a trademark of Trolltech in Norway and other countries. http://www.trolltech.com/*

API[3] and is platform independent. Qt is a cross-platform C++ GUI development library. OpenGL® is a standard for a 3D/2D cross-platform Graphics API. The playing cards are described as objects with parameters such as size, texture, position, orientation, static or movable. Therefore the system can be used not only for card games. The display software could display any real world object that the robot knows about. The user can manipulate these objects intuitively. The computers used for displaying the cards are linked to a server via TCP/IP. All events of object manipulations are logged at the server and forwarded to all other connected clients. So if objects are moved on one screen, they move on the other screens as well.

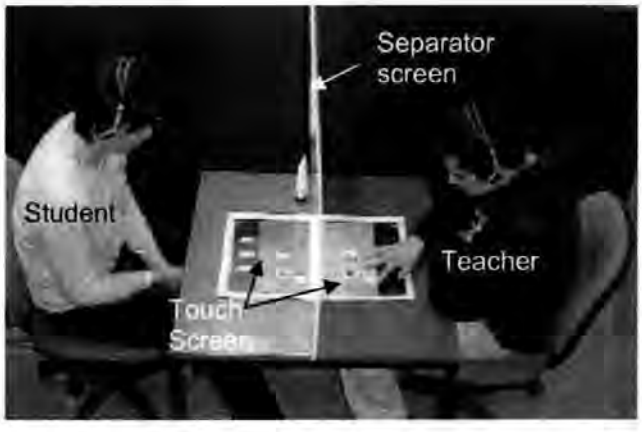


Figure 2: Setup for corpus collection

2.3 *Corpus collection*

A teacher and a student sit at a desk (Figure 2). The two are separated by a screen so they can not see each other. The desk has touch screens build into its surface. Playing cards are shown on the screens. The cards can be moved around on the screen by touching and dragging them around. Both players have a common area for cards, and an area that can only be seen by one player (black area on the touch screen in figure 2) symbolizing the cards in the hand. The common area is located near the screen and represents the virtual playing-table. The teacher will explain a card game to the student. The interaction is filmed and the dialogue recorded. To ensure high quality recordings, the subjects wear headset microphones. The coordinates and movements of the cards on the touch screens are recorded simultaneously. The data can be synchronized with a time stamp. This simultaneous recordings of voice and touch screen data from object manipulations constitutes a multi-modal corpus.

[3] *OpenGL® and the oval logo are trademarks or registered trademarks of Silicon Graphics, Inc. in the United States and/or other countries worldwide.*

We recorded 21 dialogues between teachers and students. Students who learned the game in one session became the teachers in the next. In the design of the protocol we tried to avoid two forms of bias, the vocabulary bias and the instruction strategy bias.

Pilot studies revealed that a teacher subject tends to use expressions and methods similar to those used when he/she was taught. To avoid this bias, we decided that the initial teacher (Student S0 at the top of the tree) would learn the rules of the game from a set of rules written on separate sheets and presented in random order. Subjects usually proceed by re-ordering the sheets to help learning the game. Two sets were used with different words for the same rules.

In order to maintain the chain if a subject decided to drop out, the experiment was designed in a tree structure where one teacher teaches two students, and then these students become teachers themselves. Figure 3 shows one of two trees used in this experiment. Example: Teacher S0 teaches student S1 and S8. After that S1 becomes a teacher and teaches S2 and S3.

We left at least one day between learning the game and having to teach it. This generally led to a fading of the memory of the precise words and order of instructions. Thus the chain design is expected to reduce the bias in vocabulary and lead to an increased variety of instruction styles in the corpus.

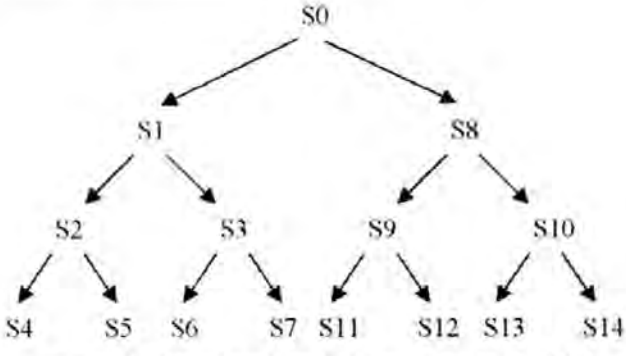


Figure 3: Tree of teaching dialogues. Two trees of this type were used to record dialogues. There are 14 dialogues in each tree, represented by the arrows and organized in three layers. S*i* is the subject number *i*.

3 Preliminary results

As a general observation, the use of a separation screen between teacher and student had the expected effect, in that gesture communication was very much restricted to the touch screen. Very few gestures "in the air" were observed.

The next observation is that the length of the dialogue for explaining the same card game decreased along the chain of dialogs (see figure 4). It appears that the way the game was explained became more efficient in lower layers of the tree.

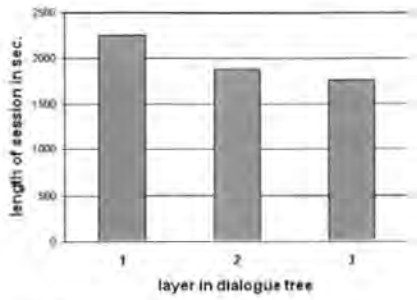


Figure 4. Dialogue length vs. layer

A typical conversation extracted from the corpus (ts_session19 03:56).

...

Teacher: "There is an orderly deck of cards...All the numbers are their usual value. Other than the fact than Jack is 8, Queen is 9 and King is 10...and ace is 1."

Student: "right"

Teacher: "So you need to remember that, obviously for when you are pairing or capturing cards."

Student: "So Jack was 8 you said."

Teacher: "Yeah"

Student: "And ehm, Queen was 9."

Teacher: "Yeah"

Student: "And King is 10."

Teacher: "Yeah. And Ace is low, number 1."

...

A first look at some transcripts suggests the presence of at least two types of primitive functions in the instructions:

- Knowledge management functions ("A king is worth 10")
- Action functions ("Put down this card")

These kind of functions will need to be implemented in the learning robot. Finally, we also noted the occurrence of:

- contradicting statements
- underspecified statements
- mixed up order of instructions

These features are likely to represent challenges for the design of dialogue and knowledge management components of the robot-student.

4 Discussion

4.1 Multimodal transcriptions

Speech transcription can done using tools such as Transcriber[3]. This produces a time stamped XML text corresponding to a recorded sound file. We are currently investigating if there are similar tools for the transcription and annotation of signs or gestures done in a card game on a touch screen. Otherwise, dedicated software will have to be developed, possibly inspired by (Bird and Liberman, 1998). The task of such software is to annotate the raw recording of trajectories on the screen with high-level "sign tags", such as *pointat(AceClubs) or turnover(AceHearts)*.

The multimodal recordings and transcriptions in these experiments are linked with time-stamps. This requires the recordings to be started simultaneously. Similarly (Knut Kvale *et al.* 2004) uses timestamps to synchronize inputs from touch screen and voice. Transcriptions are commonly done in XML. However, there is no widely accepted standard for multimodal transcriptions. We are in the process of reviewing transcription and annotation tools.

4.2 Semantic Annotation

The purpose of annotation of this multimodal data is to provide reference data for testing the system to be developed and identifying the semantics in a formal format. Two streams of data are coming into the robot: utterances and touch screen inputs.

The touch screen inputs are in the form of trajectories of card movements and must be converted into a "symbolic" format, such as *moveto(KingHearts,table)*, referred to as signs. Both, voice and signs have timestamp and duration.

Synchronizing the two allows resolving deictic references. Figure 5 shows an example, where the teacher says "this one" and starts pointing at a card. The gesture *F* means that the card was touched by the user. The gesture *M* stand for subsequent movements (wiggling) of the card, which would be recognized by the student.

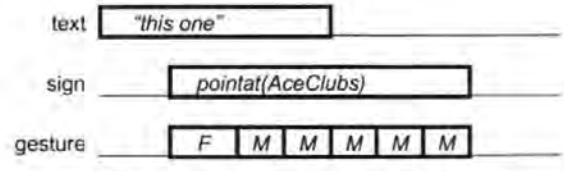


Figure 5: Timing diagram of multimodal inputs

One point worth considering is that the annotation scheme is tightly coupled with the system's concept of operation. For instance, one could decide that any utterance by the teacher requires an action by the robot. Based on the primitives noted in section 3, these actions would then be knowledge manipulation actions or actual actions on the cards in the game.

[3] *http://www.etca.fr/CTA/gip/Projets/Transcriber/*

5 Conclusions

In order to design a robot able to learn from its user, the example of game instruction is investigated. A detailed description on a setup for human-to-human multimodal instruction corpus collection is given. The use of touch screens as the "game table" simplifies the recording of gestures and will greatly simplify the design of the learning robot, which will essentially be a software agent.

The transcription and analysis of the corpus has been discussed and is likely to require new tools, especially in the area of signs definition and transcription. The semantic annotation scheme needs to be defined carefully, as it links to system design. In addition, it is possible that the designed agent will have to reproduce some of the features of the human thought processes engaged during learning and playing, to give the robot the ability to learn from natural instructions. Information on these issues is expected to be produced by the analysis of our corpus of human teacher-student dialogues.

References

Bird, S. and Liberman, M., (1998), Towards a formal framework for linguistic annotations. *Presented at the ICSLP, Sydney*

Bugmann,G., Klein, E., Lauria, S., Bos, J. and Kyriacou T., (2004) "Corpus-Based Robotics: A Route Instruction Example" *in Proceedings of IAS-8*, 10-13 March 2004, Amsterdam, pp. 96-103.

De Ruiter, J. ,P. , Rossignol, S., Vuurpijl, L., Cunningham D.W. and Levelt, W.J.M., (2003). SLOT:A research platform for investigating multimodal communication. *In Proc. Of Behavior Research Methods, Instruments & Computers 2003*, 35(3),408-419

Miura,J. , Yano,Y. , Iwase,K. and Shirai, Y., (2004), Task Model-Based Interactive Teaching, *In Proc .of IROS 2004 Workshop on Issues and Approaches to Task Level Control*, Sep. 28, Sendai, Japan, 2004.

Kvale,K., Knudsen, H., E. and Rugelbak, J., (2004) "A Multimodal Corpus Collection System for Mobile Applications", *LREC2004 Workshop, Lisboa, Portugal*, 24.5 - 30.5.2004.

Kyriacou T., (2004), Vision-Based Urban Navigation Procedures for verbally instructed robots. PhD Thesis, University of Plymouth, U.K.

Lauria, S., Kyriacou, T., Bugmann, G., Bos, J. and Klein, E. (2002). Converting Natural Language Route Instructions into Robot Executable Procedures. In *Proc. of the 2002 IEEE International Workshop on Robot and Human Interactive Communication* (Roman'02), Berlin, Germany, pp. 223-228.

Leech, J. and Brown, P.,(2004), The OpenGL® Graphics System: A Specification, Version 2, Silicon Graphics Inc.

Parlett, D., (2004), "the Oxford A-Z of Card Games", Oxford University Press, Second Ed.

Perzanowski, D., Schultz, A., C., Adams W., Marsh, E. and Bugajska, M., (2001), "Building a Multimodal Human-Robot Interface", *IEEE Intelligent Systems, 16 (1), IEEE Computer Society*, 16-21.

Timing of visual and spoken input in robot instructions.

Joerg Wolf and Guido Bugmann
Robotic Intelligence Laboratory, School of Computing, Communications and Electronics,
University of Plymouth, Drake Circus, Plymouth PL4 8AA, United Kingdom.
joerg.wolf@plymouth.ac.uk, gbugmann@plymouth.ac.uk

Abstract:
Trainable robots will need to understand instructions by humans who combine speech and gesture. This paper reports on the analysis of speech and gesture events in a corpus of human-to-human instructions of the dealing phase of a card game. Such instructions constitute an almost uninterrupted stream of words and gestures. One the task of a multimodal robot interface is to determine which gesture is to be paired with which utterance. The analysis of timing of events in the corpus shows that gestures can start at various time relatively to the speech, from 5 seconds before speech starts to 4 seconds after speech ends. The end of a gesture never precedes the corresponding utterance. A simple algorithm based on temporal proximity allows to pair correctly 83% of gestures with their corresponding utterances. This indicates that timing carries significant information for pairing. For practical applications, however, more reliable pairing algorithms are needed. The paper also describes how individual actions can be grouped into a gesture and discusses the integration of semantic information from gesture and speech.

Keywords: human-computer interaction, natural language understanding, multimodal interfaces, service robots, speech events timing.

1. Introduction

Future service robots can not be completely pre-programmed by the manufacturer. There are far too many possible tasks and user-dependent variants. These robots will need to learn interactively from their users. They will need to be *programmable by anybody* (naive users / without training) and not just by engineers, roboticists and computer scientists. A user-programmable robot thus requires an interface that is natural to the user. One approach to the design of a truly easy-to-use interface is by examining the interaction between people. By observing instructions from human teachers to human students, guidance is sought here for the design of a robot acting as the student. In a previous project in our laboratory on Instruction Based Learning project (IBL) (Kyriacou, 2004; Bugmann *et. al.* 2004) it was shown that through the analysis of the teacher's utterances, it is possible to:

- Identify primitive procedures that the robot has to be able to carry out (the robot's "prior knowledge")
- Write and tune speech-recognition software to address and combine these primitive procedures.

This approach to the definition of the robot's functionality and natural-language interface (NLI) has been described as "corpus-based robotics" (Bugmann et. al. 2004, Bugmann et al., 2005).

In the current "multimodal IBL" project (MIBL), the analysis of human-to-human multimodal instructions combining gesture and voice is explored. The test case is that of a human explaining to another human how to play a specific card game. The teacher and students communicate using voice and card manipulations on touch screens. Such actions could theoretically also be detected with a vision system and most of the results presented here are hopefully valid for multi-modal systems using vision-based gesture recognition. Previous reports on multimodal input systems using touch screens, such as (e.g. Boves et al., 2004) impose strict constraints on when inputs can be provided and note that "subjects hardly ever combined pen and speech". In the free-flowing instruction application described here, there is frequent combination of speech and gesture and there

is a need for assessing which gesture corresponds to which speech act. Other works, e.g. (Perzanowski et al., 1998) and subsequent works e.g. (Perzanowski et al., 2003), do not provide details on how elements of a stream of words and a stream of gestures are associated.

In section 2, the experimental setup and corpus collection procedure are summarized. In section 3, the methods of gesture recognition and categorization are described. In section 4, timing data are shown that suggest a method for synchronizing visual input (gestures) and auditory input (speech recognition). In section 5, the semantic information provided by gesture is discussed. Section 6 offers concluding comments.

2. Experimental setup and corpus.

The MIBL project is focused on the generation of programs for the robot from multimodal instructions. To reduce to a minimum the problems of visual perception and manipulation of real-world objects, it was decided to use a touch screen as interface between the human and the robot, in addition to a voice interface. The screen represents the world as the robot would see it through it's vision system. The user is able to point to and manipulate objects on the screen as a demonstration on how to do the task. At the same time the user gives verbal instructions. Touch-screens have been used in multimodal human-robot interfaces for different applications, for example by (Perzanowski *et al.*, 2001), or for investigations in human communication (De Ruiter *et al.*, 2003)

A great advantage of using a screen representing the robot's world is that the learning robot to be designed can be simulated, while the interaction and interface to the robot does not change from the one used to collect human data. It also allows focusing research on human-robot interfaces without the need of having to build a robot. Details on how subjects were organized into teachers and students, and the experimental protocol can be found in (Wolf and Bugmann, 2005). In short, subjects where initially students and became teachers in later sessions. 21 teaching sessions were recorded. Two of them concerned the training of the initial teachers and are not used for system development. Ten sessions are used for system development (training set) and nine sessions will be used for system testing (test set). The corpus comprises video recordings of the sessions (see figure 1), recordings of the voices of the teacher and the student, and recordings of movement of cards on the screen. For each teacher-student pair, the instructions session and two following games were recorded. This project focuses on analysing the teacher's speech and gestures, in order to build a robot able to understand human instructions.

Figure 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 screen. The separation panel between instructor and learner force the gesture component of the communication to take place via the touch screen and can easily be recorded. Each screen has a small "private" black band representing the hand of the player. The larger green area represents the table and is shared by the two players.

A transcription tool (Wolf and Bugmann, 2005) was designed that allows producing XML files including gesture and speech act timings. The entries on gestures were generated automatically using a recognition method described in the next section. The transcription of speech was done manually by adding speech tags to the gesture tags. The analysis of the corpus is not complete yet. This paper reports on data covering only the initial phase of the game instruction: how to deal cards.

3. Gesture recognition

In card games, gestures can be pointing gestures, gestures moving cards from one place to another (e.g. stack to table, hand to table), re-arranging gestures (making a group of cards look tidier) and turning over gestures. A touch screen operates as an additional mouse to a computer. The effect of a user touching the screen is signalled as a mouse button-down event. Moving cards on the touch screen is intuitively done by touching the card and dragging it to another position. The resulting data is a trail of X, Y coordinates of where the card is going. In case of a real service robot, this tracking data of cards on the screen could be the output the service robot's vision system.
The "analogue" trail of X, Y data of a cards position is then registered as a movement from a start area to a destination areas e.g. move(pile, temp1). The areas numbers and their boundaries are defined from observations of where the movements of the players usually end, namely: stockpile, table, hand1, hand2, temp1 and temp2 (figure 2). The stockpile's position is set by the system.

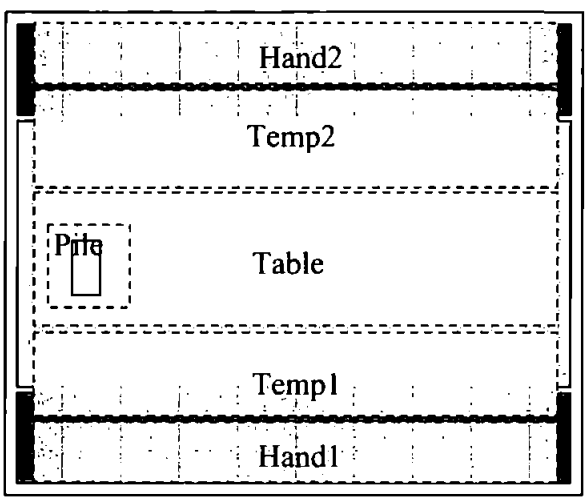


Figure 2. Areas defined on the touch screens. Temp1 and Hand1 are on the teacher's side. Temp2 and Hand2 are on the student's side. The teacher and the student can only see and manipulate cards in their hand area.

The categorization of areas is a straight forward comparison between coordinates and area boundaries. An analogue belief system, see Roy, D. (2005), could be used instead in a real service robot, if the vision system output coordinates are noisy or if there are no clear cut boundaries. In our experiment it was found that a statistical categorization is not required, in other cards games however, the situation can be more complicated. Within a move of a single card, users sometimes stop and then continue to move the same object until it reaches its final location. This is meant to be a single move by the user, but how can the robot recognize that? The strategy used here is to wait until the human picks up another object, which automatically implies that he has finished with the previous one. This is generally true with a touch screen, where there is only a single mouse cursor. With real vision, a better method might be to use a timeout.

Gestures which the same start and destination position are pointing gestures. Gestures with the same start and destination area are re-arranging gestures. Gestures with different start and

destination areas are card displacements. Gestures are recorded in a transcription file in XML format including time of start and end of movement, player doing the move, card identity and start position and destination, such as for instance:

<objmove t="2416" user="t" ID="D/5" from="+Table+Stock" to="+Temp2" until="2442"></objmove>

4. Synchronizing visual and auditory input.

In the transcriptions of the human-to-human dialogues, utterances and gestures are grouped together by the operator doing the transcription. This provides reference data, in order to design a system that can automatically group utterances with gestures. The challenge here is to pair automatically the correct gestures with the correct utterances. In the domain of card games we found that most often a single utterance U_1 was associated with several actions $G_1, G_2, G_3, \ldots$ forming a group of actions. Groups of actions, such as dealing 3 cards, are characterized by short time delay separating individual actions. Figure 3A shows that most actions in a group are separated by less than 2 seconds. However, the time intervals between groups are very short too, and time intervals between groups (end to start) are often smaller than 2 seconds (Figure 3B). Therefore, groups can not reliably be identified on the basis of time intervals. A safer and reliable method is to group actions according to their start and destination areas and to the type of action performed on the cards (see e.g. table 1). The end of a group of actions is identified either from the start of a new group of actions, or from a time-out of 2 seconds (from figure 3A).

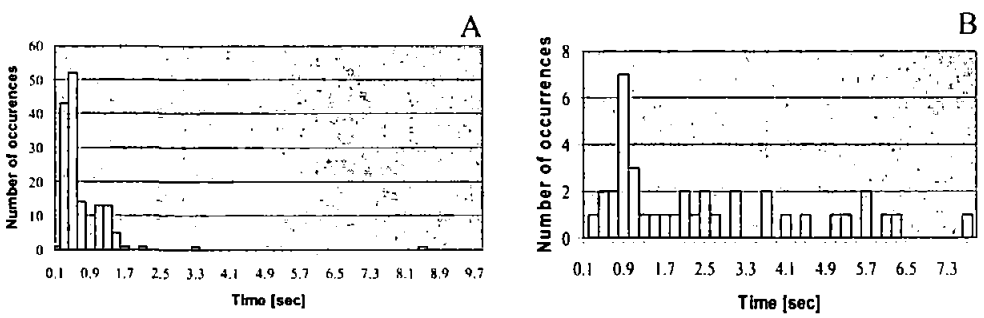


Figure 3. A) Histogram of time intervals between individual actions in a group of actions. B) Histogram of time intervals between action groups (gestures).

The second question is how groups of actions (which we will call a "gesture") can be associated with the corresponding utterances. The speech recognition engine (NUANCE 8.5) provides the times of start of speech and the times of end of speech. The same information is provided for gestures using the method cited above. Thus, we explored the possibility to associate gestures and utterances by comparing their relative timings. Gestures tend to start either before or after speech in equal proportion (Fig. 4A). However, gesture rarely starts after speech ends (Fig. 4B). Figures 4A and 4B show that the first action in a group usually occurs later than 5.5 seconds before the start of speech and not later than 4 seconds after the end of speech. This time "buffer" at each end of the utterance (see figure 5) can unfortunately not often be used to assign a start of gesture to its utterance. The reason is that the time interval between utterances (end of the previous to the start of the next) is generally smaller than 9.5 second (4 + 5.5) (Fig. 4C) and buffers belonging to different utterances generally overlap. Observation of the instruction process shows that the teacher never pauses for a long time, producing a nearly continuous flow of words and actions. This is especially true of instructions of the dealing phase. Thus, the time periods where buffers do not overlap (e.g. period B in figure 5) are relatively short. Only 40% of gestures start during this non-overlapping period and can be unambiguously paired. The start times of the remainder of the gestures fall into area A (figure 5) where pairing is uncertain. More elaborate rules of pairing are required. For

instance, additional information may be obtained from the timing of the end of the gesture. One observation might be of interest. Figure 4D shows that the end of a gesture always occurs at least 1.6 second after start of speech. In other words, subjects sometime start the gesture well before speech, but always start speaking before the paired gesture is completed. Unfortunately the reverse is not true. In several cases, subjects started a new utterance before the gesture related to the previous one had ended. Therefore, the timing of end of gestures does not immediately appear to be helpful to decide the pairing between gesture and utterance. To assess if relative timing carries useful information at all, we attempted to assign gestures starting in period A (figure 5) to the nearest utterance. This resulted in a total of 83% correct pairings (including the 40% correctly paired in time period B). This shows that timing contains significant information exploitable for pairing. However this result is of little practical use, as pairing errors are bound to cause serious problems in instruction understanding. What is needed is a pairing system that either produces a safe pairing or signals its uncertainty.

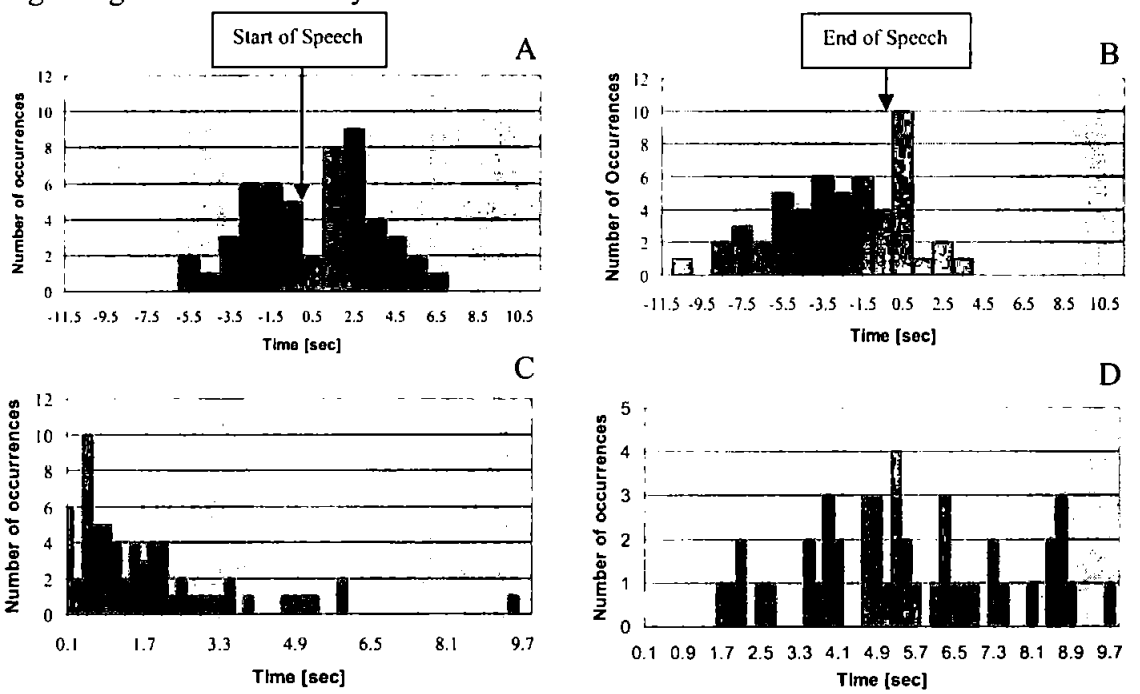


Figure 4: A) Histogram of the time intervals between start of speech and start of the first action in a group. B) Histogram if time intervals between the end of speech and the start of the first action in a group of actions. Only groups of actions associated with the speech event are plotted. C) Histogram of time-intervals between speech events (end of previous to start of next). D) Histogram of time difference between end of the last action in a group and the start of speech.

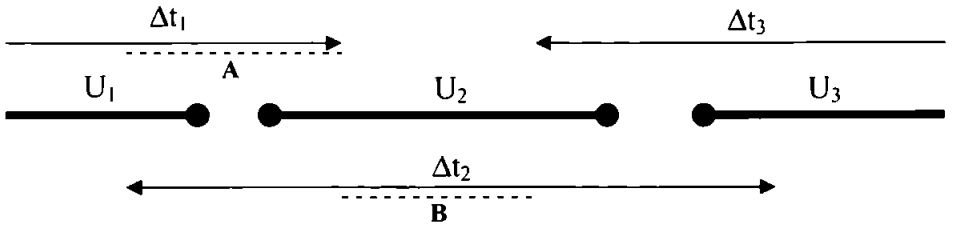


Figure 5: Illustration of overlapping time windows. Three utterances U1, U2 and U3 define each a time widow Dt1, Dt2 and Dt3 which extend the utterance duration by 5 sec before its start and 4 sec after its end. During the time span A, overlapping time windows make it impossible to assign gesture to either U1 or U2. Only during time span B can a gesture reliably be assigned to U2, based on the start time of the gesture.

5. Semantic representation and role of gesture

Card games typically consist of three phases: dealing, playing and a post-game phase where players count their points. In this paper we focus on the explanations covering the dealing phase. Table 1 shows an example of how verbal instructions and gesture are combined in such explanations. Dealing explanations comprise a sequence of actions, in contrasts to play instructions which include mainly rules (not shown in this paper). In the card game Scopa, which was used during corpus collection, every player gets three cards and another four cards are dealt face up on to the table. All subjects described dealing as a sequence of six steps (see e.g. table 1). In the MIBL corpus it was found that all teaching proceeds via spoken instructions accompanied by simultaneous execution, making descriptions of actions much more detailed, and easier to understand for a human or robotic student. The role of gesture is often to specify spatial coordinates of verbal instructions (e.g. instruction 2) or to resolve references such as "these" (e.g. instruction 4), as also noted by other authors (e.g. Perzanowski et al., 2000).

Do as I do. A curious problem with task instructions in unconstrained spoken language is that the speaker uses interchangeably "I", "you" and "we". This appear to come from the fact that the teacher is demonstrating and expects the robot to copy his/her behaviour in most cases. To some extent, the availability of an example to follow appears to require less linguistic rigour from the teacher. It is likely that instructions given over the phone would be much more precise. A formal linguistic analysis is bound to meet serious problems here, but this is not the topic of this paper. It appears that, at least in the dealing case, it could be a good strategy to ignore most of the speech and learn to mirror the teacher's actions. Only is cases where cards are invisible to the student (in area hand1) would speech processing provide necessary information. Practical implementation will verify if this is possible. Indeed, in explanation of the rules of the game speech cannot be ignored. Further analysis of the corpus will clarify this.

6 Concluding comments

The presented data show that gestures can start before, during or after an utterance within limited time windows, but never end before the utterance starts. Simple pairing rules based on parts of these data produce correct pairings for 83% of gestures. This is encouraging given the complexity of the situation analyzed here, characterized by a free flow of gesture and verbal instructions. For practical applications however, different characteristics are demanded from a pairing algorithm. It must either given a correct pairing, or signal it inability to provide a pairing. In the latter case appropriate repair dialogues can then be initiated. Another issue to be considered is the fact that dialogues with robotic systems are likely to exhibit different timing characteristics than the ones between humans. These may show a simplification of the pairing problem. Otherwise, and depending on the causes of the difficulty, one may have to introduce timing constraints on the sequence of speech/gesture through dialogue strategies. Another avenue is to exploit the semantic analysis of the speech to identify paired gesture.

Once utterances and gestures are paired, they can provide complementary information, as identified in other works. However the quality of speech in terms of grammatical rigour appears to be very poor here, certainly poorer that in the IBL corpus where gestures were not allowed. In this case, it is possible that this bears no consequence, as it may turn out that most learning can be done by imitation. In task instruction, there is a fixed amount of information to communicate and user may just "spread" that information across modalities. Thus, multimodal communication may not always carry more information than unimodal information.

7 References

L. Boves, A. Neumann, L. Vuurpijl, L. ten Bosch, S. Rossignol, R. Engel, and N. Pfleger. (2004) Multimodal Interaction in Architectural Design Applications. Proceedings, UI4ALL 2004: 8th ERCIM Workshop on "User Interfaces for All" 28-29 June 2004, Vienna, Austria.

Bugmann G., Klein E., Lauria S. and Kyriacou T. (2004) Corpus-Based Robotics: A Route Instruction Example. in Proceedings of IAS-8, 10-13 March 2004, Amsterdam, pp. 96-103.

Bugmann G., Wolf J. C., Robinson P. (2005) The Impact of Spoken Interfaces on the Design of Service Robots. Industrial Robot, 32:6, 499-504

Roy, D. (2005), "Semiotic Schemas: A framework for Grounding Language in Action and Perception", Elsevier, Artificial Intelligence Journal, Volume 167, Issues 1-2, Pages 170-205 (http://web.media.mit.edu/~dkroy/papers/pdf/aij_current.pdf)

Mellish, C.S., (1985), "Computer interpretation of natural Language descriptions", Ellis Horwood Limited, Chichester, ISBN 0-85312-828-6

Dennis Perzanowski, William Adams, Alan C. Schultz and Elaine Marsh (2000) Towards Seamless Integration in a Multi-modal Interface. Proceedings of the Workshop on Interactive Robotics and Entertainment, Carnegie Mellon University: AAAI Press, 3-9, April 2000.

Perzanowski, D. , Shultz A.C. and Adams W. (1998) , "Integrating Natural Language and Gesture in a Robotics Domain" Proceedings of the IEEE International Symposium on Intelligent Control: ISIC/CIRA/ISAS Joint Conference, Gaithersburg, MD: National Institute of Standards and Technology, 247-252, September 1998.

Perzanowski, D., Brock, D., Blisard, S., Adams, W., Bugajska, M., Schultz, A., Trafton, G., Skubic, M. (2003) Finding the FOO: A Pilot Study for a Multimedia Interface, In Proceedings of the IEEE Systems, Man, and Cybernetics Conference, Washington, DC

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

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

Table 1. Example explanations of the dealing phase. The table shows utterances spoken by the teacher, related gestures and meaning of the combined input. Some details are discussed in the text. Times are given in $1/10^{th}$ of a second.

No	Text – 03	Gestures	Semantics
1	Ill just explain how you deal the cards <tv t="2396" until="2428">		gamestate = learn_dealing
2	er what you do is first of all is er you take three cards for Yourself <tv t="2431" until="2477">	<objmove t="2416" user="t" ID="D/5" from="+Table+Stock" to="+Temp2" until="2442"/> <objmove t="2446" user="t" ID="C/2" from="+Table+Stock" to="+Temp2" until="2460"/> <objmove t="2463" user="t" ID="H/QQ" from="+Table+Stock" to="+Temp2" until="2477"/>	start_learn_sequence(seq1) set1 = D/5,C/2,H/QQ and owner(set1,robot) seq1_step1 = goal(move set1 from stock to temp2)
3	face down and ill take three <tv t="2486" until="2513">	<objmove t="2482" user="t" ID="D/QQ" from="+Table+Stock" to="+Temp1" until="2496"/> <objmove t="2499" user="t" ID="D/KK" from="+Table+Stock" to="+Temp1" until="2510"/> <objmove t="2512" user="t" ID="D/AA" from="+Table+Stock" to="+Temp1" until="2522"/>	set2 = D/QQ,D/KK,D/AA seq1_step2 = goal(move set2 from stock to temp1) owner(set2,human)
4	you take these into your black Area <tv t="2540" until="2563">	<objmove t="2531" user="t" ID="D/QQ" from="+Temp1" to="+Hand1" until="2544"/> <objmove t="2547" user="t" ID="D/KK" from="+Temp1" to="+Hand1" until="2567"> <objmove t="2570" user="t" ID="D/AA" from="+Temp1" to="+Hand1" until="2577">	Ref. Resolution: "these" = set1 seq1_step3 = goal(move set1 from temp2 to hand2)
5	so you can drag them down <tv t="2564" until="2575">		Ref. Resolution: "them" = set1 , hence "down"=hand2 no action required, already done
6	and then er turn them over <tv t="2606" until="2627">	<objrot t="2599" user="t" ID="D/AA" roty="0" /> <objrot t="2644" user="t" ID="D/QQ" roty="0" /> <objrot t="2613" user="t" ID="D/KK" roty="0" />	Ref. Resolution: "them" = set1 seq1_step4 = goal(turnover set1)
7	so you can see them and obviously i cant see them <tv t="2660" until="2675">		Ref. Resolution: "them" = set1 (this sentence can be used for confirmation)
8		<objmove t="2580" user="t" ID="D/AA" from="+Hand1" to="+Hand1" until="2582"/>	The card is not mentioned and didn't change location. Therefore this move is unimportant
9	and then what we do next is er <tv t="2680" until="2704">		- (supports we are still in seq1)
10	put four cards face up on the Table <tv t="2708" until="2760">	<objmove t="2695" user="t" ID="H/2" from="+Table+Stock" to="+Table" until="2715"/> <objmove t="2719" user="t" ID="D/3" from="+Table+Stock" to="+Table" until="2736"/> <objmove t="2740" user="t" ID="C/KK" from="+Table+Stock" to="+Table" until="2753"/> <objmove t="2668" user="t" ID="D/JJ" from="+Table+Stock" to="+Table" until="2692"/>	set3 = H/2,D/3,C/KK,D/JJ seq1_step5 = goal(set3 cards from pile to table)
11	so ill just turn those over <tv t="2821" until="2834">	<objrot t="2808" user="t" ID="D/JJ" roty="0" /> <objrot t="2820" user="t" ID="H/2" roty="0" /> <objrot t="2832" user="t" ID="D/3" roty="0" /> <objrot t="2844" user="t" ID="C/KK" roty="0" />	Ref. Resolution: "those" = set3 seq1_step6 = goal(turnover set3)

Linking Speech and Gesture in Multimodal Instruction Systems

Joerg C. Wolf, *Student Member, IEEE*, and Guido Bugmann

***Abstract*—This paper analyses the timing of gesture and speech acts in a corpus (MIBL) of free-flowing human-to-human instruction dialogues. From there, an algorithm is proposed to establish the pairing between speech and gesture of the instructor. It is shown that correct pairing requires timing and semantic information. Further work will explore the use of this algorithm in unconstrained free flowing multimodal instruction dialogues between human and robot. A brief overview of a robotic system is given, that is able to learn a card game from a human teacher.**

I. Introduction

IN the future, service robots should be programmable by anybody interacting with them. There are far too many possible tasks for the robot to be pre-programmed completely and users want to change the robots behaviour to their individual preference [1]. Users may not be experts in programming Therefore "programming" of service robots should be done in the language of humans. Humans teach by step-by-step task instructions. So the robot becomes a student and the human an instructor who teaches it. What do humans do when they teach? Humans teach by speaking and demonstrating. Therefore a service robot must be designed to understand natural language and demonstrations, in the domain where it is going to be taught. Looking at examples how humans teach is best done by collecting a corpus. This approach is called "corpus-based robotics" [1,2]. In corpus-based robotics, the interaction between human-teacher to human-student is analysed and the human-student is then replaced by a robot-student. A corpus provides the researcher with all the information required to design the robot to cope with unconstrained flow of speech and gesture (demonstrations).

So far our research group has investigated two corpora, one in the Instruction Based Learning project (IBL) and one in the Multimodal IBL project (MIBL). There are only few multimodal corpora aimed at human-robot interaction studies [3]. The IBL corpus contained route instructions mainly composed of sequences of actions. In the current project (MIBL), we focus on instructions also containing rule specifications. These are found frequently in game instructions. Using the same corpus-based method, we started with recording card game instruction dialogues between a teacher and a student. The teacher could demonstrate actions using a touch screen (figure 1) and all movements on the screen were recorded. The aim of this work is to develop a system capable of understanding such game instructions, build them into an internal representation and subsequently play the game with the user/teacher. In the chosen setup, the robot needs neither artificial vision nor effectors, as it can "see" cards moved on a touch screen and can play by moving cards on the touch screen. It allows concentrating the research on the learning process.

The plan of this paper follows the main development stages of the MIBL card-game learning system. Section II describes the corpus collection and its analysis. This includes an identification of functions referred to in utterances. It also includes an analysis of the type of gestures found in the corpus. Section III focuses on the process of determining which speech events correspond to which gesture events. In free flowing human-to-human instructions, these events start and end at different times and a combination of timing and semantic rules are required to achieve perfect pairing. Section IV proposes a system implementation based on the current findings. Section V concludes.

II. Corpus Collection and Analysis

A. Corpus Collection

The MIBL corpus was collected by recording dialogues between a person who already knows a card game (teacher) and another person who doesn't know the card game, using the setup shown in the figure below.

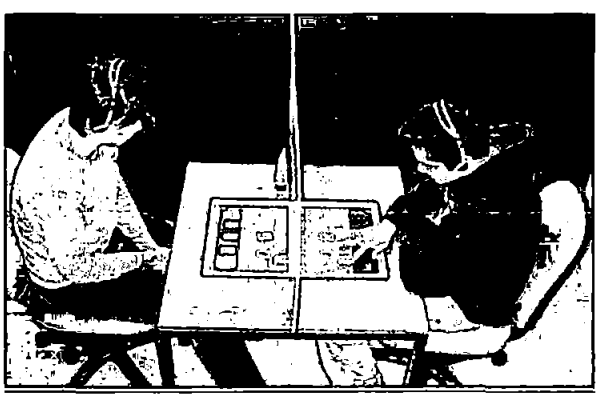


Fig. 1: Corpus collection setup. The instructor on the right moves a card on the touch screen. The learner sees a copy of the move on her screen.

This work was supported by a studentship of the University of Plymouth.
J. C. Wolf and Guido Bugmann are with the School of Computing Communications and Electronics, University of Plymouth, Drake Circus, Plymouth, PL48AA U.K., (e-mail: Joerg.wolf a_t plymouth.ac.uk, gbugmann a_t plymouth.ac.uk).

In card games, gestures can be pointing gestures, gestures moving cards from one place to another (e.g. stack to table, hand to table), re-arranging gestures (making a group of cards look tidier) and turning-over gestures. The separation panel between instructor and learner force the gesture component of the communication to take place via the touch screen and can easily be recorded. Each screen has a small "private" black band representing the hand of the player. The larger green area represents the table and is shared by the two players.

The dialogue was unconstrained; the participants were allowed to describe the card game at their own pace in their own words.

Transcription was done using dedicated multimodal transcription software called MuTra [4]. MuTra generates XML files with utterances and gesture content and timing. Table 1 shows information extracted from transcription files. U_i are the utterances and G_i are the gestures (from the touch screen). So far we have only analysed explanations of the dealing phase of the game.

TABLE I
EXAMPLE DIALOGUE (SESSION 03 FROM MIBL CORPUS)

No	Time in 10th sec.	utterance text or gesture semantics
U0	2396-2428	"I will just explain how you deal the cards"
U1	2431-2477	"er what you do first of all is.. er you deal three cards for yourself"
G1:	2416-2477	move(D/5,C/2,H/QQ, Stock , Temp2)
U2	2486-2513	"face down and I will take three"
G2	2482-2522	move(D/QQ,D/KK,D/AA, Stock , Temp1)
U3	2540-2563	"you take these into your black area"
G3	2531-2577	move(D/QQ,D/KK,D/AA,Temp1,Hand1)
U4	2564-2575	"so you can drag them down"
U5	2606-2627	"and then er turn them over"
G5	2599-2644	turn(D/5,C/2,H/QQ)
U6	2660-2675	"so you can see them and I can't see them"
U7	2680-2704	"and then what we do next is er"
U8	2708-2760	"put four cards face up on the table"
G8	2668-2753	move(H/2,D/3,C/KK,D/JJ,stock,table)
U9	2772-2778	"yep just four on the table"
U10	2801-2815	"yeh and three for each player"
U11	2821-2834	"so I will just turn those over"
G11	2808-2844	turn(H/2,D/3,C/KK,D/JJ)
U12	...	...

see text for explanations

B. *Analysis of speech transcriptions*

The corpus provides all information required to write a grammar and tune speech-recognition software. Currently a statistical language model has been trained with the corpus using NUANCE 8.5, a user independent speech recognition system.

Analysing the utterances of the transcriptions also reveals primitive procedures that the robot has to be able to carry out (the robot's "prior knowledge"). Such "language primitives" are specific to the level at which humans communicate with each other. They can constitute complex robot procedures which may require the use of micro planers (see section IV).

The following language primitive have been identified in the dealing phase:

```
start_of_sequence(name)
end_of_sequence()
deal(objects,amount,target)
move(objects,amount,source,target)
turn(objects)
owner(objects,player)
visible(objects,player)
count(objects,amount)
```

For instance U3 from the example above would need to be mapped onto a function call of the form:

```
move(objects=these?,num_of_cards=3,target=hand2)
```

Many of these primitives can only be completely specified and resolved using a combination of speech and gesture information. For instance the primitive function of U3 contains a "these" which can be resolved by the objects identified in the gestures.

C. *Analysis of gesture transcriptions*

Raw gesture data are a trail of X, Y coordinates of where the card is positioned on the touch screen. In case of a real robot, such tracking data could be the output the robot's vision system. The "analogue" trail of X, Y data of a cards position is then registered as a movement from a start area to a destination area ,e.g.

```
move(H/2,D/3,C/KK,D/JJ,stock,table).
```

Where "stock" is the source screen area and "table" is the target area of the cards. These areas, namely: stock, table, hand1, hand2, temp1 and temp2 divide the screen. The areas numbers and their boundaries are defined from observations of where the movements of the players usually end. These areas are currently simple squares and gesture labelling is straightforward [7]. If a vision system were to be used, the added uncertainty could call for the use of more complex probabilistic methods [5,6].

In general, gestures taken alone do not constitute a complete specification of the instruction. This is probably not true for the dealing phase where simply copying the gestures (without language) would be sufficient for the robot to deal correctly. However, in later phases of game instructions, such as in winning a trick, gestures only constitute examples, where objects of action are to be specified in general terms by the content of the spoken instructions. Therefore it is important to determine which speech act corresponds to which gesture.

Sections II B) and II C) have argued that language or gestures alone do not carry a complete message. Speech and

gesture are acquired through different channels and must be re-associated to reconstruct or determine the complete meaning of a message. In the next section we exploit the idea of using temporal synchronization of speech and gesture.

III. Linking Speech and Gesture

A. *Pairing of speech and gesture*

A detailed analysis was carried out measuring the timing between gesture and speech of the teacher [7]. The MIBL corpus shows that verbal instructions are always in the same order as the corresponding gestures. Timing histograms (Figure 2,3) suggest the design of a pairing algorithm based on the maximum time-difference between start-of-speech/end-of-speech and start-of-gesture.

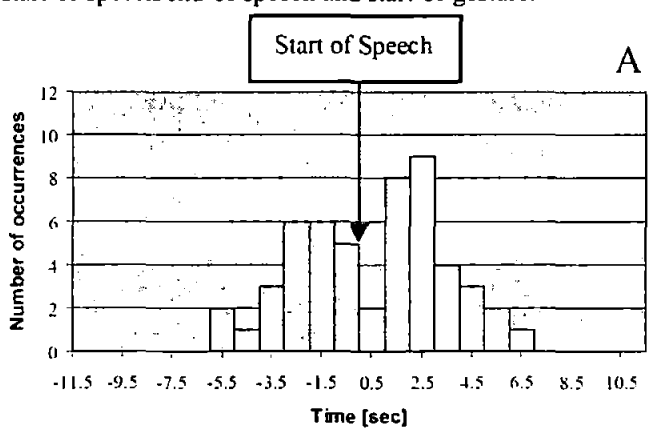


Fig 2: Histogram of the time intervals between start of speech and start of gesture.

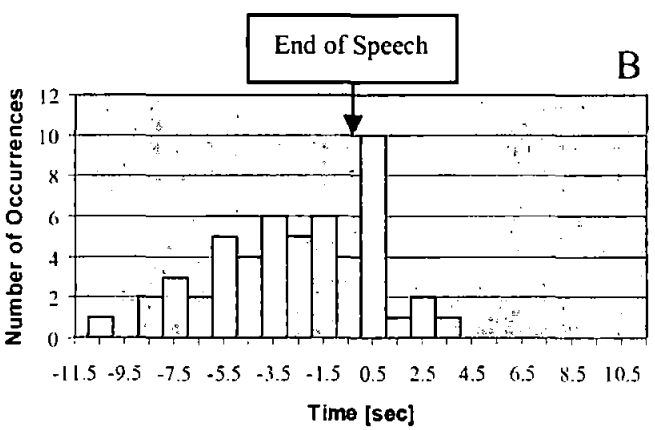


Fig 3: Histogram of time intervals between the end of speech and the start of gesture. Only gestures associated with speech events are plotted.

Figure 2 shows that gestures never start more than 5.5 sec before speech starts. Figure 3 shows that gestures never start later than 4 sec after corresponding speech ends. These observations suggest that a time window around the speech duration could be used to group speech and gesture (Figure 4). The time window borders are based on the maximum extend of the histogram.

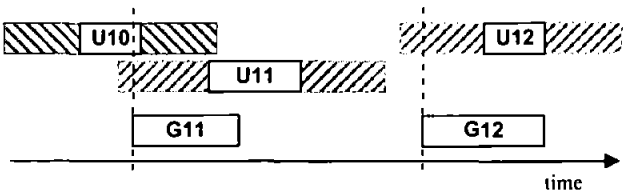


Fig. 4: This figure shows three incoming utterances and two incoming gestures. The grey areas show the maximum pairing range of the utterance. If a start-of-gesture falls within that range, it is a candidate for pairing with the utterance.

However, care must be taken with such grouping rules because time windows generally overlap.

Therefore, filters designed for grouping utterance-gesture groups can often only narrow down the candidates for grouping, but not solve the grouping problem completely.

In the example figure 4 U12 is clearly a candidate for G12, there is no confusion. U10 and U11 however could both belong to G11. In this ambiguous case, semantics must be used. In a first attempt the Gesture is assigned to the nearest neighbour utterance. In the MIBL corpus, this results in 83% correctly grouped cases.

The analysis of the 17% erroneous groupings revealed that they occur systematically with utterances which point to incompatible language primitives. For instance, in figure 4, when trying to pair G11, U10 is a reply to a question from the student and therefore not related to G11. U10 does not refer to the primitive `turn(objects)`.

Using timing alone pairs U10 with G11, while semantic filtering, as just described, eliminates U10 from the pairing candidates. Inspection of the corpus indicates that this algorithm can achieve perfect pairing.

B. *Semantic Integration of Speech and Gesture*

Once speech and gesture is paired, semantic integration must take place. Work is currently underway to develop first-order predicate logic statements that carry out the unification, although temporal logic could be considered as well. A Prolog rule that compares the parameters of the language primitive to the parameters of the gestures is at the core of the mechanism. The following 4 cases can occur as a result of pairing:

1) Completion:

A gesture and an utterance are individually incomplete, but complete each other.

$n_s = 1$, all variables are resolved

2) Confirmation:

A gesture and an utterance are individually complete. When combining they match.

$n_s = 1$, no variables exist

3) Contradiction:

The gesture supplies contradicting semantics when compared to the utterance.

$$n_s = 0$$

4) Under-specification:

The gesture and language combined are still semantically underspecified. Therefore several possible candidates are returned.

$$n_s > 1$$

Where n_s is the number of solutions of the Prolog rule.

Note that the completion-case can be used to do reference resolution. In the MIBL corpus, a specific set of cards is often co-referred with "them","these" or "those". This part of the system is not discussed in this paper.

IV. Proposed System

Shown in figure 5 is an overview of the system implementing concepts described in previous sections. Interestingly, Perzanowski [8,9] produced a similar system proposal independently.

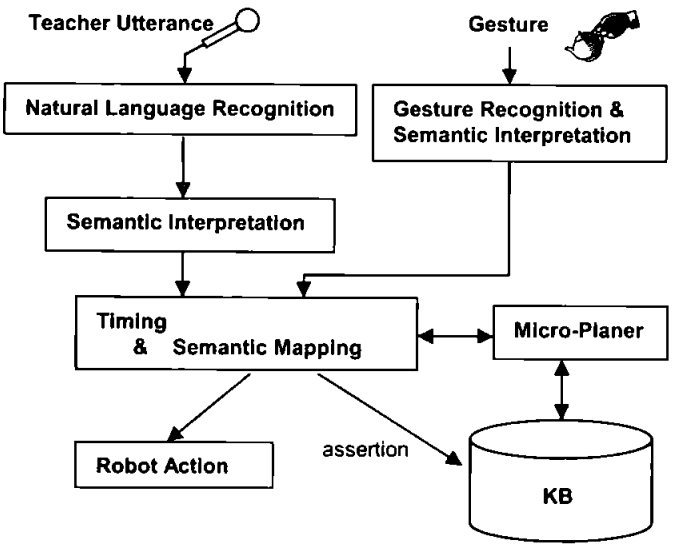


Fig. 5. Multi-modal input processing in the MIBL robot.

We are currently using a statistical language model for the language recognition. A robust interpretation grammar extracts the semantics.

A multithreaded application (one for gesture and one for speech recognition) forwards information to the Timing & Semantic Mapping process. The semantics are unified and the micro-planer is consulted.

The micro-planer produces a detailed plan of what the robo should do. Sentences such as "take out all the eights, nines and tens from the deck", are one primitive to a human, but require a variety of robot-actions to be carried out at the low level (i.e moves and comparisons). The micro-planer is a problem solver which returns the steps required for the robot to achieve the language-primitive. If a single solution-path is returned the problem is solved. The path is executed if needed, and stored if the robot is in its learning phase.

The resultant plan can be a robot action or a change in the knowledge base. Robot actions range from moving cards to replying to the user via a text-to-speech processor.

V. Conclusion

The work shows that it is possible to pair speech and gesture as occurring in unconstrained human-to-human instruction dialogue. The proposed pairing algorithm combines timing and semantic information. Further work will explore if this algorithm allows unconstrained free flowing multimodal instruction from human to robot.

References

[1] Bugmann G., Wolf J. C., Robinson P. The Impact of Spoken Interfaces on the Design of Service Robots. *Industrial Robot, 32:6*, 2005, pp 499-504.

[2] Bugmann,G., Klein, E., Lauria, S., Bos, J. and Kyriacou T. "Corpus-Based Robotics: A Route Instruction Example" *in Proceedings of IAS-8*, 10-13 March 2004, Amsterdam, pp. 96-103.

[3] Green A., Hüttenrauch, Topp E.A., and Eklundh K.S.,"Developing a Contextualized Multimodal Corpus for Human-Robot Interaction", *in the Proc. 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy 2006

[4] Wolf J.C., Bugmann G. Multimodal Corpus Collection for the Design of User-Programmable Robots. *Proc. Taros 2005*, London, pp. 251-255.MuTra link: (http://www.swrtec.de/swrtec/mibl/mutra/index.php)

[5] Naphade, M.R.,Kristjansson T.,Frey B. and Huang T.S., "Probabilistic Multimedia Objects Multijects: A novel Approach to Indexing and Retrieval in Multimedia Systems" *in Proc. IEEE International Conference on Image Processing*, Volume 3, pages 536-540, Oct 1998, USA, Chicago

[6] Roy Deb, "Semiotic schemas: A framework for grounding language in action and perception", *Artificial Intelligence, ELSEVIER*, Vol 167(1-2), pp. 170-205 , (Sept 2005)

[7] Wolf J.C., Bugmann G., "Integration of visual and spoken input in robot instructions" *in the Proceedings of the European Robotics Symposium*, Italy, Palermo, 2006

[8] Perzanowski, D., Shultz A.C. and Adams W., "Integrating Natural Language and Gesture in a Robotics Domain" *in Proceedings of the IEEE International Symposium on Intelligent Control: ISIC/CIRA/ISAS Joint Conference, Gaithersburg, MD: National Institute of Standards and Technology*, 247-252, September 1998.

[9] Perzanowski, D., Adams W., Shultz A.C. and Elaine Marsh "Towards Seamless Integration in a Multi-modal Interface." *in Proceedings of the Workshop on Interactive Robotics and Entertainment*, Carnegie Mellon University: AAAI Press, 3-9, April 2000.

Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication, Technische Universität München, Munich, Germany, August 1-3, 2008

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)

978-1-4244-2213-5/08/$25.00 ©2008 IEEE

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.

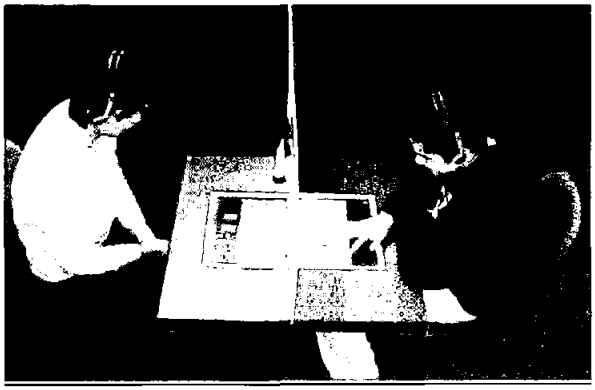


Fig. 1: Corpus collection setup. The instructor on the right moves a card on the touch screen. The learner sees a copy of the move on her

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 Primitive
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:")}
```

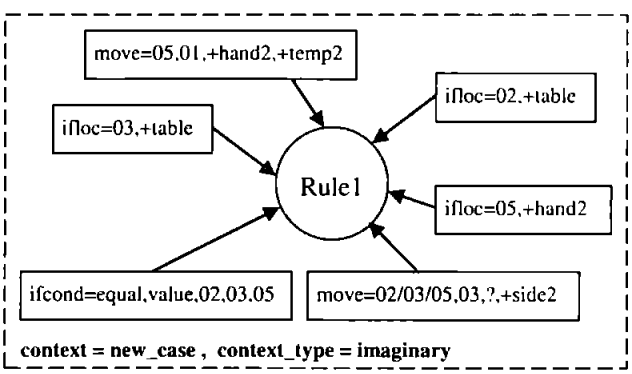


Fig 2: Rule Frame, a collection of semantics that are connected to Rule1 since the utterances were in the same context. This rule frame is used to construct a function that the robot can carry out to play the game/ perform a task. If there are question-marks left, or ambiguities, the robot can clarify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

Understanding Rules in Human-Robot Instructions

Joerg C. Wolf *and* Guido Bugmann

Abstract— **This paper presents an overview of the systematic creation of a human-robot instruction system from a multi-modal corpus. The corpus has been collected from human-to-human card game instructions. A design procedure is introduced that helps creating a speech recognition grammar which is closely linked to semantics and the corpus, so avoiding unwanted over-generation. Particular attention is paid to rule-instructions, since they are more challenging to implement than sequential and knowledge manipulating instructions. A brief overview is given on how the robot stores knowledge coming from instructions using an ontological object-oriented form. Furthermore a problem-solver is described that can reason with the newly gained knowledge. The aim of the work is to enable users to naturally instruct robots without prior knowledge about the robot. A further aim is to simplify and expedite the process of implementing multi-modal human robot instruction systems by engineers.**

I. Introduction

THE ability of future service robots to learn from end-user instructions would be a great advantage, since service robots can not completely be pre-programmed by the manufacturer [1]. There are far too many possible tasks for a service robot to be pre-programmed in advance. End-users are usually not familiar with programming; therefore the robot has to be able to understand instructions at a human level. We are investigating the structure and implementation of human-robot instruction systems that allow naive instructors to interact naturally with the robot verbally and with gestures.

In order to discover how humans speak about a task, a corpus ("body") of conversations between a human teacher and student instructing a task is collected. So far our research group has investigated two corpora, one in the IBL (Instruction Based Learning) project [2] and one in the MIBL (Multimodal IBL) project [3]. It is possible to use a corpus collection setup for human-to-human or human to wizard-of-oz to collect the corpus. We used a human-to-human setup.

In the current project (MIBL), we focus on instructions containing rule specifications. These are a found frequently in game instructions. Using the same corpus-based method, we started with recording card game instructions dialogues between a teacher and a student of the Italian card game Scopa. The teacher explains the rule verbally and could demonstrate actions using a touch screen (figure 1) and all movements on the screen were recorded.

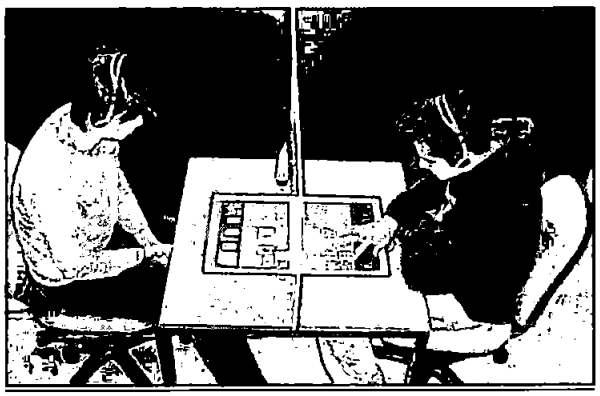


Fig. 1: Corpus collection setup. The instructor on the right moves a card on the touch screen. The learner sees a copy of the move on her screen.

The aim of this work is to develop a system capable of understanding such multi-modal game instructions, build them into an internal representation and subsequently play the game with the user/teacher. In the chosen setup, the robot needs neither artificial vision nor effectors, as it can "see" cards moved on the screen and can play by moving cards on the screen. While the IBL project has been completed by implementing and testing human-robot interaction, work is still in progress in the MIBL project. Most game instructions have already been implemented and simple human-robot interaction can take place.

In this paper we describe the instruction types found in the MIBL card game corpus followed by a detailed description on the design of a speech recognition grammar based on the corpus. Furthermore a reasoning system is described that can store, plan and carry out instructions. The paper concludes with a summary of this novel procedure of creating a human-robot instruction system. A brief overview of the steps involved in creating a human-robot instruction system is given in table 1 along with the paragraph where these are described.

Final manuscript received May 30, 2007. This work was supported by the University of Plymouth.

Guido Bugmann and Joerg C. Wolf are with the School of Computing Communications and Electronics, University of Plymouth, Drake Circus, Plymouth, PL48AA U.K., (e-mail: Joerg.wolf -a-t- plymouth.ac.uk).

-4244-1635-6/07/$25.00 ©2007 IEEE.

TABLE I
METHOD OF DESIGNING A HUMAN-ROBOT INSTRUCTION SYSTEM

Step	M/A/S	Chapter
1. Corpus design and collection	M	I.
2. Multi-modal Transcription	M+S	II.A
3. Cutting utterances into clauses	A	III.B
4. Context tagging	M	II.A
5. Semantic annotation & ontology design	M+S	II.B,V.B
6. Generalisation within semantic classes (exchangeable words)	M+S	III.A, III.C
7. Grammar generation	A	III.C
8. Multi-modal integration & reference resolution	M	IV
9. Language primitives implementation	M+S	II.B

M = must be a Manual task
S = task can be assisted by smart editor
A = can theoretically be fully automated task

II. CORPUS ANALYSIS

A. *Initial Corpus Analysis*

The recordings have been transcribed using the multi-modal transcription tool MuTra [3]. The transcriptions include start time and duration of gesture and speech. The transcription process could be simplified by adding speech recognition software. However, all transcribed text has to be confirmed manually since the corpus provides the reference data for all further system development. The transcription is stored as a XML file indicating which utterance belongs to which gesture. Each teacher explanation was tagged so that tasks and sub tasks are hierarchically divided. In our case the explanation has been divided with XML tags into the three game phases of dealing, game-play and counting points at the end. This could be referred to as *context tagging*. Inside the dealing phase we have found 6 sequential instructions of moving and turning of cards (sub-tasks). In the game-phase we have tagged the rules on pairing/capturing cards and the description of the ranking of cards. The tagging process also helps the developer to break down complex explanations into logical parts for which robot functions can be implemented correspondingly, step by step. An example below (`20.xml/912-955`) shows the tagging of the value-rule, which describes the value of a card in points.

```
<dealing>
...
</dealing>
<game>
  ...
  <value-rule>
    <tv t="912" until="955">
      and the jack queen and king become the eight nine
      and ten
    </tv>
  </value-rule>
  ...
</game>
```

The tagging process is also useful for automatically generating a grammar later.

B. *Language Primitives and Instruction Types*

Analysing the utterances of the transcriptions reveals primitive procedures which the robot has to be able to carry out before learning from the end-user can start (the robot's "prior knowledge"). Such "language primitives" are specific to the level at which humans communicate with each other. They can constitute complex robot procedures in the background. These language primitives can be categorized into facts, sequential actions, context indicators and conditionals. Some examples of transcriptions and corresponding primitives:

Facts:

```
21.xml/696-723:"erm ok so the deck were
                playing with"

21.xml/729-748:"is a forty card deck"
21.xml/758-779:"with the eights nines and tens
                removed"
```

Corresponding Primitive:

```
not_exist(card=08,card=09,card=10)
```

Sequential actions:

03.xml/2431-2481:"er what you do first of all is.. er you deal three cards for yourself face down"

Corresponding Primitive:

```
move(objects=these?,num_of_cards=3,target=hand2)
```

As for context indicator and conditionals, examining the corpus for game rules reveals that a game rule is constructed from:

1. initial *context indicator*: e.g. "suppose you" or "if"
2. *conditionals*: e.g. "have an ace" or "cards with equal rank"
3. a sequence of *instructions* that have to be carried out when the conditionals are satisfied

For example:

```
21.xml/1621-1648: "so like with two sevens
                   if you had a seven you could only
                   take one of them"
```

Corresponding Primitives:

```
context(context=new_case)
ifloc(card=07,location=hand2)
ifloc(card=07,location=?)
ifcond(type=equal,compare=value,card=07,card=07)
move(objects=them?,num_of_cards=1,target=?)
```

These primitives are inserted into the transcriptions as XML tags.

The question mark in the semantics means, that there is missing information. Missing information is completed by combining semantics, multi-modal integration (section IV) and by requests to the teacher.

These three types of primitives are implemented in different

ways. *Facts* result in manipulations of the knowledge base using Prolog statements. Sequential *instructions* are implemented as C-routines affecting the physical behaviour of the robot, i.e. moving the robot arm. The third type, *Context indicator* and *conditionals* initiate the creation of rule frames (section V. A), which are stored in the knowledge base.

Rules are also found in other domains, such as cooking, where every ingredient has to be multiplied with the number of persons. Previous to this corpus our group collected a corpus on route instructions to a driver in a town. These did not contain rules. Therefore, the primitive categories which occur are domain dependent.

III. Semantic Clause-Based Grammar

An advantage of *corpus-based robotics* (method for collecting a corpus of the domain before building the robot [1]) is the availability of a corpus that can be used for generating grammar. Generally a grammar from a corpus should truly represent the content of the corpus.

A. The overgeneration problem

Most work in grammar induction from texts is irrelevant for application specific language as it aims at generating a grammar of the whole language from a small corpus of example sentences [for example, see 4]. These grammars massively overgenerate, by design, and are unsuitable for the application-specific spoken interfaces. The main problem of overgeneration is that instruction-sentences can be recognized which the robot is not able to carry out or comprehend.

In order to create a grammar that has over-generation only in appropriate places, such as generalizing over numbers or colours, we created semantic classes for words and phrases. Word-classes are semantic categories. For example the number of cards is a different class to the rank of cards. Using the same sub-grammar would lead to overgeneration in the wrong place. One could, for example, refer to "1 card" but not to a card with rank "1" since the smallest rank of a card is "2". These word-classes are then also used as a class in the knowledge representation scheme and as language primitive parameters, when passing information from the language recognition module to the reasoning system.

B. Clause-Based Grammar

A problem of speech recognition is the fact that the speaker may pause before finishing the sentence (inappropriate end of speech). At this stage we assume that partial sentences are complete clauses. To cope with multiple clauses, they are linked at a higher level of the grammar. The concept has been named *clause-based grammar.*

Corpus utterances have been cut into clauses by the help of a natural language parser. Cutting is done if clauses are linked with words like "and", "and then" or "so". For identifying the end of a clause, The Apple Pie Parser [5] provided most accurate results on the MIBL corpus.

The implementation of this grammar is written in Nuance GSL (Grammar Specification Language), which utilizes the slot filling concept. When a grammar rule (in this case made of a clause) is hit during speech recognition, variables (slots) are filled with values. These slots are usually are in form of a first semantic interpretation such as "go=forward". Slots are the interface variables between the grammar and the application specific software that processes the interpretation. To preserve the order in which the clauses were said during interpretation, the semantic information of each clause is concatenated and passed to the reasoning system through a single slot. Now a user can arbitrarily give one or more instructions in one utterance. We are currently investigating how to optimise the search tree of the speech recognition without the loss of flexibility.

C. Full Corpus Coverage

Clauses have been grouped by their language primitive using the context tags during annotation of the corpus. The advantage of semantic grouping is that it ensures correct overgeneration at a local level. Semantic grouping also helps the grammar designer to structure and identify the semantics, which initially looks as an overwhelming task when looking at a corpus of thousands of utterances. The definition of word-classes is an easy task for non-specialists in natural language processing. The resulting structure is a grammar template which is easy to convert into GSL grammar. The most trivial solution to convert a corpus into a grammar is to simply copy all transcriptions into the grammar. Hence all clauses become GSL grammar rules. This ensures that the grammar initially covers the whole corpus. This template is the starting point for the grammar designer.

Our aim is to reflect the corpus as accurately as possible. Even colloquial expressions are kept. For example:

14.xml/17428-17440: "thats right innit"

This is important for real world applications when the robot is used in a household. It is indeed possible to add sentences in good (corrected) English to the corpus, in a controlled way.

The appropriate semantics are now attached to each utterance in the grammar, manually. Alternatively, this could be done automatically if the XML transcriptions have the utterances already annotated with primitives (semantics). Now, the bespoke word-classes can be substituted to create controlled over-generation.

An example on how to achieve correct overgeneration (generalisation) is given below. First a version of the grammar without and then with generalisation:

Simple:

```
(if you have say a five)
{
  context_type=imaginary
  ifloc=05,+hand2
}
```

Generalised:

```
(if you have say a CardCat:c)
{
  context_type=imaginary
  ifloc=$c,+hand2
}
```

This step in the grammar implementation is a time consuming but easy process. In order to expedite this process, a smart editor could be used that keeps track of word classes and suggests substitutions automatically. The smart editor could also suggest language primitives, based on previous related sentences. A related editor has been suggested by [6, 7] and a graphical grammar editor has been suggested by [8], although these do not have the full functionality required for the above task.

Once the grammar and slots for semantic interpretation have been designed, the output (slots) is ready for reasoning. The first step in reasoning is to resolve references. Here multi-modal information can be used.

IV. Multi-Modal Issues

The MIBL system allows input through gestures, namely movements of cards on the touch screen and verbal instructions. From a robot perspective these modalities are two communication channels that need to be recombined into one message. The diagram in figure 2 summarises the multi-stage process of multi-modal integration ("recombination") or sometimes referred to as multi-modal fusion.

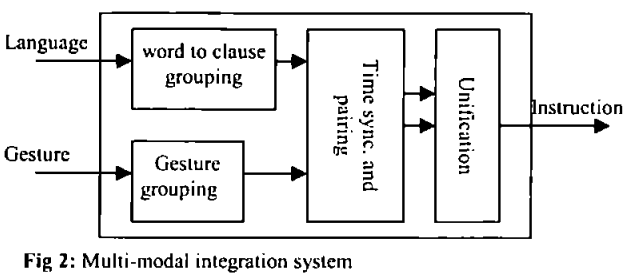


Fig 2: Multi-modal integration system

The diagram shows how gestures are first grouped in order to be represented at the same level as verbal instructions. For example the utterance "put three cards onto the table" is one instruction, but consists of at least three gestures. The integration of free flowing speech and gesture has added complexity. In this case utterances can follow very closely behind each other and, as Oviatt correctly describes in [9], it is a myth that speech and gesture always have a time overlap if they belong together. In recent work [10] we described a time synchronisation and pairing algorithm that finds the right language-gesture pair in order to unify the two information channels. Timing and semantic information must be used to achieve a satisfactory performance in multi-modal integration. With a combination of timeouts, nearest neighbour match and of gesture and language primitive verbs the algorithm achieves pairing the right gestures and utterances.

Multi-modal reference resolution can be carried out during multi-modal integration. The following example is taken from the transcriptions (03.xml/2540-2563). It mentions the word "these", referring to cards. In order to resolve what is meant by "these" multi-modal integration is essential.

```
<tv t="2540" until="2563">
  you take these into your black area
  <objmove t="2531" user="t" ID="D/QQ" from="+Templ"
to="+Hand1" until="2544"></objmove>
  <objmove t="2547" user="t" ID="D/KK" from="+Templ"
to="+Hand1" until="2567"></objmove>
  <objmove t="2570" user="t" ID="D/AA" from="+Templ"
to="+Hand1" until="2577"></objmove>
</tv>
```

Typically the instruction only gives a minimum of information. For example if the source location of these cards (in this case +Templ) is apparent, it is not mentioned in the language.

Let's review the path of the information so far. An utterance and gestural data was initially recognised using a grammar, then connected to a semantic interpretation such as "move(...)" or "ifloc(...)". These interpretations have been grouped and logically unified, which was briefly described in this section. Finally these interpretations can be used for reasoning, which is described in the next section.

V. Knowledge Representation

The representation and reasoning with the information coming from the gestures and language primitives required a hybrid system. Factual knowledge or the world-model of the robot is represented in an object oriented database. Teacher instructions that require action, on the other hand, are represented as *rule frames*. These rule frames are then turned into rules for a problem solver. The implementation of the knowledge representation of the MIBL system is entirely in Prolog.

A. Rules and Actions

Whereas facts are stored in the object database, action instructions and game rules are initially stored in a *rule frame*. A rule frame is a collection of hints on how the complete rule may be constructed. These hints consist of initial *context indicator*, *conditionals* and a sequence of *instructions* that have to be carried out when the conditionals have been satisfied (see example in section II.). An example is given

below where a teacher describes a rule of the card game Scopa. The example shows utterances with their corresponding language primitives (slots) in the grammar.

```
(the other possibility is)
  {return("context=new_case:")}

(if you have say a five)
  {return("context_type=imaginary:ifloc=05,+hand2:
")}

(and you see on the table there you got a three and a
two)
  {return("ifloc=03,+table:ifloc=02,+table:")}

(you can bring the five forward)
  {return("move=05,01,+hand2,+temp2:")}

(you take the three and the two because that is equal
to five )
  {return("ifcond=equal,?,02,03,05:")}

(the same value that is on your card so then you can
take all the cards to your side )
{return("ifcond=?,value,?,?,?:
         move=deictic_determinative,?,?,+side2:")}

(we will just take it forward so you can show your
opponent that you have got a five )
{return("move=05,01,+hand2,+temp2:")}
```

From the example it is clear that rules are not described in a single utterance. A rule frame is required as a temporary structure to collect parts of a complete rule (Fig. 3). We have also found cases were the user does not explain game rules in the expected order. Often conditionals are mixed with instructions. The frame approach allows acquiring rule information provided in random order. A rule can then be checked for consistency and logical completeness. A completion of a rule description is detected by a context change. It is rare that a user specifically mentions that a rule explanation is complete. Usually a context change is detected by the start of a new rule explanation, which triggers the completion of the previous.

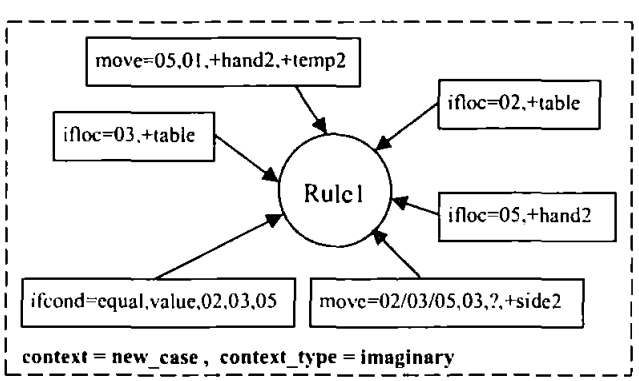


Fig 3: Rule Frame, a collection of semantics that are connected to Rule1 since the utterances were in the same context. This rule frame is used to construct a function that the robot can carry out to play the game/ perform a task. If there are question-marks left, or ambiguities, the robot can clarify information with the teacher before creating the new robot function.

If a rule explanation is complete, the content of the rule frame is translated into a Prolog program that represents a state-transition rule for a problem-solver. If it is the robots turn to play, the problem-solver is consulted. The problem-solver now tries to find and apply the appropriate game rule for the given situation in order to compete his turn. This approach requires actions to be stored as steps for the problem solver. As described these steps (state-transition rules) are not prior knowledge; they are generated by the speech and gestures from the teacher.

Templates for state-transition rules are defined by the grammar designer. These templates map language primitives to actions at the robot-level. Current robot level actions are moving cards, turning over cards, comparing ranks of cards, counting cards, removing cards. Their implementation is hardware specific.

B. Factual Knowledge

Knowledge of physical objects and their properties are stored by the robot in an object-oriented format. The robot has an innate prior knowledge of playing cards and their properties. Properties are attributes of a class. Classes are structured in a tree taxonomy. For example a playing card is a 3D-object. A 3D-object has coordinates. See figure 4.

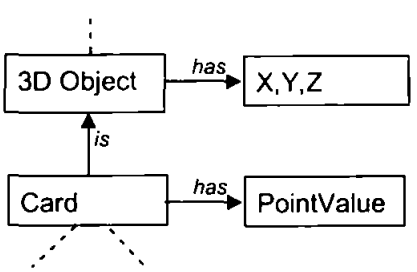


Figure 4: Extract of the ontology of cards for the MIBL reasoning system

Essentially the is-relationship indicates classes and sub-classes. A has-relationship indicates properties. Properties are inherited. For example cards inherit the property of coordinates (see Fig. 4). Currently multiple inheritance is not allowed. However, multiple ontological trees can be combined. The modelling of this ontology closely follows [11,12], who suggest ontology design based on Basic Formal Ontology (BFO).

```
is_a('face_card','rank').
is_a('cardinal','rank').
...
is_a('JJ','face_card').
is_a('QQ','face_card').
...
has_a('object3d','X').
...
has_a('cardname','rank').
has_a('cardname','suit').
...
```

Table 2: extract from the implementation of the ontology in Prolog

Actual cards that are present on the playing table are instances of classes. This representation system allows storing of the current situation and factual knowledge about the robots world. The previously described primitives that refer to a fact are manipulating this knowledge. For example the fact that the eight has been removed from the game translates into the Prolog statement:

```
forall(
    get_property(INSTANT,'rank',08),
      (delete_instance(INSTANT))
)
```

VI. Conclusion and Summary of Method

This paper described a method of creating multi-modal interfaces for instructing robots. Initially a corpus is collected using a multi-modal interface between two humans. After transcription the utterances are grouped hierarchically by phases of the task and finally utterances are grouped by language primitives. Utterances are split by a parser so that sequential instructions/clauses are separated. A grammar is generated, containing all corpus sentences. At this stage it becomes clear what the structure of the ontology of the domain could be. Ontological classes are created, which are then also used as sub-grammars for targeted overgeneration and as language primitive parameters.

Regarding the translation of rules, it has been found that rules are a combination of language primitives consisting of context indicators, conditionals and instructions that have to be carried out if the conditionals are true and the context is correct.

References

[1] Bugmann G., Wolf J. C., Robinson P. The Impact of Spoken Interfaces on the Design of Service Robots. Industrial Robot, 32:6, 2005, pp 499-504,

[2] Bugmann G., Klein, E., Lauria, S., Bos, J. and Kyriacou T., "Corpus-Based Robotics: A Route Instruction Example" *in Proceedings of IAS-8*, 10-13 March 2004, Amsterdam, pp. 96-103.

[3] Wolf J.C., Bugmann G. Multimodal Corpus Collection for the Design of User-Programmable Robots. *Proc. Taros 2005*, London, pp. 251-255. (http://www.swrtec.de/swrtec/mibl/mutra/)

[4] Perfors A. Tenenbaum J., and Regier T. "Poverty of the stimulus? A rational approach" In the Proceedings of the 2006 Cognitive Science conference. 2006

[5] Satoshi Sekine, Ralph Grishman, "A Corpus-based Probabilistic Grammar with Only Two Non-terminals", Fourth International Workshop on Parsing Technology 1995

[6] Wang Y., Acero A. "Grammar Learning for Spoken Language Understanding", In Proceedings of ASRU Workshop. Madonna di Campigilo, Italy, Dec 2001

[7] Wang Y.,Acero A. "Rapid Development of Speech Recognition Grammars", Speech Communication, Vol. 48, No. 3-4.390-46

[8] Monaco Peter C., "Creating and Editing Grammars for Speech Recognition Graphically", United States Patent No 6434523 B1, Nuance Communications, Menol Park U.S.A 13/08/2002.

[9] Oviatt, S. L. "Ten myths of multimodal interaction" Communications of the ACM, Vol. 42, No. 11, November, 1999, pp. 74-81.

[10] Wolf Joerg C., Bugmann, Guido. "Linking Speech and Gesture in Multimodal Instruction Systems" *in the Proceedings of RO-MAN 06, The 15th IEEE International Symposium on Robot and Human Interactive Communication*, Hatfield, U.K., pg. 141-144

[11] Spear Andrew D., "Ontology for the Twenty First Century: An Introduction with Recommendations", Manual for Basic Formal Ontology from the Institute for Formal Ontology and Medical Information Science, Saarland University, 2006

[12] 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)

[13] Van den Bosch, A., "Careful abstraction from instance families in memory-based language learning.", Journal of Experimental and Theoretical Arificial Intelligence, 11:3, special issue on Memory-Based Language Processing, W. Daelemans, guest ed. Pp. 339-368