Integrating Answer Set Programming with Semantic Dictionaries for Robot Task Planning

Dongcai Lu[1] **Yi Zhou**[2] **Feng Wu**[1*] **Zhao Zhang**[1] **Xiaoping Chen**[1]
[1]School of Computer Science and Technology, University of Science and Technology of China, China
[2]School of Computing, Engineering and Mathematics, University of Western Sydney, Australia
{ludc520,yizhoujoey,wufeng02}@gmail.com, zhao03@mail.ustc.edu.cn, xpchen@ustc.edu.cn

Abstract

In this paper, we propose a novel integrated task planning system for service robots in domestic domains. Given open-ended high-level user instructions in natural language, robots need to generate a plan, i.e., a sequence of low-level executable actions, to complete the required tasks. To address this, we exploit the knowledge on semantic roles of common verbs defined in semantic dictionaries such as *FrameNet* and integrate it with Answer Set Programming — a task planning framework with both representation language and solvers. In the experiments, we evaluated our approach using common benchmarks on service tasks and showed that it can successfully handle much more tasks than the state-of-the-art solution. Notably, we deployed the proposed planning system on our service robot for the annual RoboCup@Home competitions and achieved very encouraging results.

1 Introduction

Service robots are becoming more and more powerful and robust to perform low-level actions [Du Toit and Burdick, 2012; Hofmann *et al.*, 2015; Misra *et al.*, 2016]. However, when robot interacting with end users, task instructions are often expressed in natural language with abstract concepts, such as "I have a headache", "I am thirsty", and "make breakfast". To response, robots need to know how to map user instructions in an open-ended form into low-level (executable) actions. This is a key challenge for robot task planning that has recently attracted many attentions in both the AI and robotics communities [Buehler and Pagnucco, 2014; Cashmore *et al.*, 2015; Chen *et al.*, 2012; Gaschler *et al.*, 2013; Hanheide *et al.*, 2015; Keller *et al.*, 2010; Misra *et al.*, 2016; Nyga and Beetz, 2012; Stock *et al.*, 2015].

A typical approach for task planning is hierarchical task decomposition. More precisely, user instructions or high-level tasks are decomposed into a sequence of low-level subtasks based on pre-defined knowledge, obtained from either domain experts [Misra *et al.*, 2016; Tellex *et al.*, 2011] or open resources [Chen *et al.*, 2013; 2012; Nyga and Beetz, 2012] such as *Open Mind Indoor Common Sense* (OMICS) [Gupta *et al.*, 2004]. For instance, a high-level user task "serve a drink from fridge" can be decomposed into a sequence of subtasks as "go to fridge", "open the fridge door", and "take the drink", based on the knowledge in OMICS.

Unfortunately, there are two main issues with this approach. Firstly, existing knowledge resources (e.g., OMICS) are far from complete so that one cannot find corresponding decomposition rules for a large number of high-level tasks. Secondly, even such a rule exits, it could be the case that the decomposed subtasks are still relatively high-level so that they cannot be directly executed by robots.

Alternatively, researchers turned to automated planning to generate plans with manually created knowledge in the preconditions and/or postconditions. To date, there are many approaches along this track for robot task planning [Buehler and Pagnucco, 2014; Cashmore *et al.*, 2015; Gaschler *et al.*, 2013; Ghallab *et al.*, 2014; Hanheide *et al.*, 2015]. However, most of them depend on hand-coded action effects. For example, in order to automatically generate a plan, robots need to know the beer is on dinner table after executing the task "take a beer from refrigerator to dinner table".

In this paper, we propose a novel approach that integrates *Answer Set Programming* (ASP) [Gelfond and Lifschitz, 1988] with semantic dictionaries [Baker *et al.*, 1998; Fellbaum, 1998] for robot task planning. We aim to addressing two critical challenges: (a) how to obtain large-scale and general-purpose action descriptions particularly about the effects of actions, and (b) how to formalize action descriptions in order to be used in automated planning. Specifically, to address (a), we exploit definitions of common verbs in semantic dictionaries such as *FrameNet* [Baker *et al.*, 1998] and *WordNet* [Fellbaum, 1998], which are handcrafted by linguists. More importantly, such dictionaries are beneficial to both language processing and formalizing more sophisticated than simple hand-coded effects. For (b), we represent action definitions in semantic dictionaries as logic rules and call ASP solvers to generate plans. By doing so, we can do robot task planning and handle large-scale user instructions using knowledge both from action decomposition in knowledge bases (e.g., OMICS) and common-verb descriptions in sematic dictionaries (e.g., *FrameNet*) in a single framework. This significantly reduces workload of hand-coding and increases the degree of autonomy of service robots.

*Corresponding Author.

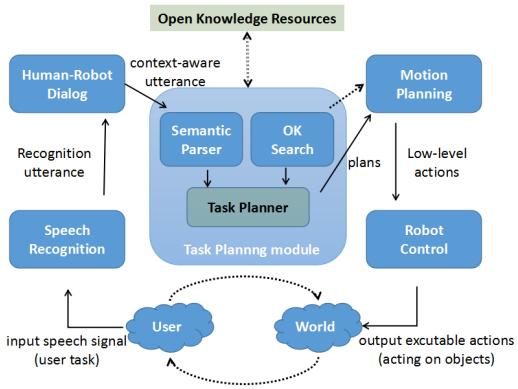

Figure 1: The whole integrated system architecture.

Empirically, we compare our approach with the state-of-the-art solution and our experimental results confirm the advantage of our method. Notably, the proposed technique has been used in our service robot and show encouraging performance in the RoboCup@Home competitions.

Our main contributions are summarized as follow:

- We introduce a meta-language (Section 3.1) to formalize semantic roles of common verbs described in semantic dictionaries (e.g., *FrameNet*), the knowledge of which is missing in existing open resources and not fully described by existing action description languages.
- We propose a method to automatically convert action descriptions in the meta-language to rules in ASP (Section 3.2), which then can be used by standard ASP solvers to generate plans (Section 3.3).

2 System Overview

Figure 1 depicts the overall architecture of our integrated system. To start, the *Speech Recognition* system transcribes speech signal/voice into utterances and the *Human-Robot Dialog* system manages the dialog with users. A sentence in the dialog is then transferred to the *Task Planning* module, which generates plans for the task expressed in natural language. Then, a sequence of commands corresponding to low-level executable actions of the robot is computed by the *Motion Planning* module. Finally, the commands are executed by the *Robot Control* module.

Here, we focus on the *Task Planning* module that takes a sentence as its input and outputs plans. As shown in Figure 1, the main components of our *Task Planning* module include an external *Open Knowledge Resource*, a *Semantic Parser*, an *Open-Knowledge (OK) Search* for (e.g., task decomposition) [Chen *et al.*, 2012] and our *Task Planner* using semantic dictionaries (e.g., *FrameNet, WordNet*).

2.1 Knowledge Resources

Our knowledge base integrates two major resources: task decomposition rules from OMICS described in [Chen *et al.*, 2012] for Open-Knowledge (OK) planner and semantic roles of common verbs from *FrameNet* [Baker *et al.*, 1998] for our ASP planner. *FrameNet* is introduced as follows.

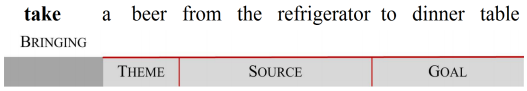

Figure 2: An example of frame-semantic representation.

FrameNet is a digital semantic dictionary providing rich semantic information for action verbs. It groups action verbs into "Frames" and specifies word definitions in terms of semantic roles called *Frame Elements* (FEs) for each frame. It is known that the connections between an action verb and its semantic roles are very useful for resolving under-specification of naturalistic instructions. However, this knowledge cannot be directly used by robots since it is not formalized in *FrameNet*. To address this, we developed a formalized version of *FrameNet*, by translating *FrameNet* knowledge into a formal meta-language, which can be automatically translated into ASP rules.

2.2 Semantic Parser

A semantic parser translates a user instruction in natural language to a task expressed in our meta-language (see Section 3). We use the frame-semantic parser SEMAFOR proposed by [Das *et al.*, 2012] and the basic idea is as follows. Firstly, a delivered natural language sentence is sent to the frame-semantic parser, which was trained using two log-linear models, and the frame-semantic representation of the sentence is returned by the parser. Then the representation is converted into a task expressed in our meta-language.

Suppose that a user task is decomposed into a sequence of steps with one of the steps as "take a beer from the refrigerator to the dinner table". The frame-semantic representation of this step, as shown in Figure 2, is computed by SEMAFOR. Then the representation is translated into the task definition:

(**meta-task** take-Bringing *Theme Source Goal*
(: **parameters** `beer refrigerator dinner-table`))

where verb "take" maps to frame "Bringing" in *FrameNet* and objects "beer", "refrigerator", and "dinner-table" fill into roles THEME, SOURCE, and GOAL respectively.

2.3 Task Planner

The *Task Planner* module generates plans for user instructions. In this paper, we introduce two types of planners. The first is OK planner [Chen *et al.*, 2013; 2012] which generates plans for high-level tasks by decomposition rules in OMICS. Nevertheless, due to lack of knowledge, i.e., decomposition rules and action effects, the performance of the OK planner is severely restricted.

Fortunately, *FrameNet* (and other semantic dictionaries) provides rich knowledge about common verbs. We have discovered that the *FrameNet* definition of an action verb can be reorganized by a set of precondition, postcondition, and invariant over semantic roles of the action (called the functional definition of action). But generally no decomposition knowledge of actions can be obtained from *FrameNet*. Therefore, we need a classical planner that can utilize formalized functional definition of an action to plan the action.

To this end, we significantly improve the OK planner and propose a new robot task planner called the ASP planner. Our new planner makes use of semantic roles of common verbs in semantic dictionaries (e.g., *FrameNet*) and automatically generates plans.

3 ASP + FrameNet for Task Planning

In this section, we present a new method for robot task planning by integrating ASP with rules obtained from *FrameNet*. In order to use *FrameNet* in ASP, we first need to present a meta-language that formalizes semantic roles of common verbs defined in *FrameNet* and their corresponding user tasks. Then, we convert such action descriptions in the meta-language into rules in ASP. Based on which, we call an ASP solver to generate plans for a working example. The main reason that we use ASP is because ASP can serve not only as an inference engine for automated plan generation but also a rich representation language to encode action descriptions in *FrameNet*.

3.1 Meta-Language

We first propose a meta language L_M to formalize semantic roles of common verbs described in *FrameNet*. In L_M, *formulas* are defined recursively upon atomic formulas (i.e, a predicate with its associated arguments) and propositional connectives such as **and**, **or** and **not**.

Then, we define *meta tasks* in the following BNF:

<meta-task-def> ::= <func-def> | <proc-def>
<func-def> ::= (**define** (**meta-task** <string>
(**:parameters** <variable list>)
(**:task-variables** <task variable list>))
[<precond-def>]*
[<postcond-def>]*
[<invariant-def>]*
<proc-def> ::= [<step-def>]*
<precond-def> ::= (**:precondition** <formula>)
<postcond-def> ::= (**:postcondition** <formula>)
<invariant-def> ::= (**:invariant** <formula>)
| (**and** <invariant-def>+)
| (**or** <invariant-def>+)
| (**not** <invariant-def>)
<step-def> ::= (**:step** <task variable>+)
<variable list> ::= <variable>*
<task variable list> ::= <task variable>*
<variable> ::= ?<string>
<task variable> ::= ?<string>

Intuitively, a meta task specifies the knowledge extracted from *FrameNet* for a common verb with a certain frame.

Here, we consider two types of meta tasks. A *procedural meta task* (i.e., <proc-def>) is specified by a sequence of steps, in which each step is a subtask. This is a hierarchical task decomposition rule stating that the high-level task can be accomplished by the sequence of steps (i.e., subtasks). In *FrameNet*, some FEs provide procedural definitions of a frame. A *functional meta task* (i.e., <func-def>) is specified by a tuple of preconditions ??i.e., <precond-def>), postconditions (i.e., <postcond-def>) and/or invariants (i.e., <invariant-def>). For executing a functional meta task, precondition/postcondition is a formula that has to be true before/after the execution, while invariant is a formulas that remains to be true during the whole execution.

Then, given the definition of semantic roles of a common verb, we can formalize its corresponding action description in our meta language L_M. For better understanding, we illustrate L_M with the following example.

Example 1 *Consider the description of verb* `take` *with frame* Bringing *in FrameNet, in which the set of linguistic meta variables is* {*Robot*, *Theme*, *Source*, *Goal*, *Carrier*}.

1. *Theme is portable;*
2. *Source is the initial location of Theme when* take *starts;*
3. *Goal is the final location of Theme when* take *ends;*
4. *Source and Goal should not be the same location;*
5. *Theme and Robot share the same location during the execution of* take*;*
6. *Theme is on Robot or in Carrier operated by Robot.*

We formalize this as a meta task take-Bringing *in our meta language L_M as follow:*

(***define*** (***meta-task*** *take-Bringing*
(***:parameters*** $?robot$ $?theme$ $?source$ $?goal$ $?carrier$)
(***:task-variables***))
(***:precondition*** (***and*** (*location* $?source$)
(*location* $?goal$)))
(***:precondition*** (*portable* $?theme$))
(***:precondition*** (*at* $?theme$ $?source$))
(***:postcondition*** (*at* $?theme$ $?goal$))
(***:precondition*** (***not*** (*equal* $?source$ $?goal$)))
(***:invariant*** (*same-location* $?theme$ $?robot$))
(***or*** (***:invariant*** (*on* $?theme$ $?robot$))
(***:invariant*** (***and*** (*in* $?theme$ $?carrier$)
(*operated* $?robot$ $?carrier$)))))

In order to convert semantic roles of common verbs in *FrameNet* into our meta-language, we first adopted existing tools, including `PREPOST` [Sil *et al.*, 2010] and some open information extraction tools [Angeli *et al.*, 2015], to help us automatically extracting information related to frames and FEs. For example, by using the tool from [Angeli *et al.*, 2015], "*Source* is the initial location of *Theme* when *take* starts" is translated to $(Source, initial\ location, Theme)$, which is then converted to **:precondition** (at $?theme$ $?source$). Occasionally, we may need to validate and revise these action formalizations. This requests much less human involvement than hand-coding all rules by experts and save us a lot of time to formalize action descriptions into our meta-language.

3.2 Converting Action Descriptions to ASP Rules

Following the same idea of Satplan [Kautz and Selman, 2006] to view automated planning as a constraint satisfaction problem, ASP can be used as an automated planner as well [Lifschitz, 2002]. Given a planning problem P, one can construct an ASP program Π^P such that $\langle s_0, a_1, s_1, \ldots, a_n, s_n\rangle$ is a trajectory of a plan of P iff there exists an answer set of Π^P that contains *holds*(f, i) for each fluent f and $0 \leq i \leq n$ such that $s_i \models f$, *holds*$(\neg f, i)$ for each fluent f and $0 \leq i \leq n$ such that $s_i \models \neg f$ and *occurs*(a_i, i) for each action a_i and $1 \leq i \leq n$. Intuitively, that *holds*(f, i) (*holds*$(\neg f, i)$) belongs to S means that the fluent f is true (false) in the state s_i; that *occurs*(a_i, i) belongs to S means that the action a_i occurs at the state s_{i-1}.

Here, one of the main challenges of ASP is how to obtain the rules about action effects. Traditionally, this is hand-coded by experts. In our approach, we present a new method to obtain those rules by exploiting semantic roles of common verbs in semantic dictionaries (e.g., *FrameNet*). In the previous section, we formalize semantic roles of common verbs in *FrameNet* into action descriptions in our meta language L_M. Now, we show how to automatically convert those action descriptions into ASP rules.

We first translate formulas in L_M into rules. Formally, let F be a form obtained from <formula>. We use *tr*(F, i) (for some nonnegative integer i) to denote the set of ASP rules:

- If F is in the form (p t_1 $\cdots$ t_m) where p is the name of a predicate and $t_1, \ldots, t_m$ are semantic roles, then $p(t_1, \ldots, t_m)$ is a fluent in the action signature, thus
$$\textit{holds}(F, i) \leftarrow \textit{holds}(p(t_1, \ldots, t_m), i).$$
- If F is in the form (**and** F_1 $\cdots$ F_m) where $F_1, \ldots, F_m$ are <formula>, then *tr*(F_j, i) for each $1 \leq j \leq m$ and
$$\textit{holds}(F, i) \leftarrow \textit{holds}(F_1, i), \ldots, \textit{holds}(F_m, i).$$
- If F is in the form (**or** F_1 $\cdots$ F_m) where $F_1, \ldots, F_m$ are <formula>, then *tr*(F_j, i) for each $1 \leq j \leq m$ and
$$\textit{holds}(F, i) \leftarrow h(F_1, i).$$
$$\cdots$$
$$\textit{holds}(F, i) \leftarrow h(F_m, i).$$
- If F is in the form (**not** F_1), then *tr*(F_1, i) and
$$\textit{holds}(F, i) \leftarrow \textit{not holds}(F_1, i).$$

Based on this, we are able to translate meta tasks in L_M into rules in ASP as well. Let ϕ be a task based on the meta task Φ, and $\mathcal{I}$ a <invariant-def> in Φ, we use *tr*$(\mathcal{I}, k, n)$ (for some nonnegative integers k and n s.t. $k \leq n$) to denote the set of ASP rules:

- If the form is (**:invariant** <formula>) and F is <formula>, then *tr*(F, i) $(k \leq i \leq n)$ and
$$\textit{true}(\mathcal{I}, k, n) \leftarrow \textit{holds}(F, k), \ldots, \textit{holds}(F, n).$$
- If the form is (**:and** $\mathcal{I}_1$ $\cdots$ $\mathcal{I}_m$) where $\mathcal{I}$'s are <invariant-def>, then *tr*$(\mathcal{I}_i, k, n)$ $(k \leq i \leq n)$ and
$$\textit{true}(\mathcal{I}, k, n) \leftarrow \textit{true}(\mathcal{I}_1, k, n), \ldots, \textit{true}(\mathcal{I}_m, k, n).$$
- If the form is (**:or** $\mathcal{I}_1$ $\cdots$ $\mathcal{I}_m$) where $\mathcal{I}$'s are <invariant-def>, then *tr*$(\mathcal{I}_i, k, n)$ $(k \leq i \leq n)$ and
$$\textit{true}(\mathcal{I}, k, n) \leftarrow \textit{true}(\mathcal{I}_1, k, n).$$
$$\cdots$$
$$\textit{true}(\mathcal{I}, k, n) \leftarrow \textit{true}(\mathcal{I}_m, k, n).$$
- If the form is (**:not** $\mathcal{I}_1$) where $\mathcal{I}_1$ is <invariant-def>, then *tr*$(\mathcal{I}_1, k, n)$ and
$$\textit{true}(\mathcal{I}, k, n) \leftarrow \textit{not true}(\mathcal{I}_1, k, n).$$

Let ϕ be a task based on the meta task Φ, we use *tr*(ϕ, k, n) $(k \leq n)$ to denote the set of ASP rules:

- For each <precond-def> in Φ and F be a <formula>, then *tr*(F, k) and
$$\leftarrow \textit{complete}(\phi, k, n), \textit{not holds}(F, k).$$
- For each <postcond-def> in Φ and F be a <formula>, then *tr*(F, n) and
$$\leftarrow \textit{complete}(\phi, k, n), \textit{not holds}(F, n).$$
- For each <invariant-def> $\mathcal{I}$ in Φ, then *tr*$(\mathcal{I}, k, n)$ and
$$\leftarrow \textit{complete}(\phi, k, n), \textit{not true}(\mathcal{I}, k, n).$$
- For each <step-def> $\mathcal{S}$ in Φ and ϕ_1 $\cdots$ ϕ_m be the sequence of task/action names obtained from <task variable list>, then for every possible sequence $n^1, \ldots, n^{m-1}$ of nonnegative integers with $k \leq n^1 \leq \cdots \leq n^{m-1} \leq n$, we denote $n^0 = k$ and $n^m = n$, then for each $1 \leq i \leq m$,
  - if ϕ_i is an action, then
  $$\leftarrow \textit{complete}(\phi_i, n^{i-1}, n^i), \textit{not occurs}(\phi_i, n^{i-1}{+}1).$$
  - if ϕ_i is a task, then *tr*(ϕ_i, n^{i-1}, n^i),

  and the rules
$$\begin{aligned}\textit{true}(\mathcal{S}, k, m) \leftarrow\ & \textit{complete}(\phi_1, k, n^1),\\ & \textit{complete}(\phi_2, n^1, n^2),\\ & \ldots, \textit{complete}(\phi_m, n^{m-1}, n).\\ \leftarrow\ & \textit{complete}(\phi, k, n), \textit{not true}(\mathcal{S}, k, n).\end{aligned}$$

Finally, *tr*(ϕ, k, n) contains rules:
$$\textit{complete}(\phi, k, n) \leftarrow \textit{not ncomplete}(\phi, k, n).$$
$$\textit{ncomplete}(\phi, k, n) \leftarrow \textit{not complete}(\phi, k, n).$$

3.3 Plan Generation

Now, we are able to use an ASP program including the rules obtained from semantic roles of common verbs in *FrameNet* for robot task planning. Suppose that we have a robot task planning problem with a high-level task and some potential low-level actions and fluents (i.e. atomic formulas with time arguments) in a domain. Firstly, we seek for related common verbs in *FrameNet*. If found, we convert the semantic roles of these common verbs into ASP rules by the aforementioned techniques. Together with rules obtained from the high-level task and other rules for ASP planning, we form an ASP program. Finally, we call an ASP solver to compute an answer set of the ASP program grounded by objects in the domain, which corresponds to a executable plan for the high-level task.

Table 1: Top 10 most frequent verbs in data sets.

Verb	Frequency	Verb	Frequency
put	4591	take	2452
turn	2754	go	1780
open	2608	find	1731
get	2568	pick	1632
place	2456	remove	1446

Table 2: Experimental results on two sets.

Open Knowledge	Tasks/Steps Sets		Help Sets	
	local	*global*	*local*	*global*
OK planner	66	94	129	144
OK planner+WordNet	111	331	140	168
ASP planner	765	889	330	340
ASP planner+WordNet	**790**	**935**	**331**	**379**

Example 2 *[Continued with Example 1] Consider the high level task "take a beer from the refrigerator to the dinner table" with some (not limited to) low level actions*

- $move(L)$[1]*: move to the location* L*;*
- $grasp(x)$*: grip the object* x *and pick it up;*
- $putdown(x)$*: put the object* x *down;*
- $open(y)$*: open the object* y*;*
- $close(y)$*: close the object* y*;*
- $takeout(x, y)$*: take the object* x *out of the object* y*;*

as well as some (not limited to) fluent:

- $location(L)$*:* L *is a location;*
- $portable(x)$*: the object* x *is portable by the robot;*
- $at(x, L)$*: the object* x *is located at the location* L*;*
- $in(x, y)$*: the object* x *is in the object* y*;*
- $on(x, y)$*: the object* x *is on the object* y*;*
- $opened(y)$*: the object* y *is opened;*
- *same-location*(x, y)*:* x, y *have the same location;*
- $equal(L_1, L_2)$*: location* L_1 *and* L_2 *are the same.*

We first analyze the high-level task "take a beer from the refrigerator to the dinner table" by the semantic parser. From which, we know that "take" is a high level task while "beer", "refrigerator" and "dinner table" are concrete objects in the domain. Then, we seek for related common verbs in FrameNet that are similar to the task name "take", and we find a matching one "task-bringing". Next, we convert the semantic roles of the common verb "task-bringing" into ASP rules, as shown in aforementioned techniques. Then, we ground the program obtained from those rules and the domain objects "beer", "refrigerator" and "dinner table", and we can found that "beer" is matched as a $?theme$*, "refrigerator" as a* $?source$*, and "dinner table" as a* $?goal$*.*

Finally, we call an ASP solver iclingo/oclingo[2] *to solve the grounded ASP program and find the following answer set:*

$$move(\textit{fridge}), open(fridge), takeout(beer, fridge), \\ move(\textit{dinner-table}), putdown(beer),$$

which is a plan for the original high level task "take a beer from the refrigerator to the dinner table".

4 Experiments

In the experiments, we evaluated our ASP planner on the common benchmarks widely used for domestic service robots [Chen *et al.*, 2013; 2012; Kunze *et al.*, 2010; Tenorth and Beetz, 2013], consisting of two data sets with 11885 task-oriented user instructions from the *Tasks/Steps* table and 467 desire-oriented user instructions from the *Help* table of OMICS. Table 1 shows the top 10 most frequently used common verbs in the data sets. It can be observed that most of them are not low-level executable actions (such as move, find, pick_up, put_down, open and close) but high-level common verbs (such as put, turn, get and place). Hence, it is crucial for service robots to do task planning, i.e., generating a sequence of low-level executable actions in order to complete the required tasks described by those high-level common verbs in the benchmark problems.

We compared our ASP planner to the OK planner [Chen *et al.*, 2013] — currently the leading task planner for domestic service robots that is capable of exploiting open knowledge. Specifically, we compared both planners in terms of the number of successfully planned tasks on the aforementioned benchmarks. In addition, we consider two sets of configurations. The first is about the searching method. While $local$ means that the system only use one decomposition rule, $global$ means that it will explore all decomposition rules in the data sets. The second configuration is whether the task planner utilizes other semantic dictionaries such as WordNet [Fellbaum, 1998] for measuring word similarities. For instance, "set_down" is semantically similar to "put_down". Here, $WordNet$ means that this technique is used.

Table 2 summarizes our results. Our ASP planner significantly outperforms the OK planner in terms of the number of successfully planned tasks in all categories. For instance, for using the local search method without *WordNet* on 11885 task-oriented instructions, while the OK planner can only successfully generate 66 plans, our ASP planner can generate 765, which is 11.6 times better. Even for the least improvements, i.e., the global search method with *WordNet* on desire-oriented instructions, our ASP planner is 2.25 (=379/168) times better than the OK planner.

There are two main reasons for this improvement. Firstly, semantic roles of common verbs in *FrameNet* brings more knowledge for task decomposition and planning. Secondly, the ASP solver can automatically explore all possible combinations of action sequences and generate plans, which is much more powerful than predefined hand-coded task decomposition rules.

Notably, the ASP planner + *WordNet* approach can achieve a high success rate of 81.16% (379 out of 467) for task planning on desire-oriented instructions. This is because the approach utilizes many semantic roles of common verbs in *FrameNet*. Indeed, as shown in Figure 3, the more frames added, the higher success rate we have. Notice that sometimes the curves experience a flat improvement (e.g., our

[1] An action name with variables is consider to be a shorthand for the set of all ground instances.

[2] http://www.cs.uni-potsdam.de/oclingo/

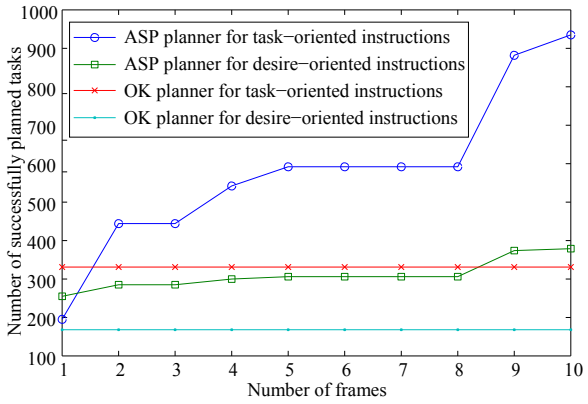

Figure 3: Evaluation Results of task planning.

ASP planner for task-oriented instructions in the middle). This is due to the limit of our robots' capability to perform some low-level executable actions so that some frames (e.g., *Mass_motion* and *Waiting*) cannot be grounded.

Nevertheless, there are still a large number of task-oriented instructions that cannot be successfully planned by our approach. There are two typical types of failures. The first failure type is about the natural language processing, that is, our semantic parser cannot retrieve any frames from *FrameNet*. As far as we have counted, 26.7% of the failures belong to this category. The second failure type is ASP failures. In this case, our ASP planner fails to generate a plan. There are two main reasons. First, *FrameNet* is far from complete so that there is a huge lack of knowledge about actions. Second, the number of low-level executable actions for our robot is restricted so that some high-level actions are essentially beyond its capability. However, the success rate will be improved by using the same method proposed in our approach with more knowledge created in new online resources and more reliable information extraction tools.

Apart from the above corpus-based experiments, it is worth pointing out that our integrated system with the proposed ASP planner has been deployed in a service robot and used in the annual RoboCup@Home[3] competitions for planning tasks with user instructions in natural language, which won the champion in 2014 and the runner-up in 2015. This confirms the usefulness of our approach for real robot task planning. A demo video with our robot system is available at: https://youtu.be/bBU86dciEwo

5 Related Work

Many approaches on task planning for service robots have been proposed in the literature. Unlike our integrated system, most of them decompose user instructions into low-level subtasks based on pre-given knowledge obtained from domain experts [Misra *et al.*, 2016; Tellex *et al.*, 2011], which usually scale poorly to large number of tasks. To improve generality and scalability, researchers have tried to exploit open knowledge resources [Chen *et al.*, 2013; 2012; Nyga and Beetz, 2012]. Still, there are a large portion of tasks that cannot be decomposed based on the decomposition rules using existing open resources. This motivates us to develop our system with ASP planner and sematic dictionaries.

There are also a number of approaches using AI techniques for robot task planning. To name a few, these planning systems are implemented for performing service tasks in a kitchen environment [Keller *et al.*, 2010], finding sequence of actions required by several different robots [Buehler and Pagnucco, 2014], or embedded into *Robot Operating System* (ROS) [Cashmore *et al.*, 2015]. [Hanheide *et al.*, 2015] implemented a three-layer architecture on a mobile robot to enable robots to plan with uncertain and incomplete information. Some approaches [Gaschler *et al.*, 2013] do not only reason about low-level executable actions but also on geometric preconditions and the effects of complex actions. However, most of them require hand-coded action effects for specific domains with restricted scalability and generality.

Our meta-language is related to the *Modular Action Description* (MAD) language proposed by [Lifschitz and Ren, 2006]. MAD is devised to compile a general-purpose database of knowledge about actions, in which one can "factor out" common elements of specific action domains on commonsense reasoning and planning. There are also several significant differences from ours. Firstly, our meta-language is motivated to express rewritten knowledge related to common verbs and their corresponding user tasks, while MAD focuses on representing common features of actions in order to make the language more elaboration tolerant [McCarthy, 1998]. Secondly, the definition of a common verb may involve a sequence of steps on how to accomplish the corresponding tasks as in our meta-language.

6 Conclusion

We have presented a novel integrated system for large-scale, general-purpose robot task planning. Specifically, we proposed a new method that combines automated planning in ASP and semantic roles of common verbs defined in semantic dictionaries (e.g., *FrameNet*). Our experimental results on the OMICS datasets show that the integrated system benefitted from the knowledge about actions obtained from *FrameNet* and significantly outperformed the state-of-the-art solution. Furthermore, based on the planning system, our service robot achieved encouraging results with one champion and one runner-up in the RoboCup@Home competitions.

For future work, one potential direction is to automatically extract knowledge about actions from not only semantic dictionaries but also other resources including plain texts and web pages. Additionally, we plan to consider a richer syntax that is able to represent different kinds of knowledge.

Acknowledgments

This work was supported by National Natural Science Foundation of China under grant No. U1613216. Feng Wu was supported in part by National Natural Science Foundation of China under grant No. 61603368, the Youth Innovation Promotion Association of CAS (No. 2015373), and Natural Science Foundation of Anhui Province under grant No. 1608085QF134.

[3] http://www.robocupathome.org

References

[Angeli *et al.*, 2015] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. *Linguistics*, (1/24), 2015.

[Baker *et al.*, 1998] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proc. of Coling*, pages 86–90, 1998.

[Buehler and Pagnucco, 2014] Jennifer Elisabeth Buehler and Maurice Pagnucco. A framework for task planning in heterogeneous multi robot systems based on robot capabilities. In *Proc. of AAAI*, pages 2527–2533, 2014.

[Cashmore *et al.*, 2015] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, et al. Rosplan: Planning in the robot operating system. In *Proc. of ICAPS*, pages 333–341, 2015.

[Chen *et al.*, 2012] Xiaoping Chen, Jiongkun Xie, Jianmin Ji, and Zhiqiang Sui. Toward open knowledge enabling for human-robot interaction. *JHRI*, 1(2):100–117, 2012.

[Chen *et al.*, 2013] Xiaoping Chen, Jianmin Ji, Zhiqiang Sui, and Jiongkun Xie. Handling open knowledge for service robots. In *Proc. of IJCAI*, 2013.

[Das *et al.*, 2012] Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. Frame-Semantic Parsing. *CL*, 39(4):1–47, 2012.

[Du Toit and Burdick, 2012] Noel E Du Toit and Joel W Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE TRO*, 28(1):101–115, 2012.

[Fellbaum, 1998] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[Gaschler *et al.*, 2013] Andre Gaschler, Ronald PA Petrick, Manuel Giuliani, Markus Rickert, and Alois Knoll. Kvp: A knowledge of volumes approach to robot task planning. In *Proc. of IROS*, pages 202–208, 2013.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proc. of ICLP*, pages 1070–1080, 1988.

[Ghallab *et al.*, 2014] Malik Ghallab, Dana Nau, and Paolo Traverso. The actor's view of automated planning and acting: A position paper. *Artificial Intelligence*, 208:1–17, 2014.

[Gupta *et al.*, 2004] Rakesh Gupta, Mykel J Kochenderfer, Deborah Mcguinness, and George Ferguson. Common sense data acquisition for indoor mobile robots. In *Proc. of AAAI*, pages 605–610, 2004.

[Hanheide *et al.*, 2015] Marc Hanheide, Moritz Göbelbecker, Graham S Horn, Andrzej Pronobis, Kristoffer Sjöö, et al. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 2015.

[Hofmann *et al.*, 2015] Andreas Hofmann, Enrique Fernandez, Justin Helbert, Scott Smith, and Brian Williams. Reactive integrated motion planning and execution. In *Proc. of IJCAI*, pages 1881–1887, 2015.

[Kautz and Selman, 2006] Henry Kautz and Bart Selman. Satplan04: Planning as satisfiability. *Working Notes on the Fifth IPC*, pages 45–46, 2006.

[Keller *et al.*, 2010] Thomas Keller, Patrick Eyerich, and Bernhard Nebel. Task planning for an autonomous service robot. In *Annual Conference on Artificial Intelligence*, pages 358–365, 2010.

[Kunze *et al.*, 2010] Lars Kunze, Moritz Tenorth, and Michael Beetz. Putting people's common sense into knowledge bases of household robots. In *AAAI*, pages 151–159, 2010.

[Lifschitz and Ren, 2006] Vladimir Lifschitz and Wanwan Ren. A modular action description language. In *Proc. of AAAI*, pages 853–859, 2006.

[Lifschitz, 2002] Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1):39–54, 2002.

[McCarthy, 1998] John McCarthy. Elaboration tolerance. In *Common Sense*, volume 98, 1998.

[Misra *et al.*, 2016] Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. volume 35, pages 281–300. SAGE Publications Sage UK: London, England, 2016.

[Nyga and Beetz, 2012] Daniel Nyga and Michael Beetz. Everything robots always wanted to know about housework (but were afraid to ask). In *IROS*, pages 243–250. IEEE, 2012.

[Sil *et al.*, 2010] Avirup Sil, Fei Huang, and Alexander Yates. Extracting action and event semantics from web text. In *AAAI Fall Symposium: Commonsense Knowledge*, 2010.

[Stock *et al.*, 2015] Sebastian Stock, Masoumeh Mansouri, Federico Pecora, and Joachim Hertzberg. Online task merging with a hierarchical hybrid task planner for mobile service robots. In *Proc. of IROS*, pages 6459–6464, 2015.

[Tellex *et al.*, 2011] Stefanie A Tellex, Thomas Fleming Kollar, Steven R Dickerson, Matthew R Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. of AAAI*, 2011.

[Tenorth and Beetz, 2013] Moritz Tenorth and Michael Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *IJRR*, 32(5):566–590, 2013.