From Structured Task Instructions to Robot Task Plans

Jianmin Ji and Xiaoping Chen
Multi-Agent Systems Lab, School of Computer Science and Technology
University of Science and Technology of China, 230026, Hefei, China

Keywords: Causal Theory, Open Knowledge, Task Planning, Task Instruction, Robot.

Abstract: For the purpose of allowing an autonomous robot to use task instructions for task planning, we present a formalization for specifying structured task instructions and provide an approach for integrating these instructions with robot's built-in knowledge to compute plans for open-ended tasks. We have implemented a prototype of the system. We also report a case study of the effectiveness of the approach.

1 INTRODUCTION

To perform everyday manipulation tasks in households, a service robot needs to automatically generate sequences of primary actions adapted to the environment and tasks, which is commonly done using AI task planning methods (McDermott, 1992), like Causal Theories (McCain and Turner, 1997), FLUX (Thielscher, 2005), or various planning systems based on PDDL (Ghallab et al., 1998). Originally, these approaches heavily rely on having a complete and certain description of the situation that the robot is faced with, which normally oversimplifies its real counterpart when the robot performs in the daily world (McDermott, 1992): Actions might have stochastic effects, the current state of the world might not be fully known, and action sequences are not expressive enough to specify competent robot manipulation behaviors. To overcome these challenges, Continual Planning (Brenner and Nebel, 2009) is proposed such that robots perform a continuous loop between planning, plan execution, and execution monitoring, where "classical" planning can be used for the most likely situations while the discrepancies between "physical reality" and "mental reality" can later be recovered during the execution monitoring phase. AI planning methods can also be integrated with computer simulations to enable robots to realistically predict real-world behaviors (Johnston and Williams, 2008).

On the other hand, everyday manipulation tasks are often open-ended, which requires the robot to infer by itself how to accomplish these tasks properly. One way to meet this requirement is to provide robots with the ability to understand and follow the task instructions that are defined by human users (Burgard et al., 1999; Rybski et al., 2007; Tenorth et al., 2010; Cantrell et al., 2012). A robot can find and extract knowledge of these instructions from human-robot dialogue, as well as web sources such as open knowledge bases like Cyc[1] and Open Mind Indoor Common Sense (OMICS)[2]. Then, after necessary conversions, the structured task instructions are represented in the robot and made use of to accomplish open-ended tasks.

Continuing both lines of research, we consider how to compute a robot executable plan for a task in an AI planning system following structured task instructions. In particular, we present a formalization for specifying structured instructions in a robot and provide an approach for integrating instructions with robot's built-in knowledge to compute plans for tasks.

In the authors' observation, for task planning, user-defined task instructions separate themselves from high-level task specifications written by designers in two major aspects. Firstly, two different types of knowledge, named functional and procedural knowledge, for task accomplishment are arbitrarily introduced in user-defined instructions and the robot may need to integrate both types of knowledge to compute plans for some tasks. Specifically, functional knowledge specifies the expected effects of a task. Given functional knowledge, the robot could consider these expected effects as goals and use AI planning approaches to compute a plan (Galindo et al., 2008). Procedural knowledge specifies a plan skeleton of how to accomplish a task. Given proce-

[1] http://www.cyc.com/
[2] http://commons.media.mit.edu/en/

Ji J. and Chen X..
From Structured Task Instructions to Robot Task Plans.
DOI: 10.5220/0004545402370244
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2013), pages 237-244
ISBN: 978-989-8565-81-5

dural knowledge, the robot needs to find out a possible sequence of actions which meets the restrictions of the plan skeleton to accomplish the task (Burgard et al., 1999; Tenorth et al., 2010). Normally, both knowledge are arbitrarily introduced in user-defined task instructions. For example, the task "heat food in microwave" can be instructed to take the following three steps: "place the food in microwave", "turn on microwave oven for two minutes", "take the food out". Note that, this instruction denotes some procedural knowledge, while the first and the third step denotes some functional knowledge, whose expected effects are "the food is placed inside the microwave" and "the food is placed outside the microwave" separately.

Secondly, user-defined task instructions may miss some details that are indispensable for robots' execution. For example, in OMICS, there is an instruction specifying how to "get food from refrigerator" in five steps. The last two steps were specified as "pick food up" and "close door". However, if the robot has only one arm, it must put the food somewhere before closing the door. This action is missing from the instruction.

We address these two aspects in this paper. We present an internal representation of both functional and procedural knowledge of task instructions and propose a unified framework to integrate the both with robot's built-in knowledge to compute plans for open-ended tasks. The planning program could also automatically detect the missing details, and add some proper actions to make the computed plan executable. A prototype of the system has been implemented based on Answer Set Programming (ASP), a logic programming language with Prolog-like syntax under stable model semantics (Baral, 2003). We choose ASP to implement the system because it can be use to specify causal theories easily and it has many efficient solvers. We also report a case study of the effectiveness of the approach.

A number of works in task planning have tackled the challenge of understanding and utilizing task instructions. (Gupta and Hennacy, 2005) presents a system which can fill-in missing information from task instructions using commonsense reasoning. (Kress-Gazit et al., 2007) specifies how to translate robot behaviors from a description in structured English to actual robot controllers via Linear Temporal Logic. (Cantrell et al., 2012) demonstrates a robot planner that can utilize natural language commands from untrained users. However, in these works, the internal representations of task instructions could not handle both functional and procedural knowledge, and the planning system could not fill-in missing steps between two successive operations in the instruction. Other related work (Tellex et al., 2011; Chen and Mooney, 2011; MacMahon et al., 2006; Tenbrink et al., 2010) focus on benefiting route instructions for robot navigation. Golog and its variants (Levesque et al., 1997; Beetz et al., 2010; Son et al., 2001) can represent and reason with both functional and procedural knowledge. However, they cannot reason about missing details that are indispensable for robots' execution.

The rest of the paper is organized as follows. Section 2 reviews a formalization of a task planning system. Section 3 reports a formalization for representing functional and procedural knowledge of task instructions. Section 4 provides an unified framework to integrate both knowledge with robot's built-in knowledge to compute plans, and describes an implementation of the system. At last, a case study is reported in Section 5 and conclusions are given in Section 6.

2 A TASK PLANNING SYSTEM BASED ON CAUSAL THEORIES

Without loss of generality, we choose the planning approach based on McCain and Turner's causal theories (McCain and Turner, 1997) for robot task planning. The idea is to specify the action domain and the planning problem in causal theories such that a causal model corresponds to a planning solution, then use sophisticated AI tools or solvers to compute robot plans for tasks. In the following, we briefly review the definition of causal theories, describe how to specify the action domain and the planning problem in causal theories, and sketch an implementation of the task planning system.

Firstly, the underlying propositional signature is consisted with three pairwise-disjoint sets: a set of *action* names, a nonempty set of *fluent* names, and a nonempty set of *time* names. The action-atoms are expressions of the form a_t and the fluent-atoms are expressions of the form f_t, where a, f, and t are action, fluent, and time names, respectively. Atoms of the language are either action-atoms or fluent-atoms. Specially, $\perp$ denotes contradiction. Intuitively, a_t is true iff the action a occurs at time t, and f_t is true iff the fluent f holds at time t. For example, $grasp(bottle)_1$ is an action-atom stands that the action "grasp bottle" occurs at time 1 and $holding(bottle)_2$ is a fluent-atom stands that the fluent "holding bottle" is true at time 2. A literal is either an atom a or the negation $\neg a$. Formulas are formed from atoms using propositional connectives, while fluent-formulas are

formed from fluent-atoms.

A *causal theory* is a set of *causal rules* of the form: $\phi \Rightarrow \varphi$, where ϕ and φ are formulas. Intuitively, the causal rule reads as "φ is caused if ϕ is true". As a syntax sugar, a causal rule with variables is viewed as shorthand for the set of its ground instances, that is, for the result of substituting corresponding variable-free terms for variables in all possible ways.

An *interpretation* I is a set of literals such that for each atom a in the language, either $a \in I$ or $\neg a \in I$ but not both. Given a causal theory T and an interpretation I, the *reduction* $T^I = \{\varphi \mid$ for some $\phi \Rightarrow \varphi \in T$ and $I \models \phi\}$. T^I is a propositional theory. We say that I is a *causal model* of T if I is the unique model of T^I.

An action domain contains the knowledge of actions of the robot and changes of the environment, which is an essential part of robots' built-in knowledge. A causal theory for an action domain will typically contain rules specifying the initial state and how fluents are changed as the result of performing an action. We take the action *grasp* and corresponding fluents for instance.

- $grasp(X)$: the action of griping the object X and picking it up.
- $holding(X)$: the fluent that the object X is held in the grip of the robot.
- $on(X,Y)$: the fluent that the object X is on the object Y.

In addition, σ is a meta-variable ranging over $\{on(X,Y), \neg on(X,Y), holding(X), \neg holding(X)\}$.

The effect of executing the action $grasp(X)$ is described as follows:

$$grasp(X)_t \Rightarrow holding(X)_{t+1} \quad (1)$$

$$grasp(X)_t \wedge on(X,Y)_t \Rightarrow \neg on(X,Y)_{t+1} \quad (2)$$

The precondition of grasping requires the grip holds nothing:

$$grasp(X)_t \wedge holding(Y)_t \Rightarrow \bot \quad (3)$$

The occurrence of the action is exogenous to the causal theory:

$$grasp(X)_t \Rightarrow grasp(X)_t \quad (4)$$

$$\neg grasp(X)_t \Rightarrow \neg grasp(X)_t \quad (5)$$

The initial state (at time 0) can be arbitrary:

$$\sigma_0 \Rightarrow \sigma_0 \quad (6)$$

The frame problem is overcome by the following "inertia" rules:

$$\sigma_t \wedge \sigma_{t+1} \Rightarrow \sigma_{t+1} \quad (7)$$

We define a *state* s for time t as a set of fluent-atoms with the time name t. Intuitively, s denotes a world specified by the fluents that are true at a time step. Given a causal theory with time names $\{0,1,\ldots,n\}$, we can define a *trajectory* as a sequence $\langle s_0, \alpha_0, s_1, \ldots, \alpha_{n-1}, s_n\rangle$, where s_i is a state for time i ($0 \leq i \leq n$), and α_j is an action-atom ($0 \leq j < n$). Note that, a causal model of such a causal theory contains exactly a trajectory of the above form, i.e., $s_0 \cup \{\alpha_0\} \cup \cdots \cup \{\alpha_{n-1}\} \cup s_n$ is a causal model of such a causal theory. We also call it a trajectory of the causal theory.

Given a description of the goals to be completed, we could use a fluent-formula φ_n formed by fluent-atoms with the last time name n to specify the requirements of the goal states. Then the task planning problem is reduced to the causal theory for the action domain by adding the causal rule $\neg\varphi_n \Rightarrow \bot$. Clearly, a causal model or a trajectory of the causal theory corresponds to a solution of the planning problem.

A task planning system based on causal theories has been implemented on our service robot (Chen et al., 2010; Chen et al., 2012). A logic programming language named Answer Set Programming (ASP) (Baral, 2003) is chosen for the calculation of causal theories and an efficient ASP solver *iclingo* (Gebser et al., 2008) is used for computing task plans. More details can be found in our robot's description paper[3].

3 A FORMALIZATION FOR STRUCTURED TASK INSTRUCTIONS

Both functional and procedural knowledge are commonly used in everyday task instructions. For example, an instruction written by web users can be found in OMICS:

> Heat bread in microwave involves the steps:
>
> 1. place the bread in the microwave;
> 2. turn on microwave oven for two minutes;
> 3. check temperature;
> 4. repeat till desired temperature is achieved.

Some steps, like "turn on microwave oven for two minutes", could be directly mapped to primitive actions which denote operating routines of the robot. Others may denote sub-tasks whose instructions could be specified by other functional or procedural

[3] http://ai.ustc.edu.cn/en/robocup/atHome/files/WEHome2013TDP.pdf

knowledge. For instance, "place the bread in the microwave" is also a task, whose expected effect is "the bread is placed in the microwave".

There has been some work on translating natural language sentences into logical representations (Baral et al., 2011) and handling the symbol grounding problem (Vogt, 2006). Our previous work (Xie et al., 2012; Chen et al., 2012) also provided an approach on converting semi-structured natural language instructions to structured representations. Here we focus on how to represent both functional and procedural knowledge of structured task instructions.

Firstly, the underlying signature is consisted with three pairwise-disjoint sets: a set of *action* names, a set of *fluent* names, and a set of *task* names. A *fluent-specification* is formed from fluent names using propositional connectives. A *procedure* is defined recursively as follows:

- an action name a is a procedure,
- a fluent-specification F is a procedure,
- a task name T is a procedure,
- if P_i $(1 \leq i \leq m)$ are procedures then $P_1; \ldots; P_m$ is a procedure,
- if P_i $(1 \leq i \leq m)$ are procedures then $P_1 | \cdots | P_m$ is a procedure,
- if P_1 and P_2 are procedures and F is a fluent-specification then **if** F **then** P_1 **else** P_2 is a procedure,
- if P is a procedure and F is a fluent-specification then **while** F **do** P is a procedure.

Intuitively, a fluent-specification represents a situation of the world and a procedure represents a skeleton of an execution program. For example, $P_1; \ldots; P_m$ represents a procedure executed from P_1 to P_m step by step and $P_1 | \cdots | P_m$ represents a nondeterministic choice from $P_1, \ldots, P_m$.

A piece of *functional knowledge* is a pair (T, F) where T is a task name and F is a fluent-specification and a piece of *procedural knowledge* is a pair (T, P) where P is a procedure. At last, we formalize a *structured task instruction* as a pair $(\mathcal{F}, \mathcal{P})$, where $\mathcal{F}$ is a set of functional knowledge and $\mathcal{P}$ is a set of procedural knowledge.

Continuing the example in the beginning of the section, a structured task instruction $(\mathcal{F}, \mathcal{P})$[4] could be extracted from the steps as

$$\mathcal{F} = \{ (Place(bread, micro), in(bread, micro)) \},$$
$$\mathcal{P} = \{ (Heat(bread, micro), P) \},$$

where P is the procedure:

$$Place(bread, micro); turnon(micro); check(bread);$$
$$\textbf{while} \neg isok(bread)$$
$$\textbf{do} \ (Place(bread, micro); turnon(micro); check(bread)),$$

$Heat(bread, micro)$ and $Place(bread, micro)$ are task names as they intimated, $in(bread, micro)$ and $isok(bread)$ are fluent names, and others are action names.

Given a robot, the structured task instruction collects its both functional and procedural knowledge of how to accomplish tasks. Intuitively, a piece of functional knowledge (T, F) means the task T would be accomplished when the fluent-specification F is satisfied at some time point, a piece of procedural knowledge (T, P) means that T would be accomplished when the robot has performed the procedure P. Now we make it precise in the task planning system based on causal theories.

Let Π be a causal theory specifying the action model of the robot and $\tau = \langle s_0, \alpha_0, s_1, \ldots, \alpha_{n-1}, s_n \rangle$ be a trajectory of Π, we define τ *satisfies* a fluent-specification F if for some $0 \leq i \leq n$, $s_i \models F_i$ where F_i is the fluent-formula obtained from F by replacing each occurred fluent name f by the fluent-atom f_i. τ *satisfies* a procedure P is defined recursively as follows:

- If $P = a$, where a is an action name, then a is the action name occurred in the action-atom α_0;
- If $P = F$, where F is a fluent-specification, then $s_0 \models F_0$;
- If $P = T$, where T is a task name, then τ satisfies some procedural knowledge P' of T or some functional knowledge F' of T;
- If $P = P_1; \ldots; P_m$, where P_i $(1 \leq i \leq m)$ are procedures, then there exists $0 \leq n^1 \leq n^2 \leq \cdots \leq n^{m-1} \leq n$ such that:
  - the trajectory $\langle s_0, \alpha_0, \ldots, s_{n^1} \rangle$ satisfies P_1;
  - the trajectory $\langle s_{n^1}, \alpha_{n^1}, \ldots, s_{n^2} \rangle$ satisfies P_2;
    $\ldots$
  - the trajectory $\langle s_{n^{m-1}}, \alpha_{n^{m-1}}, \ldots, s_n \rangle$ satisfies P_n;
- If $P = P_1 | \cdots | P_m$, where P_i $(1 \leq i \leq m)$ are procedures, then τ satisfies P_j for some $1 \leq j \leq m$;
- If $P =$ **if** F **then** P_1 **else** P_2, where P_1 and P_2 are procedures and F is a fluent-specification, then τ satisfies F implies τ satisfies P_1 otherwise τ satisfies P_2.
- If $P =$ **while** F **do** P_1, where P_1 is a procedure and F is a fluent-specification, then τ does not satisfy F or there exists $0 \leq n^1 \leq n$ such that:
  - the trajectory $\langle s_0, \alpha_0, \ldots, s_{n^1} \rangle$ satisfies P_1, and

[4]The meaning of action, fluent, task names would be explained in Section 5.

– the trajectory $\langle s_{n^1}, \alpha_{n^1}, \ldots, s_n \rangle$ satisfies P.

Intuitively, τ satisfies a if the action a is executed at time 0; τ satisfies F if the situation condition F is true at the initial state; τ satisfies $P_1; \ldots; P_m$ if τ satisfies the sequence of procedures $P_1, \ldots, P_m$ continuously; τ satisfies $P_1 | \cdots | P_m$ if τ stochastically satisfies a procedure P_i ($1 \leq i \leq m$); and τ satisfies the conditional and the loop procedures are explained as usual.

At last, given a piece of functional knowledge (T, F) and a piece of procedural knowledge (T, P), let Π be a causal theory for the action model of the robot, if there exists a trajectory τ of Π such that τ satisfies F or P, then the task T would be accomplished if the robot performed the trace specified by τ.

4 INTEGRATING STRUCTURED TASK INSTRUCTIONS FOR PLANNING

In this section, we consider how to compute a robot plan for tasks with the help of structured task instructions. Following the notions used in previous sections, an *extended task planning problem* Δ is a tuple $(\Pi, \mathcal{T}, \mathcal{F}, \mathcal{P})$, where Π is the causal theory for the action domain, $\mathcal{T}$ is a set of task names that need to be accomplished, and $(\mathcal{F}, \mathcal{P})$ is a structured task instruction. A sequence of action names $\langle a^1, \ldots, a^m \rangle$ is a *plan* of Δ, if there exists a trajectory $\tau = \langle s_0, \alpha_1, s_1, \ldots, \alpha_m, s_m \rangle$ of Π such that

- the action name a^i occurs in the action-atom α_i for each $1 \leq i \leq m$ and
- for each task $T \in \mathcal{T}$, either of following two conditions is true
  - there exists a pair $(T, F) \in \mathcal{F}$ such that τ satisfies F; or
  - there exists a pair $(T, P) \in \mathcal{P}$ such that τ satisfies P.

Clearly, the robot could execute a plan in its action domain and accomplish the required tasks after the execution.

Now we provide an approach for computing plans of extended task planning problems by converting structured task instructions into causal rules.

For each pair $(T, F) \in \mathcal{F}$ of an extended task planning problem $\Delta = (\Pi, \mathcal{T}, \mathcal{F}, \mathcal{P})$, we use $tr(T, F)$ to denote the set of the causal rule:

$$F_t \Rightarrow T_t$$

where t is a meta-variable ranging over time names of Π. For each pair $(T, P) \in \mathcal{P}$ of Δ, we use $tr^*(P)$ to denote the set of causal rules: $\neg p_t \Rightarrow \neg p_t$ where p is a new task name for the procedure P and

- If $P = a$ where a is an action name, then
$$a_t \Rightarrow p_t$$
- If $P = F$ where F is a fluent-specification, then
$$F_t \Rightarrow p_t$$
- If $P = T'$ where T' is a task name, then
$$T'_t \Rightarrow p_t$$
- If $P = P_1; \ldots P_m$ where P_i ($1 \leq i \leq m$) are procedures, then $tr^*(P_i)$ for each $1 \leq i \leq m$ and
$$p^1_{t_1} \wedge \cdots \wedge p^m_{t_m} \Rightarrow p_t$$
where p^i is the new task name for the procedure P_i for each $1 \leq i \leq m$, $t_1, \ldots, t_m$ are also meta-variables for time names such that $t_1 \leq \cdots \leq t_m \leq t$.
- If $P = P_1 | \cdots | P_m$ where P_i ($1 \leq i \leq m$) are procedures, then for each $1 \leq i \leq m$, $tr^*(P_i)$ and
$$p^i_t \Rightarrow p_t$$
- If $P =$ **if** F **then** P_1 **else** P_2 where P_1 and P_2 are procedures and F is a fluent-specification, then $tr^*(F; P_1)$, $tr^*(\neg F; P_2)$, and
$$\rho^1_t \Rightarrow p_t$$
$$\rho^2_t \Rightarrow p_t$$
where ρ^1 is the new task name for the procedure $F; P_1$ and ρ^2 is the new task name for the procedure $\neg F; P_2$.
- If $P =$ **while** F **do** P_1, where P_1 is a procedure and F is a fluent-specification, then $tr^*(F)$, $tr^*(P_1)$, and
$$\neg \rho_t \Rightarrow p_t$$
$$p^1_{t_1} \wedge p_t \Rightarrow p_t$$
where ρ is the new task name for the procedure F, p^1 is the new task name for the procedure P_1, and t_1 is the meta-variable for time names such that $t_1 \leq t$.

In fact, the above conversion directly explains the semantics of a procedure in causal rules. Then we use $tr(T, P)$ to denote the set of causal rules: $tr^*(P)$ and

$$p_t \Rightarrow T_t$$

At last, with a slight abuse of the symbol, we use $tr(\Delta)$ to denote the set of causal rules combined with:

Π	
$tr(T, F)$	for each $(T, F) \in \mathcal{F}$
$tr(T, P)$	for each $(T, P) \in \mathcal{P}$
$\neg T_0 \wedge \cdots \wedge \neg T_n \Rightarrow \bot$	for each $T \in \mathcal{T}$

where time names of Π are $0,\ldots,n$.

The following proposition shows that a causal model of $tr(\Delta)$ corresponds to a plan of Δ.

Proposition 1. *Given a causal theory of an action domain with time names $\{0,\ldots,n\}$, a sequence of action names $\langle a^1,\ldots,a^n\rangle$ is a plan of an extended task planning problem Δ w.r.t. the action domain, if and only if, there exists a causal model S of $tr(\Delta)$ such that for each $1 \leq i \leq n$, the action-atom a^i_{i-1} occurs in S.*

As discussed in Introduction, there are mainly two difficulties in applying structured task instructions for planning: Functional and procedural knowledge need to be integrated in a uniform framework for computing task plans, and indispensable details might be missing between two successive steps in a procedure. In our approach, both functional and procedural knowledge are converted to causal rules, then they are used in the uniform reasoning mechanism of causal theories and worked together to compute plans for required tasks. On the other hand, if an action cannot be executed in a procedure, then based on the formalization of the action domain, some proper actions would be automatically added to achieve preconditions of the action and make the computed plan executable.

We have implemented a prototype of the system for computing plans of extended task planning problems and run experiments on many instructions[5]. Again, ASP is used for computing causal models of causal theories. (Ferraris, 2007) specified a polynomial cost conversion from a causal theory to an ASP program such that a causal model corresponds to an answer set. In the prototype, an extended task planning problem Δ is firstly converted to the causal theory $tr(\Delta)$, then $tr(\Delta)$ is converted to its corresponding ASP program whose answer sets are computed by a sophisticated ASP solver *iclingo*. At last, plans of Δ are extracted from these computed answer sets.

5 A CASE STUDY

In this section, we illustrate our approach of integrating structured task instructions for planning on the running example "Heat bread in microwave". Given the structured task instruction $(\mathcal{F},\mathcal{P})$ described in Section 3, we could create an extended task planning $\Delta = (\Pi, \{Heat(bread, micro)\}, \mathcal{F}, \mathcal{P})$ where Π is the causal theory for the action domain of the robot. In particular, Π[6] is a causal theory with action names:

- *move*(L): move to the location L.
- *grasp*(*bread*): grip the bread and picking it up.
- *putdown*(*bread*): put *bread* down.
- *open*(*micro*): open the door of the microwave.
- *close*(*micro*): close the door of *micro*.
- *turnon*(*micro*): turn on *micro* for two minutes.
- *putin*(*bread*, *micro*): put *bread* in *micro*.
- *takeout*(*bread*, *micro*): take *bread* out of *micro*.
- *check*(*bread*): check the temperature of *bread*.
- *wait*: wait for two minutes.

fluent names (some of them):

- *location*(X,L): the object X is located at L.
- *in*(*bread*, *micro*): *bread* is in *micro*.
- *isok*(*bread*): temperature of *bread* is acceptable.

task names:

- *Heat*(*bread*, *micro*): heat *bread* in *micro*.
- *Place*(*bread*, *micro*): place *bread* in *micro*.

Initially, the robot, bread and microwave are located at different places, i.e., *location*$(robot,1)$, *location*$(bread,2)$, *location*$(micro,3)$. The door of *micro* is closed and *bread* needs to be heated three times in *micro* to reach the required temperature.

As proposed in the previous section, the extended task planning problem Δ can be converted to a causal theory $tr(\Delta)$:

$$\Pi \cup tr(\,Place(bread,micro),\, in(bread,micro)\,) \cup tr(\,Heat(bread,micro),\,P\,) \cup \{Heat(bread,micro)_0 \wedge \cdots \wedge Heat(bread,micro)_n \Rightarrow \bot\}$$

In particular, $tr(\,Place(bread,micro),\, in(bread,micro)\,)$ contains the causal rule:

$$in(bread,micro)_t \Rightarrow Place(bread,micro)_t$$

where t is a meta-variable ranging over time names of Π. $tr(\,Heat(bread,micro),\,P\,)$ contains causal rules[7]:

$$\neg p^*_t \Rightarrow \neg p^*_t$$
$$\neg p^w_t \Rightarrow \neg p^w_t$$
$$\neg p_t \Rightarrow \neg p_t$$
$$Place(bread,micro)_{t_1} \wedge turnon(micro)_{t_2} \wedge check(bread)_{t_3} \Rightarrow p^*_t$$
$$isok(bread)_t \Rightarrow p^w_t$$
$$p^*_{t_1} \wedge p^w_t \Rightarrow p^w_t$$
$$Place(bread,micro)_{t_1} \wedge turnon(micro)_{t_2} \wedge check(bread)_{t_3} \wedge p^w_{t_4} \Rightarrow p_t$$

[5] http://ai.ustc.edu.cn/en/robocup/atHome/aspcontrol/

[6] A complete specification can be found at http://ai.ustc.edu.cn/en/robocup/atHome/aspcontrol/example.html.

[7] Due to the limit of space, these causal rules are slightly simplified.

where p^*, p^w, p are the new task names for corresponding procedures, and t_1, t_2, t_3, t_4 are meta-variables for time names such that $t_1 \leq t_2 \leq t_3 \leq t_4 \leq t$.

Time names of Π are $0, \ldots, n$. In practice, the last time n is tested one by one from $1, 2, \ldots$ until $tr(\Delta)$ has a causal model. In the example, $n = 27$ and a computed plan of Δ is:

$$move(2), grasp(bread), move(3),$$
$$putdown(bread), open(micro), grasp(bread),$$
$$putin(bread, micro), close(micro), turnon(micro),$$
$$wait, open(micro), takeout(bread, micro), check(bread),$$
$$putin(bread, micro), close(micro), turnon(micro),$$
$$wait, open(micro), takeout(bread, micro), check(bread),$$
$$putin(bread, micro), close(micro), turnon(micro),$$
$$wait, open(micro), takeout(bread, micro), check(bread).$$

At the beginning, the procedure P requires the fluent $in(bread, micro)$ to be true, then based on the action domain, the system computes a sub-plan to fulfill the requirement:

$$move(2), grasp(bread), move(3), putdown(bread),$$
$$open(micro), grasp(bread), putin(bread, micro).$$

After that, the procedure $turnon(micro); check(bread)$ needs to be performed. However, both actions cannot be executed directly as their preconditions are not satisfied at the moment. Then the system fills the missing details by respectively performing the action $close(micro)$ and the sequence of actions:

$$wait, open(micro), takeout(bread, micro).$$

In the loop procedure, as the environment has been changed, the action $putin(bread, micro)$ could be executed to accomplish the task $Place(bread, micro)$. Then the process continues until *bread* has been heated in *micro* for three times. The above process has been converted to the reasoning process of the causal theory $tr(\Delta)$ and executable plans of Δ can be extracted from corresponding causal models.

6 CONCLUSIONS

We present a formalization for structured task instructions which represents both functional and procedural knowledge of how to accomplish a task and provide an approach for integrating these structured instructions with robot's built-in knowledge to compute plans for tasks. The approach provides a basis for allowing a robot to fulfill open-ended tasks using knowledge obtained from web sources and/or through natural human-robot communication. We have implemented a prototype of the system and a case study is reported on the effectiveness of the approach.

In our approach, both functional and procedural knowledge are converted to causal rules, then they are used in the uniform reasoning mechanism of causal theories and worked together to compute plans for required tasks. Moreover, the approach could compute executable plans when task instructions miss some indispensable steps between two procedures. In particular, if an action cannot be executed in the procedure, the system would automatically add some proper actions to satisfy preconditions of the action which make the computed plan executable. Furthermore, the robot could add new knowledge during the running process and use new knowledge to fulfill tasks without changing its decision-making algorithm.

ACKNOWLEDGEMENTS

This work is supported by the National Hi-Tech Project of China under grant 2008AA01Z150 and the Natural Science Foundation of China under grant 60745002 and 61175057, as well as the USTC Key Direction Project, the USTC 985 Project, the Fundamental Research Funds for the Central Universities, and the Youth Innovation Fund of USTC. Authors are grateful to the other team members and the sponsors.

REFERENCES

Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, New York, NY, USA.

Baral, C., Dzifcak, J., Gonzalez, M., and Zhou, J. (2011). Using inverse lambda and generalization to translate english to formal languages. *Arxiv preprint arXiv:1108.3843*.

Beetz, M., Jain, D., Mösenlechner, L., and Tenorth, M. (2010). Towards performing everyday manipulation activities. *Robotics and Autonomous Systems*, 58(9):1085–1095.

Brenner, M. and Nebel, B. (2009). Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems*, 19(3):297–331.

Burgard, W., Cremers, A., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55.

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 *Proceedings of the 7th annual ACM/IEEE international conference on Human-Robot Interaction (HRI-12)*, pages 471–478. ACM.

Chen, D. L. and Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI-11)*, pages 859–865.

Chen, X., Ji, J., Jiang, J., Jin, G., and Wang, F. (2010). Developing high-level cognitive functions for service robots. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, pages 989–996.

Chen, X., Xie, J., Ji, J., and Sui, Z. (2012). Toward open knowledge enabling for human-robot interaction. *Journal of Human-Robot Interaction*, 1(2):100–117.

Ferraris, P. (2007). A logic program characterization of causal theories. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 366–371.

Galindo, C., Fernández-Madrigal, J., González, J., and Saffiotti, A. (2008). Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955–966.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Thiele, S. (2008). Engineering an Incremental ASP Solver. In *Proceedings of the 24th International Conference on Logic Programming (ICLP-08)*, pages 190–205.

Ghallab, M., Howe, A., Christianson, D., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl—the planning domain definition language. *AIPS98 planning committee*, 78(4):1–27.

Gupta, R. and Hennacy, K. (2005). Commonsense reasoning about task instructions. In *Proceedings of the AAAI-05 Workshop on modular construction of human-like intelligence*, pages 05–08.

Johnston, B. and Williams, M.-A. (2008). Comirit: Commonsense reasoning by integrating simulation and logic. *Frontiers in Artificial Intelligence and Applications*, 171:200–211.

Kress-Gazit, H., Fainekos, G. E., and Pappas, G. J. (2007). From structured english to robot motion. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-07)*, pages 2717–2722. IEEE.

Levesque, H. J., Reiter, R., Lesperance, Y., Lin, F., and Scherl, R. B. (1997). Golog: A logic programming language for dynamic domains. *The Journal of Logic Programming*, 31(1):59–83.

MacMahon, M., Stankiewicz, B., and Kuipers, B. (2006). Walk the talk: connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st national conference on Artificial intelligence (AAAI-06)*, pages 1475–1482. AAAI Press.

McCain, N. and Turner, H. (1997). Causal theories of action and change. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 460–465.

McDermott, D. (1992). Robot planning. *AI magazine*, 13(2):55–79.

Rybski, P., Yoon, K., Stolarz, J., and Veloso, M. (2007). Interactive robot task training through dialog and demonstration. In *Proceedings of the 2nd ACM/IEEE international conference on Human-robot interaction (HRI-07)*, pages 49–56. ACM.

Son, T. C., Baral, C., and McIlraith, S. A. (2001). Planning with different forms of domain-dependent control knowledge – an answer set programming approach. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, pages 226–239.

Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A. G., Teller, S., and Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI-11)*.

Tenbrink, T., Ross, R. J., Thomas, K. E., Dethlefs, N., and Andonova, E. (2010). Route instructions in map-based human–human and human–computer dialogue: A comparative analysis. *Journal of Visual Languages & Computing*, 21(5):292–309.

Tenorth, M., Nyga, D., and Beetz, M. (2010). Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *2010 IEEE International Conference on Robotics and Automation (ICRA-10)*, pages 1486–1491. IEEE.

Thielscher, M. (2005). Flux: A logic programming method for reasoning agents. *Theory and Practice of Logic Programming*, 5(4-5):533–565.

Vogt, P. (2006). Language evolution and robotics: Issues on symbol grounding. *Artificial cognition systems*, pages 145–173.

Xie, J., Chen, X., Ji, J., and Sui, Z. (2012). Multi-mode natural language processing for extracting open knowledge. In *Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Intelligent Agent Technology (IAT-12)*.