The Use of Scripts Based on Conceptual Dependency Primitives for the Operation of Service Mobile Robots

Jesus Savage[1], Alfredo Weitzenfeld[2], Francisco Ayala[1], and Sergio Cuellar[1]

[1] Bio-Robotics Laboratory
Department of Electrical Engineering
Universidad Nacional Autonoma de Mexico, UNAM
Mexico City, Mexico
`savage@servidor.unam.mx`
[2] CANNES
Department of Computer Engineering
ITAM
Mexico City, Mexico
`alfredo@itam.mx`

Abstract. This paper describes a Human-Robot interaction subsystem that is part of a robotics architecture, the ViRbot, used to control the operation of service mobile robots. The Human/Robot Interface subsystem consists of tree modules: Natural Language Understanding, Speech Generation and Robot's Facial Expressions. To demonstrate the utility of this Human-Robot interaction subsystem it is presented a set of applications that allows a user to command a mobile robot through spoken commands. The mobile robot accomplish the required commands using an actions planner and reactive behaviors. In the ViRbot architecture the actions planner module uses Conceptual Dependency (CD) primitives as the base for representing the problem domain. After a command is spoken a CD representation of it is generated, a rule base system takes this CD representation, and using the state of the environment generates other subtasks represented by CDs to accomplish the command. In this paper is also presented how to represent context through scripts. Using scripts it is easy to make inferences about events for which there are incomplete information or are ambiguous. Scripts serve to encode common sense knowledge. Scripts are also used to fill the gaps between seemingly unrelated events.

1 Introduction

There is a need for reliable Human-Robot interaction systems as experienced by the proliferation of new humanoid robots in particular in Japan with advanced interaction capabilities including spoken language. This can also be appreciated by new competitions to exalt the robots social interaction with humans such as a new league in the robots' competition Robocup: RoboCup@Home. The goal of

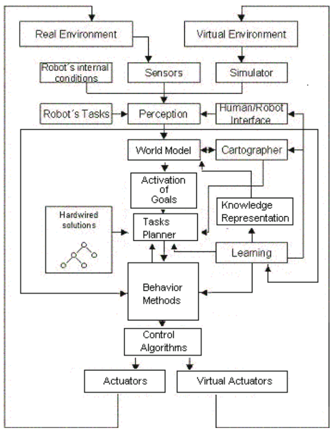

Fig. 1. The ViRbot System consists of several subsystems that control the operation of a mobile robot

this league is to promote the development of real-world applications and human-machine interaction with autonomous robots, or as they put it: "the aim is to foster the development of useful robotic applications that can assist humans in everyday life" [1]. One of the test of this competition is that a service robot helps a person to prepare a cooking recipe, to solve this task in this paper it is proposed the use of scripts that describe a set of possible of actions that a robot and persons can do under certain conditions. Also in this paper is presented the Human/Robot interface subsystem of a mobile robot architecture, the ViRbot system, whose goal is to operate autonomous robots that carry out daily service jobs in houses, offices and factories. The ViRbot system [2] divides the operation of a mobile robot in several subsystems (see figure 1). Each subsystem has a specific function that contributes to the final operation of the robot.

2 Human/Robot Interface

The Human/Robot Interface subsystem in the ViRbot architecture has tree modules: Natural Language Understanding, Speech Generation and Robot's Facial Expressions. In this paper is presented the Natural Language Understanding module.

2.1 Natural Language Understanding

The natural language understanding module finds a symbolic representation of spoken commands given to a robot. It consists of a speech recognition system coupled with Conceptual Dependency techniques [3].

2.2 Speech Recognition

For the speech recognition system it was used the Microsoft Speech SDK engine [4]. One of the advantage of this speech recognition system is that it accepts continuous speech without training, also is freely available and with the C++ source code included, it means that it can be modified as needed. It allows the use of grammars, that are specified using XML notation, which constrains the sentences that can be uttered and with that feature the number of recognition errors it is reduced. Using a grammar the transitions from one word to another is restricted, reducing the perplexity considerable. Perplexity specifies the degree of sophistication in a recognition task [5]. Almost every speech recognition system currently developed has the problem of insertion words, words incorrectly added by the speech recognition system. These worlds may cause a robot to fail to perform the asked command, then it is necessary to find a mechanism that, even if these errors exists, that a robot should be able to perform the required commands. One of the goals of this research is to find an appropriated representation of the spoken commands that can be used by an actions planner.

2.3 Conceptual Dependency

One way to represent a spoken command is by describing the relationships of objects mentioned in the input sentence [6]. During this process the main event described in the sentence and participants are found. In this work the participants are any actors and recipients of the actions. The roles the participants play in the event are determined, as are the conditions under which the event took place. The key verb in the sentence can be used to associate the structure to be filled by the event participants, objects, actions, and the relationship between them.

Dominey [7] describes the Event Perceiver System that could adaptively acquire a limited grammar that extracts the meaning of narrated events that are translated into(action,object, recipient).

Another approach is the use of Conceptual Dependency, this technique represents the meaning contained in a sentence. Conceptual Dependency is a theory developed by Schank for representing meaning. This technique finds the structure and meaning of a sentence in a single step. CDs are especially useful when there is not a strict sentence grammar.

One of the main advantages of CDs is that they allow rule base systems to be built which make inferences from a natural language system in the same way humans beings do. CDs facilitate the use of inference rules because many inferences are already contained in the representation itself. The CD representation uses conceptual primitives and not the actual words contained in the sentence. These primitives represent thoughts, actions, and the relationships between them.

Some of the more commonly used CD primitives are, as defined by Schank [8]:

ATRANS: Transfer of ownership, possession, or control of an object (e.g. give.)
PTRANS: Transfer of the physical location of an object (e.g. go.)
ATTEND: Focus a sense organ (e.g. point.)

MOVE: Movement of a body part by its owner (e.g. kick.)
GRASP: Grasping of an object by an actor (e.g. take.)
PROPEL: The application of a physical force to an object (e.g. push.)
SPEAK: Production of sounds (e.g. say.)

Each action primitive represents several verbs which have similar meaning. For instance give, buy, and take have the same representation, i.e., the transference of an object from one entity to another.

Each primitive is represented by a set of rules and a data structure containing the following categories, in which the sentence components are classified:

An Actor: The entity that performs the ACT.
An ACT: Performed by the actor, done to an object.
An Object: The entity the action is performed on.
A Direction: The location that an ACT is directed towards.

The user's spoken input is converted into a CD representation using a two step process. The CDs are formed first by finding the main verb in the spoken sentence and choosing the CD primitive associated with that verb. Once the CD primitive has been chosen the other components of the sentence are used to fill the CD structure.

For example, in the sentence "Robot, go to the kitchen", when the verb "go" is found a PTRANS structure is issued. PTRANS encodes the transfer of the physical location of an object, has the following representation:

(PTRANS (ACTOR NIL) (OBJECT NIL) (FROM NIL) (TO NIL))

The empty (NIL) slots are filled by finding relevant elements in the sentence. So the actor is the robot, the object is the robot (meaning that the robot is moving itself), and the robot will go from the living room to the kitchen (assuming the robot was initially in the living room). The final PTRANS representation is:

(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM living-room) (TO kitchen))

Another example, the phrase: John gave the book to Mary, can be represented by the following CD:

(ATRANS (ACTOR John) (OBJECT book) (FROM John) (TO Mary))

The initial structure issued to represent a sentence contains NILs of the knowledge representation that need to be filled.

(ATRANS (ACTOR NIL) (OBJECT NIL) (FROM NIL) (TO NIL))

These NILs can be filled either by the previous or following sentences, or by inference using the context surrounding the sentence.

In continuous speech recognition each sentence is represented by a CD. The NILs in the sentence's knowledge representation (i.e. the CD structure) may represent some of the sentence's words that either were wrongly recognized or were not part of the recognition vocabulary. The knowledge NILs will be filled by a mechanism in which an inference engine will use rules and context to look for the needed information.

There are also primitives that represent the state in which an object is, for the previous example the book now belongs to Mary is represented by:

(POSSESS (OBJECT Mary) VALUE Book))

Conceptual dependencies can also be used with multi-modal input [9]. If the user said "Put the newspaper over there", while pointing at the table top, separate CDs will be generated for the speech and gesture input with empty slots for the unknown information (assuming the newspaper was initially on the floor):

Speech:
(PTRANS (ACTOR Robot) (OBJECT Newspaper) (FROM Floor) (TO NIL))
Gesture:
(ATTEND (ACTOR User) (OBJECT Hand) (FROM NIL) (TO Table_top))

Empty slots can be filled by examining CDs generated by other modalities at the same time, and combining then to form a single representation of the desired command:

(PTRANS (ACTOR Robot) (OBJECT Newspaper) (FROM Floor) (TO Table_top))

In this section it was explained how to transform a multi-modal input into a CD structure which can be manipulated more easily. CD structures facilitate the inference process, by reducing the large number of possible inputs into a small number of actions. The final CDs encode the users commands to the robot. To carry out these commands an actions planning module must be used as described in the following section. CDs are suitable for the representation of commands and for asking simple questions to a robot, but they are not suitable for the representation of complex sentences.

3 Planner Subsystem

The ViRbot organization consists of several hierarchical layers. After receiving the CD representation from the Human/Robot interface the Perception subsystem perceives a new situation that needs to be validated by the World Model subsystem. The World Model validates the situation by the information provided by the Cartographer and the Knowledge Representation subsystem. The Planner subsystem takes as an input the output of the World Model subsystem and tries to take care of the situation presented. Planning is defined as the process of finding a procedure or guide for accomplishing an objective or task. In the ViRbot planner subsystem there are two planning layers, the upper is the actions planner, based on a rule base system, and the lower later the movements planner, based on the Dijkstra algorithm. Action planning requires searching in a state-space of configurations to find a set of the operations that will solve the problem.

3.1 Actions Planner

The Robot is able to perform operations like grasping an object, moving itself from on place to another, etc. Then the objective of action planning is to find a sequence of physical operations to achieve the desired goal. These operations can be represented by a state-space graph.

We use the rule base system CLIPS developed by NASA [10], as an inference engine that uses forward state-space search that finds a sequence of steps that leads to an action plan. Actions planning works well when there is a detailed representation of the problem domain. In the ViRbot architecture the actions planning module uses conceptual dependency as the base for representing the problem domain. After a command is spoken, a CD representation of it is generated. The rule base system takes the CD representation, and using the state of the environment will generate other subtasks represented by CDs and micro-instructions to accomplish the command. The micro-instructions are primitive operations acting directly on the environment, such as operations for moving objects.

For example when the user says **"Robot, go to the kitchen"**, the following CD is generated:

(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM Robot's-place) (TO Kitchen))

It is important to notice is that the user could say more words in the sentence, like **"Please Robot, go to the kitchen now, as fast as you can"** and the CD representation would be the same. That is, there is a transformation of several possible sentences to a one representation that is more suitable to be used by an actions planner.

All the information required for the actions planner to perform its operation is contained in the CD. The planner just needs to find the best global path between the Robot's place and the Kitchen, see figure 2, thus the rule base system issue the following command to the movements planner:

(MOVEMENTS-PLANNER get-best-global-path Robot's-place to Kitchen)
And the answer of the movements planner is the following:

(best-global-path $Robot's - place\ place_1\ place_2 ... place_n\ Kitchen$)

Now a new set of PTRANS are generated asking the robot to move to each of the places issued by the planner:

(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM Robot's-place) (TO $place_1$))

(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM $place_1$) (TO $place_2$))

.
.
.

(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM $place_i$) (TO $place_j$))
and finally
(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM $place_n$) (TO Kitchen))

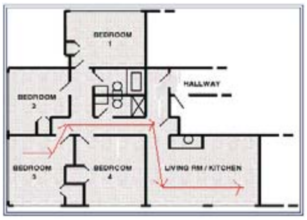

Fig. 2. The movements planner finds a global path from one of the bedrooms to the kitchen

For a more complex example, such as the user saying **"Robot, give the newspaper to the Father."**

First, the following CD representation is generated:

(ATRANS (ACTOR Robot) (OBJECT newspaper) (TO Father) (FROM newspaper's-place))

Then the actions planner finds a sequence of actions to perform the requested order, that is, the Robot needs to go for the object, to pick it up, and to deliver it to the place in which the Father is. These actions are represented by the following CDs:

(PTRANS (ACTOR Robot) (OBJECT Robot) (TO newspaper's-place) (FROM Robot's-place))

(GRASP (ACTOR Robot) (OBJECT newspaper) (TO Robot's-hand) (FROM newspaper's-place))

(PTRANS (ACTOR Robot) (OBJECT Robot) (TO father's-place) (FROM newspaper's-place))

After these CDs are issued some rules of the rule base system are fired, issuing new actions, until the Robot reaches the place where the object is supposed to be. When the Robot finds the newspaper, and is in the same room of the recipient, it will carry this object to the Father. Then the new position of the object and who has it are updated in the rule base system.

4 Context Recognition

In the previous sections it was shown how CDs can be used to recognize multi-modal commands and aid the planning necessary to carry the commands out. The final level of the rule base system recognizes sets of events, or context. Context plays an important role for understanding the information provided in a conversation, so it is used to increase the speech recognition accuracy and the robot performance.

The rule base system receives the best M phrases recognized from the speech recognition module. Starting with the first phrase the rule base system will try to interpret it according to the present context.

Occasionally, the speech system gives recognition errors, so if the top speech output does not make any sense given a particular context, then the rule base system will start looking at the second best phrase, and so on until it finds the phrase that make most sense as in the follows example:

if the user says **"Robot, bring the milk"**, and the speech recognition module returned the following best sentences:

1. Robot, bring the mail
2. Robot, bring the milk
3. Robot, bring the ink

The speech recognition module made an incorrect recognition by failing to distinguish milk from mail. The rule base system can correct this by looking at the context surrounding the sentences. If a user was in the kitchen making breakfast, then from this context is more probable that the second sentence is the correct one; that is, the user is asking for the milk and not for the mail. If this process did not work for all M phrases then the rule base system may ask the user to repeat the phrase or to provide more information.

4.1 Context Representation Using Scripts

One way to represent the context is through scripts [11]. The term script was taken from the theater community and it describes all possible sequences of events that an actor or entity may perform under certain conditions, they include the actions place. Using scripts it is easy to make inferences about events for which there are incomplete information or are ambiguous. Scripts serve to encode common sense knowledge. Scripts are also used to fill the gaps between seemingly unrelated events. Every script has different slots associated with it that are filled with actors, objects and directions when it is instantiated.

The scripts in the ViRbot rule base system are written using CD representations. As it was mentioned before, the spoken sentences are represented using CDs, and these CDs correspond with some of the script's CDs.

The first problem that appears is how to choose the appropriate script given the present conditions. The script is activated when the user says a sentence related to it, or when he performs certain actions related to the script. In either case they are CDs that match the CDs of the first steps of the script to be activated.

For example when the user says the sentence **"I will make breakfast"**, or if he is in the kitchen and he starts picking objects related to making food, then the script for making breakfast is triggered.

Here is the making breakfast script, each of the actions may not occur in the established order, and some actions may not occur at all.

* The actor enters the kitchen:
(PTRANS (ACTOR ?actor) (OBJECT ?actor) (FROM ?actor's-place) (TO kitchen))
The notation ? after a name it means a variable name that will used in all the script.
* The actor ask the robot to come to the kitchen:
(SPEAK (ACTOR ?actor) (OBJECT "Robot come to the kitchen") (TO robot))
* The robot enters the kitchen:
(PTRANS (ACTOR robot) (OBJECT robot) (TO kitchen) (FROM robot's-place))

First the actor needs to obtain all the food and tools elements to prepared the food. He will pick or ask them to the robot.
* The actor goes to the the cupboard or fridge:
(PTRANS (ACTOR ?actor) (OBJECT ?actor) (TO cupboard) (FROM ? actors-place-kitchen))
or
(PTRANS (ACTOR ?actor) (OBJECT ?actor) (TO fridge) (FROM ?actors-place-kitchen))
* The actor opens the fridge or cupboard. This action is represented by two primitives, first the actor moves its hand to the cupboard's or fridge's handle, then it applies force to it opening the door:
(MOVE (ACTOR ?actor) (OBJECT ?actor-hands) (TO cupboard's-handle))
or
(MOVE (ACTOR ?actor) (OBJECT ?actor-hands) (TO fridge's-handle))
(PROPEL (ACTOR ?actor)(OBJECT ?cupboard's-handle) (FROM ?actor-hands))
or
(PROPEL (ACTOR ?actor) (OBJECT ?fridge's-handle) (FROM ?actor-hands))
* The actor picks some of the food or tools elements:
(PTRANS (ACTOR ?actor) (OBJECT ?object) (TO ?actor's-hands) (FROM ?object-place))

The objects that are more probable to be used are of the food type and kitchen tools: eggs, onions, milk, pans, glasses, cups, knifes, etc. Some of the food needs to be transfered from one container to another, for example the milk can be transfered from the milk bottle to a glass:

(PTRANS (ACTOR ?actor) (OBJECT ?obj) (FROM ?container-obj) (TO glass))
* The actor asks for the food or tool elements to the robot:
(SPEAK (ACTOR ?actor) (OBJECT "Robot bring the ?object") (TO robot))
This instruction is represented as follows:
(ATRANS (ACTOR robot) (OBJECT ?object) (FROM ?robot) (TO ?actor's-hands))

* After the actor has all the elements to prepare food he will start making it. The representation of this actions requires to many primitives. It is assumed that he finish to prepare food when he says "The food is ready":

(SPEAK (ACTOR ?actor) (OBJECT "The food is ready") (TO robot)),

or when he starts taking the food to the dining room table:

(PTRANS (ACTOR ?actor) (OBJECT food) (FROM kitchen) (TO dining-room-table)),

or when he asks the robot to do that:

(SPEAK (ACTOR ?actor) (OBJECT "Bring the food to the table") (TO robot))

(PTRANS (ACTOR robot) (OBJECT food) (FROM kitchen) (TO dining-room-table))

After this point the script for having breakfast is selected. When one variable of the script is found then it is instantiated whenever it appears in them. In the previous example, if the Mother was the actor preparing the food then whenever the slot ?actor appears is substituted by Mother. In this way is possible that some CD's will be complete even if direct actions did not occur or were not observed.

Using scripts helps to answer questions about information that was not observed when the actions happened. For instance, if the user asks for milk to drink it, he will need to put the milk in a glass, and even if this action was not observed the script for making food specify that he did that. Then, it can be asked if the user used a glass for the milk, and the answer should be yes.

Using scripts helps to make the speech recognition more reliable, because it rejects words that does not fit into the currently active script. In the example of the previous section in which the user says **"Robot, bring the milk"**, the speech recognition module found the following sentences:

1. Robot, bring the mail
2. Robot, bring the milk
3. Robot, bring the ink

In the script of making food the second sentence is more probable than the first one, because in this script the objects that are more likely are the ones related to food.

5 Experiments and Results

The Human-Robot interaction subsystem was tested under two situations. In the first situation it was tested how well the robot performed for sentences of the type: "Robot, go with the Father", "Robot, go to the kitchen", "Robot, give the newspaper to the Mother", "Robot, where is the newspaper?", etc. The environment represented a house, set in our laboratory, based on the layout proposed by the Robocup@Home competition[1]. In this environment the robot TPR was used, see figure 3.

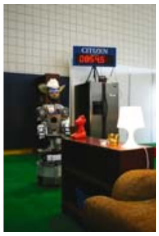

Fig. 3. Test Robot TPR8

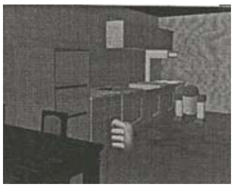

Fig. 4. In this virtual kitchen, the user's actions can be detected

In the second situation it was used a virtual reality mobile robot simulator that recreated a virtual house with a kitchen included. In this virtual environment, see figure 4, it was tested the script of making breakfast. This simulator was able to give the position and the orientation of the user's head and hand. With this information it could be detected which object the user was pointing at or grabbing and thus the user's actions could be related to the CDs of the making breakfast script.

The speech recognition system alone, under noise conditions, found correctly the following categories, in which the sentence components are classified in the spoken sentences, with the following frequencies: Actors with 94%, Objects with 90%, Recipients 73%, Verbs with 83%, Questions with 86%, and words that were not part of the sentence (insertion words) 70%. The insertion words, words incorrectly added by the speech recognition system, may cause the robot to fail to perform the asked command, which means that the robot could perform correctly, the commands only 30% of the time. Combining all the information provided by the Microsoft Speech Recognizer with Conceptual Dependency techniques, the system was able to overcome this problem and it found a representation that it was executed correctly by the robot 75% of the time.

6 Conclusions and Discussion

Conceptual Dependency representation helped to increase the recognition rate and also helped the actions planner to perform the desired user's command. The use of Conceptual Dependency to represent spoken command given to a robot helped to the actions planner in the ViRbot system to find a set of steps to accomplish the required commands. Conceptual Dependency is useful to represent basic commands and simple dialog with a robot, but is not useful to represent more complex spoken commands. The most significant contribution in this research is the successful combination of a commercially available speech recognition system and AI techniques together to enhance the speech recognition performance. At this time, the system has the following features: speaker independent, medium size vocabulary, loose grammar and context dependent. For future research we will have more scripts of common daily activities. The ViRbot system was tested in the Robocup@Home [1] category in the RoboCup competition at Bremen, Germany in 2006 and in Atlanta in 2007, where our robot TPR8, obtained the third place in this category. The scripts module will be used in the test in which a service robot helps a person to prepare a cooking recipe in the Robocup@Home competition, at the RoboCup 2008 in China.

References

1. Robocup@Home (2006), http://www.ai.rug.nl/robocupathome/
2. Savage, J., Billinhurst, M., Holden, A.: The ViRbot: a virtual reality robot driven with multimodal commands. In: Expert Systems with Applications, vol. 15, pp. 413–419. Pergamon Press, Oxford (1998)
3. Schank, R.C.: Conceptual Information Processing. North-Holland Publishing Company, Amsterdam (1975)
4. Microsoft Speech SDK (2006), http://www.microsoft.com/speech/
5. Rabiner, L., Biing-Hwang: Fundamentals of Speech Recognition. Prentice Hall, Englewood Cliffs (1993)
6. Savage, J.: A Hybrid System with Symbolic AI and Statistical Methods for Speech Recognition. PhD Dissertation University of Washington (August 1995)
7. Dominey, P.F., Weitzenfeld, A.: A Robot Command, Interrogation and Teaching via Social Interaction. In: IEEE-RAS International Conference on Humanoid Robots, December 6-7, Tsukuba, Japan (2005)
8. Lytinen Steven, L.: Conceptual Dependency and Its Descendants. Computer Math. Applic. 23(2-5), 51–73 (1992)
9. Cohen, P.R.: The Role of Natural Language in a Multimodal Interface. In: Proceedings UIST 1994, pp. 143–149. ACM Press, New York (1992)
10. CLIPS Reference Manual Version 6.0. Technical Report Number JSC-25012. Software Technology Branch, Lyndon B. Johnson Space Center, Houston, TX (1994)
11. Schank, R.C., Leake, D.: Computer Understanding and Creativity. In: Kugler, H.-J. (ed.) Information Processing 1986, pp. 335–341. North-Holland, New York (1986)