Integrating Robot Task Scripts with a Cognitive Architecture for Cognitive Human-Robot Interactions

Yuchul Jung, Yoonjung Choi, Hugun Park, Wookhyun Shin and Sung-Hyon Myaeng

School of Engineering, Information and Communications University
119, Munjiro, Yuseong-gu, Daejeon, 305-732, Korea
{enthusia77, choiyj35, gsgphg, commne, myaeng}@icu.ac.kr

Abstract

In spite of the long history of intelligent humanoid robots, endowing human-like commonsense knowledge to a robot is still a difficult problem. Since a knowledge base consisting of a large number of rules and facts is not an efficient structure that can express situational and background knowledge for humanoid robots, more compact yet forgiving representation is required. Our proposal is to employ a script design that contains richness and diversity needed for robot’s task planning, in conjunction with a robust cognitive architecture called EM-ONE, the latest extant account of an implemented cognitive architecture. The script structure has its advantages in flexibility and extensibility for a variety of situations or tasks at hand, along with reusability. As the number of scripts increases, the coverage for diverseness of Human-Robot Interaction (HRI) situation grows. In this paper, we discuss three cognitive models used as our cognitive architecture basis and describe our efforts for generating task scripts in a semi-automatic way by reusing the already existing scripts.

1. Introduction

Humanoid robots need to be flexible and sensible enough to deal with an uncontrolled environment where various types of objects may exist. Such an environment would encompass a complex and unpredictable setting with uncertainties. A robot in a house or a hospital can be seen as in such an environment since existing objects may move and/or new objects may be introduced.

However, previously developed humanoid robots face hardships due to the lack of robustness even though they have domain specific knowledge. It is not possible to provide a robot with all the necessary knowledge, or commonsense knowledge, for robustness, such as facts, task rules, location information, user profiles, etc. A cognitive architecture is amenable for a commonsense thinking paradigm, especially when it is combined with script representation, where more robust reasoning and high-level cognitive behaviors are expected.

To our knowledge, there exists no standard for robot task knowledge representation. The reason seems to be attributed to the difficulties in dealing with cognitive aspects such as estimated or experienced knowledge on the related context and the actions full of changes. We believe that using script (or story) representation is feasible for large scale robot task knowledge, more so than simply adopting decomposition of hierarchical Task Network (HTN) [2]. There are four advantages for using script representation [1]:

1) Scripts connect knowledge to pursue.
2) Scripts help us control inference.
3) Scripts are easy to acquire.
4) Narratives can contextualize knowledge.

Generally, we expect that home-service robots should evolve by utilizing previously learned experiences and currently obtained information simultaneously, learning new knowledge by mistakes, and interacting with humans. Script representation seems to be the most natural way to support the evolutionary process accommodating the diverse information. Let’s suppose a situation: When a result of prediction/reasoning for the next action (or behavior) of the robot is unclear, an interaction with the human can make it possible for the robot to catch the necessary clues from the human. These interaction details can be also contained with preconditions and promising delegates of post-conditions.

In EM-ONE [1], Singh implemented the lower three layers of commonsense thinking: reactive, reflective, and retrospective layers. Our cognitive models are affected by the layers and believed to be a new direction for cognitive robotics, focusing on human-robot interaction (HRI) and use of script knowledge. We have built a robot task script structure as a container for maintaining task-related information, such as pre-condition, post-condition, scene, action sequence, action type, and other contextual
information. Prior to designing the script structure, we handcrafted several variations of tasks according to changes of situations to ensure the extensibility of our script structure. Especially, our script structure is designed taking into consideration ambiguity and importance of the specific script in a specific situation.

Because we need a large number of robot task scripts that contain real situations associated with a full and varied human life, we have developed a user friendly script editor for easy script generation. Our goal is to allow even non-expert users can easily make their own robot task scripts by providing various authoring functionalities, such as selecting action sequences, browsing task scripts, and revising existing sub-scripts.

Through a semi-automatic generation method, with the help of the editing tool, it becomes more realistic to produce a large number of scripts. The tool automates the process of finding relevant scripts. In other words, it helps users with finding reusable scenes and sub-tasks from the script DB.

The next section reviews the previous work, and Section 3 describes our cognitive robot architecture in detail. The functionality and searching algorithm of our script editing environment are illustrated in Section 4 and 5, respectively, and the conclusion follows.

2. Related Work

Cognitive Robotics: The term “cognitive robotics” was first introduced by Ray Reiter et al. Cognitive robotics is concerned with endowing robots and software agents with higher level of cognitive functions that enable them to reason, act, and perceive in changing, incompletely known, and unpredictable environments [3]. That is, a cognitive robot can reason, perceive, and act in new environment intelligently through an interaction with human. Cognitive robotics must address several artificial intelligence methods, such as knowledge representation, automated reasoning, planning, etc.

If the robot has higher level intelligence enough to understand human needs, its behaviors can be diversified and more effective [4]. Scheutz et al. [5] introduced architecture for complex affective cognitive robots for human-robot interaction by emphasizing understanding of human’s emotion.

Considering needs and efforts of the previous approaches, we chose to borrow the EM-ONE’s commonsense thinking architecture as a basis of our cognitive robot architecture.

Script: A script is to represent a narrative composed of a series of linked sentences. Schank et al. [6] proposed using frame-like structures which record the normal sequence of events for a given type of occurrence [7]. In other words, the script is an event list. The script was originally used in Artificial Intelligence to represent stereotyped sequences of events. The script is to record patterns of the occurrence of events from the real world, and thus natural to represent robotic behaviors, making it useful for humanoid robots. The sequence of scripts will facilitate reasoning about unobserved events. For example, the script schemes were used in two navigation test domains; the indoor topological navigation and the search and rescue [8].

However, the script paradigm has been applied seldomly in cognitive robotics research area. For cognitive robotics, a cognitive script model suggested by Choo [9] can be one choice. His cognitive script is action sequences that represent knowledge about frequently experienced events. It can have influence on not only behavior, but also emotions. The emotional factors are also very important issues in cognitive robotics.

3. The Architecture

3.1. Cognitive Architecture for Human-Robot Interaction

Figure 1 shows the overall architecture that we considered for our robot task script structure in real robot’s task planning. The major three models are adopted from the EM-ONE architecture. The first one is the reactive model that contains a lot of knowledge scripts. Scripts may have various lengths and sizes. This reactive model is dominated by the deliberative and reflective models. The deliberative model is composed of the “cognitive and rule-based planner” and the “exceptional event solver” to handle ambiguity and exceptional situations arisen in real daily life. The planner infers next actions, and the solver handles exceptional events. The planner basically plays a role of hybrid planning that combines a rule-based planning and an experience planning. Rules come from the reflective model, and
experience from the scripts of reactive model. It means that the planner and the solver are under control of the reflective model. The highest model is reflection of past history and user’s cognitive knowledge structure due to the usages of contains rule, user profile, and ontology.

In these processes, both robot and human equally participate in completing the given task. In order to do it, our architecture includes two managers: “Task Manager” and “Interaction Manager”. The Task Manager usually concentrates on the surrounding context and what the task would be. It forecasts what task is appropriate for the current context information. In the middle of the process, they determine when the interaction has to be intervened. The interaction manager coordinates this communications between the robot and the human. Depending on the context, its urgency and importance may be different and its timing and communication direction are governed. The interaction manager is involved in facilitating collaboration between the human and the robot.

3.2. Resolving Ambiguity

We assume that human’s cognitive process can be mimicked moderately with the above three models in Figure 1. Based on the architecture, the robot task manager steps in and out the three components, regarding ambiguity. This ambiguity is determined by the following equations.

$$ \text{Ambiguity} = 1 - \left\{ \text{ratio}_1 \cdot \left( P(C_1, C_2, \ldots | A_1, A_2, \ldots) + \text{ratio}_2 \cdot \left( P(C_1, C_2, \ldots | A_1, A_2, \ldots) + \ldots + \text{ratio}_n \cdot P(C_1, C_2, \ldots | A_1, A_2, \ldots) \right) \right\} $$

Ambiguity measures how much following actions satisfy their corresponding pre-conditions and post-conditions. Therefore, it is evaluated by the sum of the possibilities to fulfill the specific conditions $C$, given that the scheduled actions, $A_i$, occur per scene. Ratio emphasizes important actions and is obtained from each scene’s basic information. If the ambiguity value is larger than a heuristically assigned threshold, the system requests user’s intervention. Through user’s intervention, the robot can obtain appropriate actions to be executed and can reduce ambiguity.

3.3. Resolving Exceptional Event

When an exceptional event occurs in the middle of executing a determined task, it presents a problem. There were some efforts to resolve exceptional events [9-11]. The most important consideration points that they dealt with are how to make the system to have an ability to be prioritized, interrupted, and resumed. They used different computing models, but all their models are evaluated for sudden events with how much they are important and urgent. In our resolver, we handle the problem, considering the interrupt cost which is assessed with importance and urgency as well. It is more helpful to provide adaptive and flexible solutions. Freed’s work [9] also contains an interrupt cost concept but we extend the influence of the “interrupt cost”. We assume that all decisions are made after considering the interrupt cost against continuing the previous task.

In our exceptional event solver, importance means how the work has dominance at the given specific situation. For example, wearing a tuxedo and a wedding dress is very important at a wedding ceremony. In contrast, we can hardly say that wearing training pants is of high importance in wedding ceremony. This importance is essential because, in an exceptional case, we have to find the best alternative as soon as possible. This is the reason why we should consider importance.

The meaning of urgency is how quickly the work should be performed in the given specific situation. For example, wearing a tuxedo and a wedding dress is very important at a wedding ceremony. Therefore, urgency can be considered as the remaining time expected to complete the task. It measures the degree to which the work has high urgency.

Our exceptional event resolver provides appropriate solutions regarding interrupt cost when external exceptions occur. Following formula shows how the importance and urgency measures contribute to finding adaptive and flexible solutions.

$$ \text{Interrupt Cost} = \alpha \cdot I(\text{scene}) + (1 - \alpha) \cdot U(\text{scene}) $$

$$ I(\text{scene}) = \sum_{\text{pre-condition}} \text{Impact factor} + \sum_{\text{post-condition}} \text{Impact factor} $$

$$ U(\text{scene}) = \text{the expected time available to complete the task} $$
3.4. Script Design based on the Cognitive Architecture

A script can be defined as a structured representation describing a sequence of actions and the environment in a particular context. In HRI, the robot must have some knowledge about the situation, actors, objects, and so on. To manage the knowledge systematically, a well-organized script model is essential.

Given these requirements, our proposed script structure has four properties; pre-condition, post-condition, environment, and scene (Figure 2). The pre-condition and the post-condition properties are the conditions for starting and ending the script, respectively. Each condition has a unique IDentifier (ID) and actor information. The environment property refers to the information surrounding the current situation. For example, if the robot is located in a kitchen and there exists cups, a refrigerator, a table, and so on, the location and the objects belong to the environment property. Each item in the environment property should have its unique ID.

The script structure consists of multiple numbers of scenes. Each scene is a presentation of the temporal aspect of a script. That is, it can be a wide scope action such as ‘going to refrigerator’, ‘bringing something’, and so on. Each scene has three properties: pre-condition, post-condition, and action. The pre-condition and the post-condition properties are the same as those explained above. Each scene can contain several actions as its children nodes (Figure 1).

An action is associated with a situation in the scene. Each action is based on the primitive action of the Conceptual Dependency (CD) theory developed by Schank et al. [6]. Based on the CD theory, a given situation can be classified as one of the standard action types, such as ATTRANS, PTRANS, PROPEL, MOVE, GRASP, etc. For example, if the scene is ‘to bring the beer can in the refrigerator’, the actions are ‘to open the refrigerator (PROPEL)’, ‘to pick up the beer can (GRASP)’, and ‘to close the refrigerator (PROPEL)’.

Table 1. Primitive Actions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvokePathPlanner</td>
<td>Execute path planner to get the shortest paths</td>
</tr>
<tr>
<td>Navigation</td>
<td>Move to the goal position according to the path list</td>
</tr>
<tr>
<td>GoRight</td>
<td>Go right</td>
</tr>
<tr>
<td>GoLeft</td>
<td>Go left</td>
</tr>
<tr>
<td>GoForward</td>
<td>Go forward</td>
</tr>
<tr>
<td>GoBackward</td>
<td>Go backward</td>
</tr>
<tr>
<td>Stop</td>
<td>Stop movement</td>
</tr>
<tr>
<td>Wait</td>
<td>Wait for a while</td>
</tr>
<tr>
<td>Grasp</td>
<td>Grasp</td>
</tr>
<tr>
<td>HandOver</td>
<td>Hand over the object</td>
</tr>
</tbody>
</table>

The action can also be split into several primitive-actions. A primitive-action is the smallest action unit that a robot can operate in some situation (action). Each primitive action is based on atomic action in Kim et al.’s work [13]. Among them, we choose some primitive actions needed in our approach as in Table 1. For example, if the action is ‘to grasp the beer can’, the primitive-actions are ‘to move arm’, ‘to grasp the can with the gripper’, and ‘to move arm’.

Our script, which will be stored as a case in the robot script database (DB), should have a dynamic, efficient, and rich data structure. For this reason, our script is defined by employing the concepts of scene, action, and primitive-action and using the representation capability of XML (Figure 3).

Using the script model, we can represent most of the situations and tasks HRI should handle in home-service environments. For example, for ‘want something to drink’ in Figure 3, the script starts with 2 scenes; ‘to determine which drink is wanted’ and ‘to find where the drink is’. Also, each scene could contain one or more actions, for instance ‘MTRAMS: to question and answer about which drink is wanted’ in the first scene. In addition, each action can be divided into multiple primitive-actions. For example, at the first action case, it could have four parts; ‘AT0301: to recognize user’s voice’, and ‘AT0305: to speech something’. The Figure 2 is a part of script model.

Let’s think about the above example with the proposed cognitive architecture. If the robot recognizes the fact that the owner wants something to drink, the robot can suggest a plausible solution by the reactive model as shown Figure 1. In the deliberative model, it assesses the hypothesis and does a planning according to the existing rules and experiences. For example, in the recent action, the owner...
wants something to drink. Then, it can find and adjust one or more scenes and corresponding actions which can help the owner solve the problem of being thirsty. Through the processes, a new script is derived to cope with the present situation more feasibly. In the reflective model, it investigates potential problems of the recent action suggested by the reactive and deliberative models. For example, although the owner wants to drink a cola in the suggested script, the owner has diabetic symptoms, and the robot recommends juice. The red dotted part of Figure 3 is a result of the reflective model’s reasoning.

Figure. 3 Example of script model

4. Scripting Environment

The purpose of developing a script editor is to help robotic system builders generate and manipulate scripts. In fact, people are very prone to writing scripts which are conflict each other in a context, or scripts which are syntactically invalid. To avoid this situation, the script editor provides the users with simple and sequential methods for script generation. Instead of writing a script from scratch, users can start their scripting with a suitable action list for the given situation recommended by the editor. In addition, it shows contextually similar scripts for a user to avoid making redundant scripts and to give scripting hints for the user. That is, the process of creating a new script can be more efficient by reusing previously built scripts with slight changes. As Figure 4 shows, action sequences and their situations can be edited in a graphical view. Currently, the editor is still being revised for stabilizing the above functionalities.

Figure. 4 Our Script Editor

5. Searching Scripts

Although, story-like scripts have a rich representation capability compared to other simple knowledge representation methods, there may be some difficulties in finding relevant ones from the script DB for a given situation. However, an elaborate search approach combined with the search part of Case-Based Reasoning (CBR) [14] and information retrieval (IR) techniques [15] can overcome the problem. It helps generating script semi-automatically and decreasing manual efforts for script coding remarkably by reusing the similar task scripts in the given context.

Figure. 5 Script Search Algorithm based on CBR+IR

For evaluation of the script search, 150 scripts are manually generated by three authors and tested against the searching mechanism. Scripts are indexed and searched using Apache Lucene [16]. For a given query node, top most k scripts (currently, k=10) are retrieved and re-ranked from the script DB using the CBR+IR approach (Figure 5). IR technique mostly involves finding similar
scripts based on appearance of keywords. Then, CBR reassigns relevance scores of retrieved scripts considering weights of each key feature, such as pre-condition, post-condition, length of script, node type, etc. Although calculating precision/recall is somewhat impetuous due to insufficient number of crafted scripts, we observed that retrieved results were quite feasible; Especially we could find very promising scripts within the 5 topmost results.

6. Conclusion

In this paper, cognitive HRI based script design issues were described in the context of the three cognitive models: reactive, deliberative, and reflective model. Our script design has started from the firm belief that a robot task planning system that considers cognitive HRI in a home-service environment would correspondingly be required to support composition of action sequences and their interactions. In particular, our script representation combines substantial task knowledge, contextual information, and cognitive considerations that are needed for robot’s task planning.

In addition, we have built a robot task script editor for easy and consistent generation of robot task scripts. Currently, it implements a retrieval capability that searches for similar scripts (or scenes) from the script DB. Its main functionality concentrates on finding the most relevant solutions that could handle given problems using CBR+IR techniques and helping users avoid redundancy and errors in creating scripts.

In future work, we will implement a total simulation environment which supports more sophisticated script search functionality and action sequence adaptation for goal completion. In addition, to check the real adoptability into dynamic situations of a daily life, an elaborate visual step-wise evaluation module is being constructed. In case of an error-prone script, it should be exterminated before putting into the robot task script knowledge base. For the evaluation and weeding-out process, the consistency checking over goal pursuit, task objectives, object characteristics, location constraints, and so on will be provided.

7. Acknowledgment

This research (paper) was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea

8. References