16th IEEE International Conference on Robot & Human Interactive Communication
August 26 ~ 29, 2007 / Jeju, Korea

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.

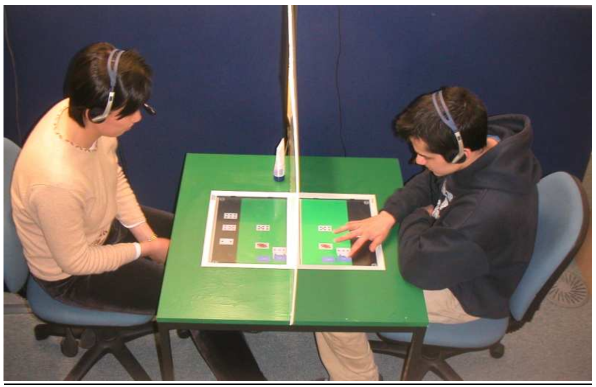

Fig. 1: Corpus collection setup. The instructor on the right moves a card on the touch screen. The learner sees a copy of the move on her screen.

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).

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.

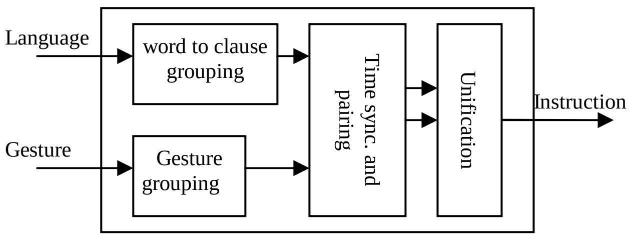

Fig 2: Multi-modal integration system

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="+Temp1"
to="+Hand1" until="2544"></objmove>
  <objmove t="2547" user="t" ID="D/KK" from="+Temp1"
to="+Hand1" until="2567"></objmove>
  <objmove t="2570" user="t" ID="D/AA" from="+Temp1"
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 +Temp1) 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.

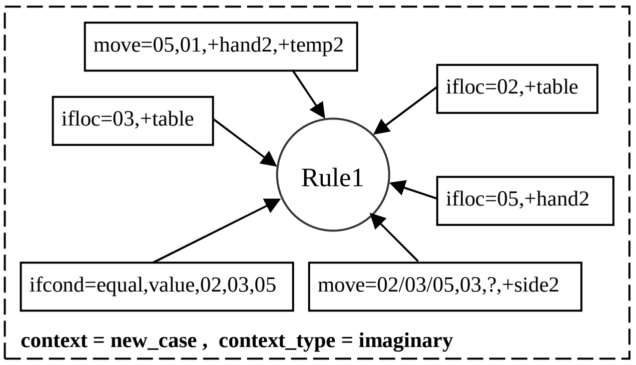
move=05,01,+hand2,+temp2
ifloc=02,+table
ifloc=03,+table
Rule1
ifloc=05,+hand2
ifcond=equal,value,02,03,05
move=02/03/05,03,?,+side2
context = new_case , context_type = imaginary

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.

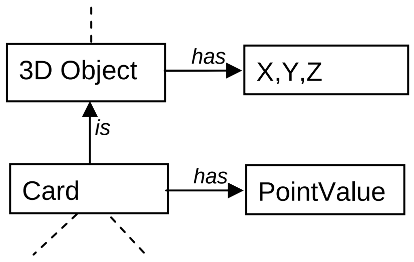

Figure 4: Extract of the ontology of cards for the MIBL reasoning system

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