Natural Programming of a Social Robot by Dialogs

Javi F. Gorostiza and Miguel A. Salichs
RoboticsLab
University Carlos III of Madrid
Av. Universidad, 30
28911 Leganes, Madrid
jgorosti@ing.uc3m.es salichs@ing.uc3m.es

Abstract
This paper aims at bringing social robots closer to naive users. A Natural Programming System that allows the end-user to give instructions to a Social Robot has been developed. The instructions derive in a sequence of actions and conditions, that can be executed while the own sequence verbal edition continues. A Dialogue Manager System (DMS) has been developed in a Social Robot. The dialog is described in a voiceXML structure, where a set of information slots is defined. These slots are related to the necessary attributes for the construction of the sequence in execution time. The robot can make specific requests on encountering unfilled slots. Temporal aspects of dialog such as barge-in property, mixed-initiative, or speech intonation control are also considered. Dialog flow is based on Dialog Acts. The dialog specification has also been extended for multimodality management. The presented DMS has been used as a part of a Natural Programming System but can also be used for other multimodal human-robot interactive skills.

Introduction
One of the main problems associated with social robots is how they are going to adapt to human common life environments. To contribute to that adaptation, in this paper an End-user Programming System based on a Dialogue Manager System (DMS) is proposed. The system is framed in what is known as the Instruction-Based Learning paradigm, a special kind of Natural Programming.

In (Biggs and Macdonald 2003), natural programming systems are distinguished in two different methods: manual programming and automatic programming. The former requires the user to directly enter the desired behavior of the robot using a graphical interface or a text-based programming language. This manual method has been the most common way of programming robots to date. On the other hand, we have automatic programming systems, where the user has no direct access to the system code. They are divided into three different types: Learning Systems, Programming by Demonstration, and Instructive Systems. This paper focuses on the latter one.

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Instruction is a natural way for humans to give information of how to accomplish some specific tasks. By means of verbal information, humans can refer to different actions and also relate those actions in some order. Speech can express rules and sequences of instructions in a very concise way.

The works presented in (Lauria et al. 2002) and (Lauria et al. 2001) show a natural programming system that includes conversational interaction. A mobile robot in a city environment is instructed to navigate from one place to another. Instruction-Based Learning (IBL) is presented as a framework where the user can teach the robot a sequence of serial routes around the environment by voice. The system has a domain of approximately 14 action primitives. A DMS based on TRINDI (Larsson et al. 2004) allows the semantic interpretation of user utterances. This semantic representation is transformed into the internal language of the robot. The robot is able to prompt the user to repeat or rephrase an utterance when the recognition confidence value is below a certain threshold. Similar works have been presented in (Dominey, Mallet, and Yoshida 2007), where a conversational-based programming system for HRP-2 robot is shown.

These systems are very similar to the system described here. But we have increased the possibilities both in the complexity of the program structures (including parallelism), number of sequence actions and conditions, and in the communicative interaction level.

The purpose of the present work is to combine a DMS with an End-User Natural Programming System in a Social Robot. The DMS is based on the interpretation of voiceXML structures1 and Speech Act theory (Searle 1970) and Dialog Acts based paradigm. Some extensions for multimodality have also been included. The dialogs between the user and the robot are mixed-initiative. The robot can understand different natural ways of saying the same information, and manages misunderstandings and coherence lacks. It is also able to manage interruptions, turn changes, prosodic control in the speech, user silences, etc. The DMS communicates with a Sequence Interface that creates a program from the information gathered through the human-robot conversational interaction.

1A description of the standard can be consulted in the web site http://www.w3.org/TR/voicexml20/
One of the main goals of our purpose is to allow the user to make a program for the robot that uses complex programming structures like concurrent execution branches, condition selection, and loops. Therefore, in our approach, a Sequence Function Chart (SFC) or GRAFCET structure is used. This work is the continuation of the results presented in (Gorostiza and Salichs 2009), where a Sequence Verbal Edition System is shown.

The global system presented will increase the natural aspect of Human-Robot Interaction since it integrates a complex Sequence Generation System and a versatile Dialogue Manager System in a Sequence Verbal Edition System.

**Dialogue Manager System and Sequence Verbal Edition**

Figure 1 shows a general diagram of the Sequence Verbal Edition System. It is divided into three main parts: the Skill Level, the Dialogue Manager System, and the Sequence Generator. Two important skills for human-robot dialog are shown in the diagram: asrSkill and ettsSkill. The former takes charge of the Automatic Speech Recognition, the latter of the emotional Text-to-Speech synthesis. The DMS loads and parses a voiceXML structure. It also communicates with the Sequence Generator System, that generates a sequence from the dialog information. The SFC sequence is executed by the Sequencer, that parses a special XML structure where the sequence is defined. These main parts are described in detail in the next sections.

**Speech Skills**

**Speech Recognition and Semantic Grammars**

The Automatic Speech Recognition skill, asrSkill, uses a Loquendo-ASR-7.0 engine that performs speech recognition using a special type of grammars with two main parts: literal and semantic ones. The literal part is ABNF-based and defines the context-free formal language the recognizer is able to recognize, that is, a set of different lexemes and syntactic structures that relate these lexemes. In the semantic part, simple scripts are implemented and are executed when a specific coincidence between a literal part and the user utterance happens. In these scripts an assignment is made to the semantic attribute. This attribute will later be used as an input in the voiceXML structure of the DMS.

One important semantic attribute is the Dialogue Act associated with an utterance. This attribute is implemented in the semantic grammar as an attribute that takes values that are inspired by the DAMSL (Dialogue Act Markup in Several Layers) standard with some modifications. For example, ad (action directive), for the addition of an action or a condition to the sequence, fp (conventional opening), for greetings for beginning the conversation, ky (acknowledgment or “yes” answer), etc. The Dialogue System uses the Dialogue Act attribute to respond properly to the user utterances. E.g. if the user utters a greeting, the robot responds with another greeting, etc.

Each speech recognition result has a global confidence value that is used in the DMS. Therefore, the robot can ask for confirmations when this value is too low.

Grammars have been designed from the set of initial actions and conditions that users can take for creating their programs, and from how users are going to perform this natural programming process. We have done a study of a small corpus of user speech in two senses. First of all, we wanted to know how humans describe a program to a robot. Then, we wanted to study the essential elements of a natural dialogue between the end-user and the social robot. The results of these studies have been used to implement the semantic grammars, in both sides: their literal part and the semantic attribute-value pairs.

**Speech Synthesis and Prosodic Control**

An exhaustive description of the speech synthesis capability and the processes of prosodic control is out of the main scope of this paper. Nevertheless, as this skill is part of the whole DMS some main features are commented. The Emotional Text-to-Speech skill, ettsSkill, synthesizes the text to be spoken with a natural intonation. Analogously to asrSkill, ettsSkill uses Loquendo-TTS-7.0 engine that performs the Text-to-Speech synthesis. The skill can control certain prosodic features as volume, pitch or velocity so the spoken text could express some emotional and intentional aspects of natural speech. The skill also allows the articulation of fillers as coughs, laughs, yearns, etc.

These features and what is going to be said is implemented a priori by the programmer as canned text in the voiceXML file.

On the other hand, in natural dialogs speakers usually interrupt each other, and stop the own ongoing utterance exchanging the speaking turn. Therefore, the robot has to be able to stop its own speaking in certain dialog circumstances, for example when the user begins to speak, etc. This feature, called barge-in, is also included in the ettsSkill.

**Dialogue Manager System and voiceXML**

The DMS connects the speech skills to each other by parsing a voiceXML structure. This structure is implemented in a set of vxml files that define the Dialogue System: prompts to be synthesized, fields to be recognized and other dialogue features as time-outs, barge-in property, etc. More than one file can be used to define the overall dialog.

A multimodal extension has also been implemented in the voiceXML structure. This extension has two main purposes. In one side, it is used for communication with the Sequence Generator where the sequence is created, and with the Sequencer that executes the created sequence. In the other side, it handles multimodal information both for non-verbal expression and recognition skills as will be explained later.

**voiceXML Features**

This W3C standard allows the definition of a dialogue system based on slots of information. The main part of a
**voiceXML** dialogue is the *form*, where a set of *fields* are defined. Each of these fields has a *name*, a *prompt* and a *filled* element. The name defines an ECMAScript variable where the content of the field is stored. The prompt is used by the DMS to control the output of synthesized speech. The filled element specifies an action to perform when a combination of one or more fields is filled.

The Dialogue System is connected with the semantic grammars. Each field of a form matches up with each attribute from the semantic grammar. Each field is checked by a specific algorithm: the Form Interpretation Algorithm (FIA). If the information gathered from the user utterance does not fill the field, then the prompt is visited and output speech is synthesized. In the other side, if users fill some field by their utterances, the actions described in the filled element are executed.

Therefore, the FIA defines the main flow of the dialog. But some special events can change the dialog movements. For example, if there is a misunderstanding between the robot and the user, the `<nomatch>` event is thrown; also if the user keeps a long silence, the `<noinput>` event is thrown. These events can be managed in a specific way, so the robot can ask for repetitions.

**voiceXML** allows mixed-initiative dialogs, therefore the robot can take the initiative and ask for some initial field. The standard also allows the definition of some temporal properties of dialog. For example, the maximum time-out for waiting for a user utterance, barge-in property, and also a definition of prosodic properties for Text-to-Speech synthesis.

**Multimodal Extension in the DMS**

The DMS has been extended for multimodal management. Multimodal input information is grouped using the Natural Language Semantic Markup Language (NLSML) standard\(^3\). As an example of a non-verbal skill, the `touchSkill` is included in figure 1. This skill emits any tactile event associated with the robot capacitive sensors to the DMS.

Any type of non-verbal input information that can be represented in NLSML format can be introduced in the DMS. The name of the labels created in the NLSML structure has to match up with the name of the fields in the forms of the voiceXML structure.

The DMS sends commands to the rest of the architecture by means of the `#$-parser`. This parser checks the `<prompt>` voiceXML labels looking for `#` and `$` special symbols in the prompt text. This allows embedding function calls inside of the prompt label. Therefore, the DMS can send any command to the rest of the architecture inside a `<prompt>` label from the voiceXML structure. This label allows verbal and non-verbal expression.

`#$-parser` is also used to send sequence edition commands from the DMS to the Sequence Generator. Once a sequence is created, the DMS can also send commands to the Sequencer to execute it.

**Sequence Interface: Edition and Execution**

### Sequence Function Chart Features

The Sequence Function Chart specification is used to represent the programs that the user creates. The SFC is based on GRAFCET and Petri nets. In a sequence there are two types of nodes that are connected in an alternated order: *steps* and *transitions*. The former are associated with the actions that the sequence can do. Each step can be activated or

---

\(^3\)NLSML specification can be consulted in http://www.w3.org/TR/nl-spec/
deactivated. On the other hand, each transition is associated with a condition of the environment or the own robot. Transitions control the activation and deactivation of the actions to which these are connected.

Actions and conditions can be connected following a complex sequence structure as shown in its specifications (see (I.E.C. 1988)) and in our previous work (Gorostiza and Salichs 2009). To sum it up, there are three basic sequence structures that can be used to combine the actions and conditions:

• Simple Sequence. It is the classic sequential program: one node follows another in a condition-action sequence.

• Sequence Selection. This structure allows the evaluation of different conditions and then the execution of one specific sequence branch among different options.

• Simultaneous Sequences. This structure allows the execution of different sequence branches at once, in parallel.

We implement the sequence in a XML structure where the configuration of actions and conditions is described. Each action is associated with two functions: Activation Function and Deactivation Function. Each condition is associated with one function: Evaluation Function. These functions are implemented in the Python programming language. We have chosen this language because it can be implemented and interpreted at execution time by the Sequencer.

Therefore, to create a sequence it is necessary to have an initial set of actions and conditions that could be combined in a sequence structure.

Initial Set of Actions and Conditions

Maggie is the social robot where all the systems that have been described here have been implemented and tested. The details of the robot hardware and control features have been presented in several works such as (Gorostiza et al. 2006) or (Salichs et al. 2006). In the low level of hardware possibilities, she has multiple multimodal sensors: tactile, telemeter, speech recognition, camera, etc; and multiple actuators: she can move her head, her eyelids, her arms, translate or rotate, speak, etc. Maggie also has several interactive skills, such as following a person, face detection, teleoperation, etc.

The challenge we want to face is how to make all those possible movements accessible to the end-users so they can combine them and build a program by natural interaction with the robot. We have started by creating an initial set of actions and conditions that can be explicitly implemented in execution time with the information gathered by the Dialogue Manager System. As explained above, we have chosen the programming language Python for implementing actions and conditions. At execution time Python functions are created. These functions bridge between the set of actions and conditions of the robot and the spoken information of the user.

The interactive skills of Maggie, as described above, are multiple, but in this work a reduced initial set of actions and conditions has been chosen just to test the functionality of the developed system. The whole design allows to increase ad libitum this initial set, easily. To be managed, an action or condition has to be described with three main features:

<table>
<thead>
<tr>
<th>Syntax I</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>raise</td>
<td>left arm, lArm.move(ArmTop)</td>
</tr>
<tr>
<td></td>
<td>right arm, rArm.move(ArmTop)</td>
</tr>
<tr>
<td>put down</td>
<td>left head, lArm.move(ArmDown)</td>
</tr>
<tr>
<td></td>
<td>right head, rArm.move(ArmDown)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax II</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>turn</td>
<td>head, Neck.turn(NeckLeft)</td>
</tr>
<tr>
<td></td>
<td>right, Neck.turn(NeckRight)</td>
</tr>
<tr>
<td></td>
<td>to the left, Base.turn(BaseLeft)</td>
</tr>
<tr>
<td></td>
<td>to the right, Base.turn(BaseRight)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax III</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward</td>
<td>Base.move(BaseFront)</td>
</tr>
<tr>
<td>Backward</td>
<td>Base.move(BaseBack)</td>
</tr>
</tbody>
</table>

Table 1: Syntactic structures in the mention of an action.

- Context-free grammar that is used by the asrSkill to recognize the action or condition from the user utterances.
- Semantic grammar or set of pairs attribute-values that allows the system to extract the main features of the action or condition from the recognized user utterance.
- Python functions that activates or deactivates the action or evaluates the condition.

The possibilities of Maggie’s movement are related to a set of semantic attributes which describe those movements formally. The Python functions are built at execution time by combining the different semantic attributes. For example, the Python activation functions of the initial set of actions are described in the last column of Table 1, that will be explained later.

Adding an Action to the Sequence

Table 1 shows the essential verbal information for identifying each action of the initial domain of actions. We have made a previous study on how humans mention different actions. Different syntactic structures are used and same lexemes can be used with different meanings. We have separated the different ways the user can mention an action of the initial domain in three types of syntactic structures. Therefore, there is a specific semantic attribute that identifies the used syntax in the user utterance. It is called syntax, and in the example it can take values 1, 2, or 3. This attribute is used by the DMS to know how to combine the values of the other attributes to build the Python functions. In the example, the three different syntactic structures are described as follows.

The first type of syntax is:

Verb + [Name-Complement] + Name

The second type of syntax is:

Verb + [Name] + Verb-Complement
And the third one is just:

Verb

where the square brackets indicate an optional element.

This is shown as an example on how to semantically classify an utterance syntax scheme, and could be extended to cover more user utterances in function of the domain of the accessible actions or conditions that the robot handles.

As an example, if the user utters something like

“Maggie, raise the left arm”

the semantic attributes involved would be as described next:\n
@da = ad
@type = action
@verb = move
@body = Arm
@limit = Top
@axis = left
@syntax = 1

These attributes show that the Dialogue Act (@da) is an action directive (ad) of adding an action (specified in the attribute @type). They also describe the properties of this action. The DMS takes these attribute values and creates the Python method call associated with the action as follows:

leftArm.move( ArmTop )

This function is embedded in a XML file where the sequence is implemented as a SFC Activation Function. When this method is executed, it communicates with the specific skill so the left arm is moved to the ArmTop position value that, in the example, is chosen a priori:\n
Notice that the same attribute could take a different semantic value depending on the mentioned action or condition. For example, in the utterance “Raise the left arm”, the lexeme “left” points at the lexeme “arm” to specify which arm is involved in the action, but in the utterance “Turn left”, the same lexeme has a different meaning. This distinction is described in the @syntax attribute as described in table 1.

Semantic grammars allows different lexemes to be gathered in one attribute. E. g. the user can ask the robot to turn left with different utterances: “turn left”, “twist to the left”, “spin left”, etc, but in all the cases the same verb attribute is assigned to the value turn.

Adding a Condition to the Sequence There are two different types of conditions: explicit and implicit ones. The former are explicitly mentioned by the user. For example, in the utterance “Wait until I touch your head”, the condition would be a function that checks if the touch sensor of the head is active. The implicit conditions are not explicitly mentioned by the user. For example, if the user has added an action of rising the left arm and then he/she utters something like “then...”; the condition would be a function that checks if the left arm has finished rising, that is, the end of the preceding action.

In the initial domain of conditions that we have used, the explicit conditions are associated with tactile sensor events (touching the head, one arm, the back, etc), so every transition function checks the state of one specific sensor.

Experimental Results

The overall system has been tested. Different voiceXML files have been parsed at execution time. The robot has proved to be able to react to the different dialogue acts obtained by the speech recognition from user utterances. She takes the initiative and asks for the specific attributes necessary to complete a sequence edition command. For example, if the user utters an action addition like “Raise your arm”, the robot will ask about which arm the user refers to (left or right), etc.

The robot also manages temporal aspects of dialogue. For example, if the user is in silence more than a threshold, the robot takes the initiative and request a response. Barge-in property is also managed with success.

Figure 2: Two scenarios where the user is editing and executing a sequence

Some special scenarios have also been tested. For example, relational ones, where robot and user interact to know each other: greetings, asking for help, etc. Misunderstandings moments are also managed successfully as the robot requests a repetition or asks for a specific attribute-field.

When the speech recognition score value is very low, a confirmation subdialog is used.

Complex sequences structures have been created by human-robot dialog. For example, creating some parallel execution branches. The user can change the focus of edition from one branch to another using utterances like “in the first branch”, etc. The Sequence Interface adds a node (action or condition) in the branch where the focus is.
Once the sequence is created, the user can ask for sequence execution, and stop the sequence at any point.

Figure 2 shows two different scenarios where the user is editing and executing a sequence. The dialogue between the user and the robot begins with some greetings. The user asks for adding actions and explicit conditions. In the dialog of the figure, the user asks for adding an explicit condition, and the robot requests some missing information. When the user asks for executing the sequence, the robot begins to execute the activation and deactivation functions of the sequence steps, and the evaluating functions of the sequence transitions, depending on the structure of the sequence. This execution is made by the Sequencer, as shown in Figure 1. The user can stop or pause the sequence at any time, and make some modifications in the sequence, as well.

Conclusions

A general purpose Dialogue Manager System has been designed, implemented, and tested on a Social Robot. The DMS has been used for Natural Programming of the Social Robot by the end-user. The overall system has been designed from the state-of-the-art of dialogue management, describing the basic features that every DMS has to include. The implemented DMS parses voiceXML structures where information slots and prompts are described. It uses a set of semantic grammar rules that have been designed from the study of a corpus on how humans describe a sequence and on how humans speak naturally with a Social Robot. The DMS is mixed-initiative, based on Dialogue-Acts theory, and is able to handle some temporal features of natural human-robot dialogues. A multimodal extension has also been included in the system, using the NLSML standard. The DMS is implemented on a Verbal Sequence Edition System that uses Sequence Function Chart standard for creation of complex programming structures.

Future Works

Currently, the DMS parses a voiceXML structure that is implemented by the programmer a priori. We want to design a Planning System that could implement this structure automatically. This system would take three different types of data as input. First of all, the robot necessity for conversational interaction with the user. This will come from the robot control architecture. Also a set of information slots associated with the sequence features, what action or condition to add, what edition command to perform, etc. And finally, the necessary prompts for these information slots, that could also be learnt by human-robot interaction.

An extension of the initial set of actions and conditions is also desirable.

Temporal and non-verbal aspects of conversational interaction are also under development. voiceXML allows the management of some temporal features described above, as silence, barge-in, or turn changes. Complex non-verbal skills both for expression and perception are also in development process, so the user can edit the sequence using gestures in combination with verbal information, and the robot can express its messages in a multimodal message. These skills could be easily synchronized in the dialog description made in the voiceXML structure.

Acknowledgment

The authors gratefully acknowledge the funds provided by the Spanish Ministry of Science and Innovation (MICINN) through the project named “A New Approach to Social Robots” (AROS DPI2008-01109).

References


