Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Modelling and Analysis of Natural Language Controlled Robotic Systems

Yu Cheng * **Yunyi Jia** * **Rui Fang** ** **Lanbo She** ** **Ning Xi** * **Joyce Chai** **

* *Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824, USA (Tel: 517-432-1925; e-mail: chengyu9@msu.edu, jiayunyi@msu.edu, xin@egr.msu.edu).*
** *Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA (Tel: 517-432-9239; e-mail: fangrui@cse.msu.edu, shelanbo@cse.msu.edu, jchai@cse.msu.edu).*

Abstract: Controlling a robotic system through natural language commands can provide great convenience for human users. Many researchers have investigated on high-level planning for natural language controlled robotic systems. Most of the methods designed the planning as open-loop processes and therefore cannot well handle unexpected events for realistic applications. In this paper, a closed-loop method is proposed for task-planning to overcome unexpected events occurred during implementation of assigned tasks. The designed system is modeled and theoretically proved to be able to stabilize the robotic system under unexpected events. Experimental results demonstrate effectiveness and advantages of the proposed method.

Keywords: natural language, robot, action scheduling, task planning, controllability, observability, Lyapunov stability.

1. INTRODUCTION

Instructing with unconstrained natural language is an intuitive and flexible way to interact with robots, which has attracted increasing attention for its obvious convenience. Firstly, people no longer need to get trained for how to control the robot with constrained language, which is not flexible and convenient at all for users. In addition, it is not necessary for users to get acquaint with the complicated mathematical model, any programming skills or even the graphical interfaces, which reduces users' cognitive loads. Furthermore, normally, robot operators have to keep an eye on their screens to monitor the performance of the robots, with their hands controlling a keyboard or joystick. While using natural language control to control robots, their eyes and hands are freed from monitoring and manipulation, reducing personnel requirements and fatigue. Natural language controlled robotic systems would greatly benefit people, especially for the old and disabled.

Currently, there exist two main challenges for natural language controlled robotic systems. One is to represent natural language commands in a formal representation understandable for the robots, since robots are not able to understand and process natural language commands directly. The other is to design a reliable and task-independent task planner for the robotic system which takes the language commands in formal representation transformed from natural language and produces an implementable action sequence that can lead to desired behaviors.

Research on natural language controlled robots mainly moves along two directions. One is to teach the robots new behaviours using natural language commands based on their prior knowledge (Rybski et al. 2007)(Yoon and Rybski 2007)(Chen and Mooney 2011)(Cantrell et al. 2012), sometimes with the help of human demonstrations. The other is to command the robots to accomplish assigned tasks with natural language directives (Barabas et al. 2012)(Duvallet et al. 2013). All these efforts involved in producing appropriate event trajectories in task planning layer based on given natural language input. It is a key step for the natural language controlled robots. The approaches to perform the action scheduling process can be classified into two types: probabilistic model and logic model.

Probabilistic models (Kollar et al. 2010)(Matuszek et al. 2012), provide possible action sequences for each natural language command. Useful features, such as predicates, subjects, locations and so forth, are extracted from natural language commands, and then are used to calculate the probability of correspondence between atomic actions and natural language commands. Finally the one with the highest likelihood is looked as the most appropriate interpretation for the language command.

Logic models (Kress-Gazit et al. 2008)(Kress-Gazit et al. 2009), use formal representations as the input to the task planner to drive the evolution of robotic systems. Language commands are first transformed into some formal representations (like linear temporal logic), and then a discrete controller for the robot is synthesized based on these formal formulas if no behaviours violate the given specifications.

All these proposed methods offer primary solutions to the realization of natural language controlled robots, and provide useful insights for subsequent research. However, for most of the proposed approaches, the task-planning processes is designed

⋆ This research work is partially supported under U.S. Army Research Office Contract No. W911NF-11-D-0001, and U.S. Army Research Office Grant No. W911NF-09-1-0321 and W911NF-10-1-0358, and National Science Foundation Award No. CNS-1320561 and IIS-1208390

as open-loop, which neither guarantee the performance of the robotic system to satisfy given task specifications, nor are able to tackle uncertainties happened during the implementation. In addition, these action-schedulers have not been proved to have the ability to work reliably in real applications.

In this paper we develop a closed-loop action scheduling method, which is capable of guaranteeing the robot's performance consistent with the natural language instructions, especially reasonably react to immediate changes of environment. We demonstrate the usefulness and feasibility of the proposed method by applying it to a natural language controlled robot with several simple but illustrative scenarios. This paper also reports the uncertainty encountered in some detail and tries to provide at least partial remedies for the action scheduling problem.

The rest of this paper is organized as follows. Section 2 presents the design for both the plant and controller. Section 3 analyzes the properties of the proposed action scheduling process. Section 4 provides experimental configurations and results. We close this paper with conclusions and future work.

2. MODELING NATURAL LANGUAGE CONTROL

In this section, we first explain how we use supervisory control framework (Ramadge and Wonham (1989)) to model the task planner for the natural language controlled robotic system, and then introduce the control strategy used to search for a shortest action plan for each given specific task.

Supervisory control is widely used in the domains of manufacturing system, vehicular traffic, computer and communication networks and so on. However, it is the first time to be used in the natural language controlled robotic system. The advantages will be illustrated in detail in Section 2 and 3.

2.1 Preliminaries and Robotic System Model

Supervisory control framework is an effective discrete extraction of a continuous process. It can be used to describe a system that evolves in accordance with external events at possibly unknown and irregular time. Additionally, it can be compositional, which enables the construction of complicated robot behaviours from simple ones. Furthermore, it has a close relationship with both language and control. Due to these inherent features of the framework, we use it to capture the subset of physical behaviours of the robotic system for task planning.

Here, we use a six-tuple automaton

$$G = \{Q, \Sigma, \delta, Q_0, Q_m, E_v\} \tag{1}$$

to model the robotic system, where

Q is a finite nonempty set of states abstracted from the robotic system, denoting the status of the plant.

Σ represents the discrete set of events, in which the elements drive the system to change from one state to another. The events of Σ can be classified into two parts: Σ_c in which events can either be enabled or disabled, and Σ_u where events are set to be always enabled by default. By continuously enabling actions from Σ, actions are implemented one by one and thus a state trajectory as well as an action sequence will be produced. Since the language generated by an automaton is comprised of elements from event set Σ, it is also called an alphabet.

$\delta : \Sigma \times Q \to Q$ is the transition function that captures conditional state changes.

Q_0 : elements of Q_0 denote the initial states from where a language or a system starts.

$Q_m \subset Q$: a subset of the state set Q, called the set of marker states. Usually, Q_m are used to represent the successfully completed tasks.

E_v is defined as the set of valid event trajectories, denoting the event sequences that are physically possible in G.

The behaviours of an automaton G can be described by the set of the output event trajectories, also called strings or languages, and G is called a language generator in this case.

Let Σ^* represents the set of all the finite strings s comprised of elements from Σ, including the empty string ε. The language $L(G)$ is the set of all event trajectories that are physically possible for the plant

$$L(G) = \{s : s \in \Sigma^*,\ \delta(q_0, s) \text{ is defined}, q_0 \in Q_0\}. \tag{2}$$

The marker language $L_m(G)$, describes all the event sequences that can reach the marker states

$$L_m(G) = \{s : s \in \Sigma^*,\ \delta(q_0, s) \in Q_m,\ q_0 \in Q_0\}. \tag{3}$$

In the following part, we present the behaviour models that capture the grasp-move-place manipulation of the plant and the design for the task-planner with a simple but illustrative example, to demonstrate how a satisfactory action trajectory can be generated to obey given task specifications. In this scenario, we consider a 7 DOF robot arm mounted on a mobile base in front of which there are four objects to be manipulated.

To implement the assigned tasks we need to know two types of information about the gripper: the open-close status and its current position. In order to describe the complete task, these two types of behaviours should be synchronized. The finite-state diagrams for the open-close status and positions of gripper are given in Fig. 1 and Fig. 2, respectively.

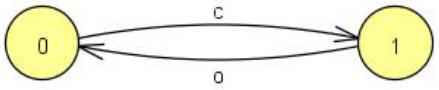

Fig. 1. Gripper status, 0 for open and 1 for close

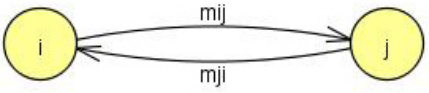

Fig. 2. Position of the gripper, $i, j \in \{0,1,2,3,4\}, i \neq j$

The synchronous composition of these two behaviours is shown in Fig. 3, where nodes represent the states of the plant, and the symbols over arcs denote the events from Σ. States of the plant are characterized with two bit numbers, where the first bit means the position of the gripper: 0 denotes a fixed initial position and 1$\sim$4 represent four flexible object positions. And the second bit means the status of the gripper, indicating whether or not there is an object in it. The event set: $\Sigma = \{m_{ij}, c, o, 0 \leq i, j \leq 4, i \neq j\}$, where c and o are actions of closing and opening

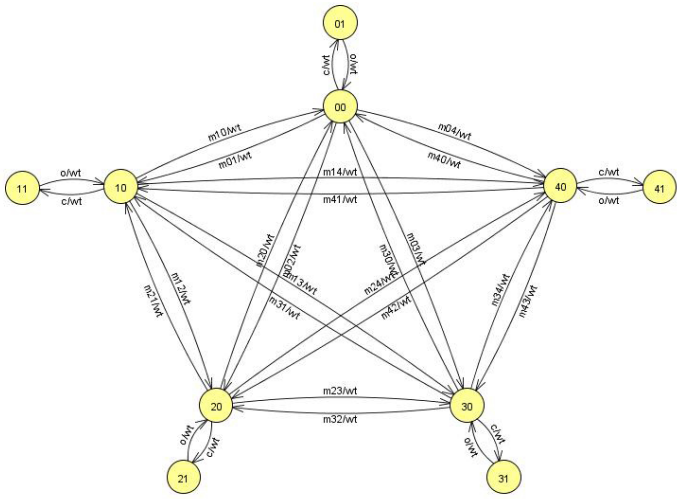

Fig. 3. The shuffle of the weighted product

gripper, separately, and m_{ij} denotes moving from position i to position j ($i \neq j$). The initial state is not limited to a fixed one, however, it can be any state of the finite state machine. This is also the case for the final state of the task. The marker set of states Q_m is comprised of all the states above, i.e. $Q_0 = Q_m = Q$. The state transition functions are shown in the synchronous finite-state diagram.

2.2 Controller Design

In Section 2.1 all legal behaviours in the task domain have been captured in the plant model. Our aim is to design a task planner (also discrete controller) capable of generating appropriate event sequences which satisfy the specific task requirements transformed from real-time natural language commands. Additionally, it should be able to stabilize the system where environment changes may cause current action plan no longer fitting the task requirements.

The working procedure is shown in Fig. 4. Natural language commands are firstly transformed into formal representation by the natural language processing module. In the meanwhile, the controller keeps updating its internal and external environment information. It takes formal representation and current status information about the robot as input and generates an action for continuous controller to control the robotic system. Then both the information of the robot and its working environment are dispatched to the action-scheduler for reference to produce the next action.

Our goal is not only to accomplish the desired task, but also to implement it with minimun transitions. With the given formal representation input, desired status can be determined, and with sensor feedback, the current status of the robot can be obtained. Now the problem is formulated as finding an action path with minimum moves among all the possible paths. In order to find the shortest action sequence, we assign each action a cost value, and the path that has the minimum cost would be regarded as the optimal action plan. There are plenty of developed shortest path algorithms suitable for our situations, such as Bellman-Ford, Floyd-Warshall, Dijkstra, and so forth. The weighted automaton is shown in Fig. 3.

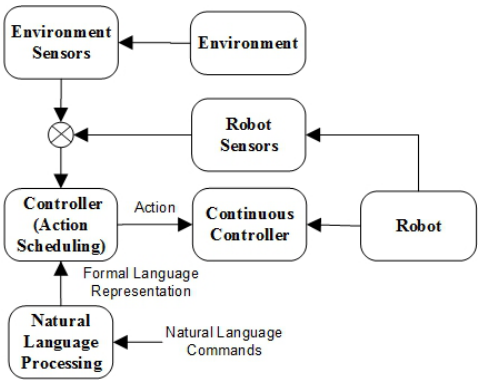

Fig. 4. Block diagram of control scheme

With the shortest path algorithm and real-time sensor information feedback, the robot can produce an action sequence with minimum moves, also, it is capable of reacting reasonably to abrupt changes during each execution. For instance, if the target is moved to another position before picking up manipulation, which leads to the failure of current action plan. The task planner is able to regenerate a new action plan for the changed environment layouts.

The number of minimum moves can be defined as

$$min = \inf\{n : \xi(s, q_0) = q_m, s = \sigma_1\sigma_2...\sigma_n, \sigma_i \in \Sigma\}. \quad (4)$$

And the corresponding shortest event path is defined as

$$s_{min} = \sigma_1\sigma_2...\sigma_{min}. \quad (5)$$

The language generated by the controlled robotic system can be represented as

$$L(C/G) = \{s_{min} : \xi(s_{min}, q_0) = q_m,\ q_0 \in Q_0, q_m \in Q_m, s_{min} \in \Sigma^*\}. \quad (6)$$

3. SYSTEM ANALYSIS

It is essential and necessary to analyze properties of the designed controller before real application. In this section, we analyze the properties of the controller from the perspective of control, and demonstrate the controller has several excellent properties which guarantee the robotic system's performance with respect to given task requirements.

3.1 Controllability of Robotic System

The main objective of our goal is to design an action trajectory generator (controller) that achieves overall controllability of a system. This controller is capable of enabling or disabling controllable events in order to ensure correct and desirable behaviours of the system. A system is controllable if any desired state of it can be reached based on current status and appropriate control strategy. The existence of a controller is guaranteed if all the desirable behaviours of the system are controllable (Ramadge and Wonham 1987)(Ramadge and Wonham 1989).

The closure of a language K, denoted by $\overline{K}$, is the set of all the prefixes of strings from language K, i.e., $\overline{K} = \{s : s \in \Sigma^*$ and $(\exists t)\ t \in \Sigma^*, st \in K\}$. Language K is said to be $(\Sigma_u, L)-$ invariant if $\overline{K}\Sigma_u \cap L \subset \overline{K}$. The controllability for a language can be defined as follows (Ramadge and Wonham 1987)(Ramadge and Wonham 1989).

Definition of Controllability: Let $K \subset \Sigma^*$, $L \subset \Sigma^*$ be two arbitrary languages generated by the automaton G. We say that K is controllable if $K \subset L(G)$ and K satisfies: $\overline{K}\Sigma_u \cap L \subset \overline{K}$.

In our case we set $K = L(C/G)$ in (6), which means the language generated by the plant under control, and $L \subset \Sigma^*$. Then we have:

$$L(C/G) \subset L(G).$$

In our language generator model, $\Sigma_u = \Sigma - \Sigma_c = \emptyset$. We have

$$\overline{K}\Sigma_u \cap L(G) = \emptyset \subset \overline{K}.$$

Then we conclude the language generated by (1) is controllable.

3.2 Observability of Robotic System

Observability is the basis for the control over robotic system. With the system observable, occurrences of events can be sensed and offer feedback information to the controller.

A task is said to be observable if the event sequence generated by the plant is observable. If the system fails to be observable, then some events produced by the plant are indistinguishable to the controller, which hampers the controller from tracking and controlling. Here is the definition of observability (Ramadge 1986)(Lin and Wonham 1988).

Definition of Observability: A language is said to be observable if it satisfies the condition that there exists a right congruence γ such that ker $P \leq \gamma \leq act_K$.

Here, both ker P and act_K are right congruences (Lin and Wonham 1988). A right congruence is an equivalence relation θ on Σ^* and satisfies $s \equiv s' \pmod{\theta} \Rightarrow st \equiv s't \pmod{\theta}$.

Then an analysis is provided to show the observability of controlled language generated by the plant.

Firstly, we define a right congruence $eq(G)$ on Σ^*:

$$s \equiv s' (\text{mod } eq(G)) \quad \text{iff} \quad \delta(s,q_0) = \delta(s',q_0)$$

To show that the left inequality holds:

Let $\gamma = eq(S)$, where S is a proper controller for (1) (Ramadge and Wonham 1987). Since for $S = \{\Sigma, X, \xi, X_0, X_m, E_{vx}\}$, and ξ is Σ_{uo} − null(Lin and Wonham 1988), then we can have:

$$\text{ker } P \leq \gamma.$$

To show that the right inequality holds:
1) We claim that $A_K(s) \cap IA_K(s') = \emptyset$. Otherwise $\exists \sigma \in \Sigma$ that $s\sigma \in \overline{K} \wedge s' \in \overline{K} \wedge s'\sigma \in L(G) - \overline{K}$. Then we can have:

$$\begin{aligned} &\xi(s\sigma, x_0) \neq \xi(s'\sigma, x_0) \\ \Rightarrow &\xi(\sigma, \xi(s, x_0)) \neq \xi(\sigma, \xi(s', x_0)), \end{aligned}$$

while,

$$x' = \xi(s, x_0) = \xi(s', x_0),$$

we can have:

$$\xi(s, x') \neq \xi(s', x').$$

Because $s \equiv s' \pmod{eq(G)} \Rightarrow \xi(s,x') = \xi(s',x)$, a contradiction. And similarly, we can obtain

$$A_K(s') \cap IA_K(s) = \emptyset.$$

2) If $s \in \overline{K} \cap L_m(G) \wedge s' \in \overline{K} \cap L_m(G)$, then $s' \in \overline{K} \subseteq L(S/G)$, and therefore $\delta(s', q_0)!$ and $\xi(s', q_0)!$. And then:

$$\left.\begin{aligned} &s \in k = L_m(S/G) \\ &x = \xi(s, x_0) = \xi(s', x_0) \in X_m \\ &s' \in L_m(G) \Rightarrow \delta(s', x_0) \in x_0 \end{aligned}\right\} \Rightarrow s' \in k$$

Similarly, with $s' \in k$ and $s \in \overline{K} \cap L_m(G) \wedge s' \in \overline{K} \cap L_m(G)$ we can get $s \in K$.

Thus we have $\gamma \leq act_K$. Language K is observable.

Lin and Wonham's work provides a theorem (Lin and Wonham 1988) that if the language is both controllable and observable, then properties like completeness, nonblockingness and nonrejectingness (Ramadge and Wonham 1987) are all guaranteed automatically.

3.3 Stability of Natural Language Controlled Robot

Characteristics of an automaton are the inherent nonlinearity and asynchronous occurrences of events. Attempting to control such a real-time dynamic system introduces the stability of the discrete event system. The work by Passino et al. (Passino et al. 1994) provides the foundation for analyzing the stability as well as asymptotical stability in the sense of Lyapunov for system modelled by automata.

An event sequence uniquely leads to a resulting state trajectory with applying a specified sequence of δ. The set of possible system states is limited by both physical and task specifications. Those constraints on the admissible state space can be realized by appropriate event trajectories. Thus a measure is needed to determine whether or not the resulting state is limited to a well-defined set (invariant set), which has no violation of any imposed physical or task specifications, and is capable of expressing boundedness of the state space by its values. A metric on X is represented by $\rho : X \times X \to M$ and $\{X, \rho\}$ denotes a metric space. The distance from point x to a set X_z is defined as $\rho(x, X_z) = \inf\{\rho(x,x') : x' \in X_z\}$. A functional is defined as a mapping from an arbitrary set to the metric space M. The set $S(X_z; r) = \{x \in X : 0 < \rho(x, X_z) < r, r > 0\}$ denotes the r-neighbourhood of an arbitrary set $X_z \subseteq X$. Let E_k represents an event sequence containing k-occurred events where $k \in N$, and E_a denotes the set of allowed event trajectories where $E_a \subseteq E_v$ of (1).

Definition of Lyapunov Stability: A closed invariant set $X_m \subseteq X$ of G is called stable in the sense of Lyapunov w.r.t. E_a if for arbitrary $\varepsilon > 0$, a quantity $\delta > 0$ can be found for any E_k such that when $\rho(x_0, X_m) < \delta$ we have $\rho(X(x_0, E_k, k), X_m) < \varepsilon$ for all E_k that $E_k \in E_a(x_0)$ and $k \in N$.

Theorem 1. (Passino et al. 1994): A closed invariant set $X_m \subseteq X$ of G is said to be asymptotically stable in the sense of Lyapunov w.r.t. E_a, in a sufficiently small neighbourhood $S(X_m; r)$ of the set X_m, if it is necessary and sufficient to find a functional which satisfies the following four conditions:

1) For an arbitrary small $c_1 > 0$, a $c_2 > 0$ can be found in such a way that $V(x) > c_2$ for $x \in S(X_m; r)$ and $\rho(x, X_m) > c_1$.
2) For an arbitrary small $c_3 > 0$, a $c_4 > 0$ can be determined such that $V(x) \leq c_4$, $\forall x \in S(X_m; r)$, $\rho(x, X_m) < c_3$.
3) $V(X(x_0, E_k, k))$ is a nonincreasing function if for $k \in N$, $x_0 \in S(X_m; r)$, $\forall k \in N$, $X(x_0, E_k, k) \in S(X_m; r)$ and $\forall E_k$, $E_k E \in E_a(x_0)$.
4) For all E_k that $E_k E \in E_a(x_0)$ and for all $k \in N$ that $X(x_0, E_k, k) \in S(X_m; r)$, as $k \to \infty$, $V(X(x_0, E_k, k)) \to 0$.

In our case, the concepts for stability change a little bit. Here asymptotical stability means that the robotic system can reach any given target state q_m rather than a set from any initial state q_0 in Q_0.

From the definition and theorem 1 we could get that the system is neither stable nor asymptotically stable in the sense of Lyapunov without control. Since it is both controllabe and observable, the system can be stabilized by some control strategies. With the proposed control method, the system can achieve asymptotical stability.

First we define the distance function:

$$\rho(q,q_m) = \inf\{n : \xi(s,q) = q_m, s = \sigma_1\sigma_2...\sigma_n, \sigma_i \in \Sigma\} \quad (7)$$

where $q_0 \in Q_0$, $q_m \in Q_m$.

Thus we have:

$$\rho(q,q') = 0 \quad \text{if} \quad q = q',$$

Choose the Lyapunov functional

$$V(q) = \rho(q,q_m).$$

We can set a random state of the plant as initial state q_0, and another arbitrary state as the target state q_m. Take $r = 4$.

1) We can choose $c_1 = c_2$ as small as possible, Since $V(q) = \rho(q,q_m)$, for $\rho(q,q_m) > c_1$, we can always have $V(q) > c_2$.
2) Choose $c_3 = c_4$ as small as possible, then, for $\rho(q,q_m) < c_3$, we can always have $V(q) < c_4$.
3) and 4) Any state of the plant can be set to be initial state or target state. The control strategy, i.e., the shortest path algorithm, minimizes the Lyapunov functional and makes the system more robust to external exceptions, which are similar to disturbances in continuous system. With the algorithm, the generated formal languages, or the action sequences, are represented as in (6). In each situation, the generated event sequence will always have minimum moves. If some errors or unexpected environment changes occur which lead to the following state different than the expected one, the controller is capable of regenerating a new event trajectory based on the goal and the current state to satisfy the task requirements. Thus, $V(X(q_0,E_k,k))$ is a decreasing functional for $k \in N$, and $V(X(q_0,E_k,k)) \to 0$ as $k \to \infty$ (in our case it only needs finite steps).

The robotic system achieves asymptotical stability under the control strategy, which implies the robust performance of the robotic system is consistent with task requirements.

4. EXPERIMENT RESULTS

The experiment setup is configured as described in Section 2. A flat table is placed in front of the mobile base on which a 7 DOF robot arm is mounted, and a camera is mounted right under the arm to offer object recognition with each object position. Then the visual information is dispatched to the robot and processed by MOPED: a real-time object recognition software run under ROS (robot operating system). On the table we have four objects: three wooden blocks with different sizes, colours and surfaces and an empty plastic bottle, which are all suitable in size for a gripper to manipulate. The experiment platform is shown in Fig. 5.

As Fig. 4 shows, at first a human operator gives the robot a natural language command, which then is transformed into formal representation. In our experiments, the work of natural language recognition is done by Dragon Speech Recognition software and transformed into formal representation by the collaborative effort from a research group of computer scientists. Then based on the given target state and the sensed current state from state feedback, the controller is able to produce an optimal event trajectory for the following step by step implementation. If some exceptions occur, a replanning process is invoked. It can be easily seen that with the same task specifications, generated action plans may vary a lot with different environment arrangements and uncertainties encountered during implementation.

The experiment platform is shown below. Here are two example

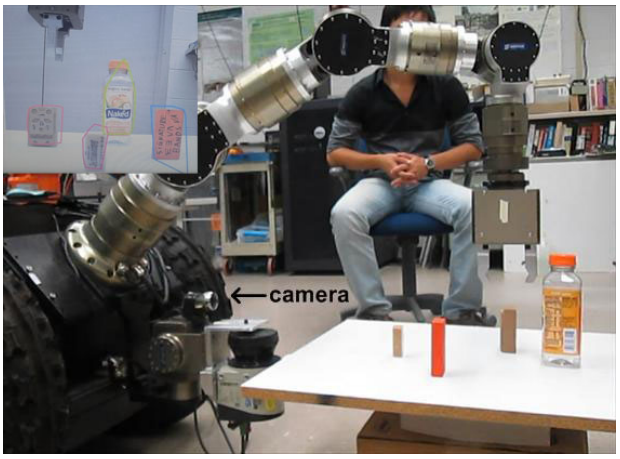

Fig. 5. Experiment platform

scenarios for our proposed method. For the first one, the given natural language instruction is "Take the small brown block." If the gripper is idle, with the same command, the action sequence would only include moving the gripper to the target position and next moving down to grab the desired object. While now the red block is in the gripper, i.e. the initial state is (31), so firstly the gripper has to put down the object in it, and then moves to the target position and picks up the desired object. After speech recognition and natural language processing, the target state (21) is obtained and sent to the controller for action scheduling. With state feedback we can obtain current state (31). Then the event trajectory is generated as: $o \to m_{32} \to c$, and the corresponding state sequence is: $(31) \to (30) \to (20) \to (21)$. The execution process is shown in Fig. 6.

If an uncertainty occurs during the implementation phase, for instance, in scenario 2, the directive given is "Grasp the big brown block", with the gripper idle as initial status. The final target state is (11). And the action sequence with initial state (00) is $m_{01} \to c$, during the process of approaching the big brown block, the positions of the target and red block are switched, leading to the failure of current action plan. Based on the feedback keeping updating, we can get current state. And the controller will regenerate a new action plan: $m_{13} \to c$, to lead the system to accomplish the assigned goal. Fig. 7 tells the whole process. Experiment results are shown in Table 1.

The experimental results demonstrate two advantages of proposed method. Firstly, the task-planner of robotic system generates action plans for a class of tasks rather than for a specific task, i.e. the designed controller is task-independent. In addition, with the same task requirements, the task-planner is able to make reasonable response to different initial environment arrangements and layouts. Also, it can stabilize the robotic system under uncertain events to guarantee system's performance.

Experiments	Initial Status	Commands	Exception	Action Sequence	Final Status
Exp1	Big brown block in gripper	Take that small brown block	No	$o \rightarrow m_{32} \rightarrow c$	Small brown block in gripper
Exp2	Gripper is idle	Grasp the big brown block	Positions switched	$m_{01} \rightarrow m_{13} \rightarrow c$	Big brown block in gripper

Table 1. Experimental results

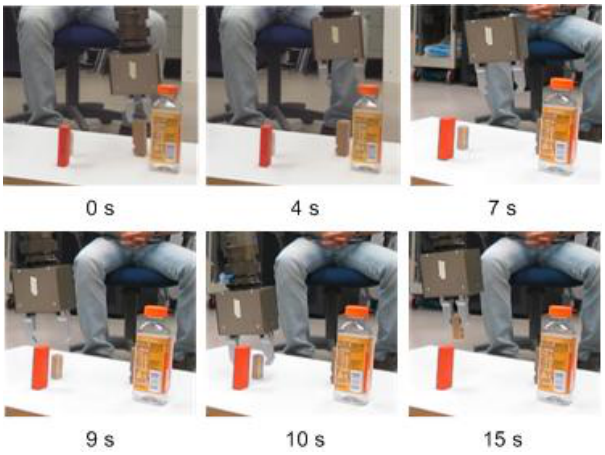

Fig. 6. "Take that small brown block"

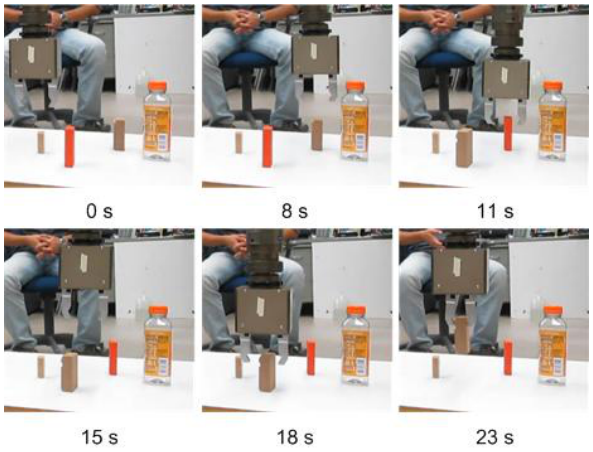

Fig. 7. "Grasp the big brown block"

5. CONCLUSIONS

Most task planners for natural language controlled robotic systems are designed as open-loop, and therefore have no capability to guarantee the performance of the system to meet given task specifications under unexpected events.

In this paper, we develop a method to model the behaviours of robotic system with supervisory control framework combined with shortest path algorithm of discrete level. Property analysis of the system demonstrate the advantage of the method and great potential for real application. The effectiveness and efficiency of the method are illustrated by experimental results.

For the future work, we are going to integrate information from force and tactile sensors into current robotic behaviour model. The new model is expected to be more powerful in guiding the robot to do more complicated and subtle tasks.

REFERENCES

Barabas, P., Kovacs, L., and Vircikova, M. (2012). Robot controlling in natural language. In *Cognitive Infocommunications (CogInfoCom), 2012 IEEE 3rd International Conference on*, 181–186. IEEE.

Cantrell, R., Talamadupula, K., Schermerhorn, P., Benton, J., Kambhampati, S., and Scheutz, M. (2012). Tell me when and why to do it! run-time planner model updates via natural language instruction. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, 471–478. IEEE.

Chen, D.L. and Mooney, R.J. (2011). Learning to interpret natural language navigation instructions from observations. In *AAAI*, volume 2, 1–2.

Duvallet, F., Kollar, T., and Stentz, A. (2013). Imitation learning for natural language direction following through unknown environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 1047–1053. IEEE.

Kollar, T., Tellex, S., Roy, D., and Roy, N. (2010). Toward understanding natural language directions. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, 259–266. IEEE.

Kress-Gazit, H., Fainekos, G.E., and Pappas, G.J. (2008). Translating structured english to robot controllers. *Advanced Robotics*, 22(12), 1343–1359.

Kress-Gazit, H., Fainekos, G.E., and Pappas, G.J. (2009). Temporal-logic-based reactive mission and motion planning. *Robotics, IEEE Transactions on*, 25(6), 1370–1381.

Lin, F. and Wonham, W.M. (1988). On observability of discrete-event systems. *Information sciences*, 44(3), 173–198.

Matuszek, C., Herbst, E., Zettlemoyer, L., and Fox, D. (2012). Learning to parse natural language commands to a robot control system. In *Proc. of the 13th Intl Symposium on Experimental Robotics (ISER)*.

Passino, K.M., Michel, A.N., and Antsaklis, P.J. (1994). Lyapunov stability of a class of discrete event systems. *Automatic Control, IEEE Transactions on*, 39(2), 269–279.

Ramadge, P.J. (1986). Observability of discrete event systems. In *Decision and Control, 1986 25th IEEE Conference on*, volume 25, 1108–1112. IEEE.

Ramadge, P.J. and Wonham, W.M. (1987). Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1), 206–230.

Ramadge, P.J. and Wonham, W.M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 81–98.

Rybski, P.E., Yoon, K., Stolarz, J., and Veloso, M.M. (2007). Interactive robot task training through dialog and demonstration. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, 49–56. IEEE.

Yoon, K. and Rybski, P.E. (2007). Teaching procedural flow through dialog and demonstration. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 807–814. IEEE.