Semantic Attachments for Domain-Independent Planning System

Elective in Artificial Intelligence

Reasoning Agents

Prof. Fabio Patrizi

Giorgia Natalizia - 1815651

SAPIENZA
UNIVERSITÀ DI ROMA

Outline

- **Introduction**
- **Examples of real problems**
- **Semantic attachements**
- **Implementation**
- **Experiments**
- **Conclusions**

Introduction

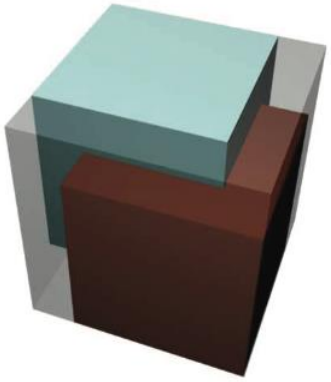
Planning of high level tasks could be very challenging and several approaches has been developed for example hierarchical decomposition:

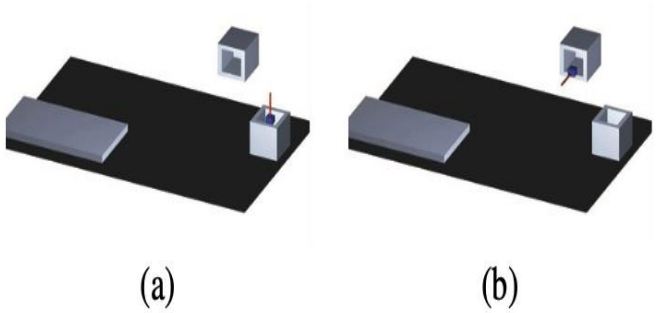
- Top down
- Bottom up

Third approach that integrates high and low-level planning. Low-level reasoner can provide information to the high-level planner during the planning process, but is only evoked if relevant to the high-level planner.

To integrate information about special-purpose reasoning into symbolic planning are used ***semantic attachments*** to a planning domain description such as PDDL.

Examples of real problems





Transport domain

Common logistic prooblem when trucks deliver multiple packages to different locations

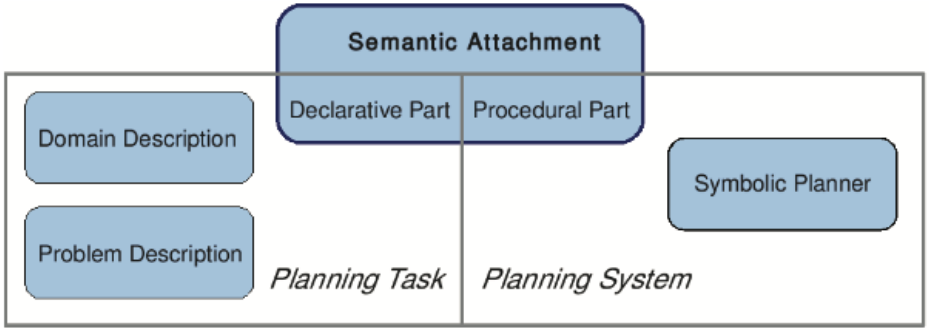
Robot manipulation domain

Blocks-word domanin in which we have a box which is open at the top, a shelf, a table, and a little moveable box The gripper is simply a stick that can connect to moveable objects

Semantic attachments

Semantic attachments are external procedural reasoning modules that may compute the valuations of state variables at planner run-time.

- **Declaratve part**: describes their use in planning domain
- **Procedural part**: is the actual algorithm for computing the value of a state variable



Implementation

It is necessary to extend the description language for planning tasks.
We use new PDDL requirement :modules to indicate that a planning domain uses semantic attachments. They are very similar to predicates and have *condition-checker (test preconditions)* and *effect-applicator (compute changes to any state variable)*.

```
(:modules
   (canLoad ?v - vehicle ?p - package
    conditionchecker canLoad@libTrans.so) )
```

Sound and **complete** under the following assumption:

- Condition-checker always terminates and return a truth value
- Effect -applicator always terminates and return same values for the same settings.

Implementation

Modules are implemented as dynamically loaded shared libraries.
Planner and the modules need to use a common interface that defines two function types for the two kinds of semantic attachments implemented.
Two search planners has been extended:

- FF/M: *FF/M* is an extension of FF supporting semantic attachments.

- TFD/M: is an extension of a domain-independent progression search planner supporting semantic attachments. Has 2 main phases: Translation and search.

Sound and **complete** under some assumption.
Can be integrated in **heuristics**.

Experiments

EXPERIMENT 1

This experiment is based on *crew-planning* domain of IPC 2008 which contains different operators with the predicate *available* in common.
In the experiment there is a comparison of the runtime of TFD and TFD/M in the execution of a callback requesting the truth value of the predicate *available*.

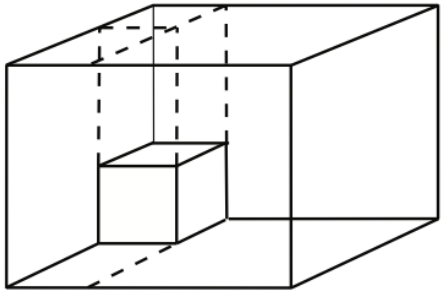
#	TFD	TFD/M	%	#	TFD	TFD/M	%
01	0.01	0.01	0	**16**	0.61	0.78	28
02	0.01	0.02	100	**17**	0.73	0.96	32
03	0.01	0.02	100	**18**	0.85	1.10	29
04	0.04	0.05	25	**19**	1.89	2.38	26
05	0.08	0.10	25	**20**	3.19	4.06	27
06	0.14	0.18	29	**21**	2.47	3.12	26
07	0.16	0.24	50	**22**	0.16	0.19	19
08	0.18	0.24	33	**23**	0.12	0.14	17
09	0.29	0.37	28	**24**	0.20	0.26	30
10	0.59	0.75	27	**25**	—	—	—
11	0.47	0.61	30	**26**	1.50	1.89	26
12	0.58	0.76	31	**27**	—	—	—
13	0.05	0.08	60	**28**	3.82	4.71	23
14	0.08	0.12	50	**29**	5.74	7.21	26
15	0.06	0.07	17	**30**	5.55	6.89	24

There is an overhead introduced by module calls but usually the module's calculations takes the majority of the time.

Experiments

EXPERIMENT 2

Full implementation of PDDL/M with non-trivial semantic attachments in a classic logistic task.



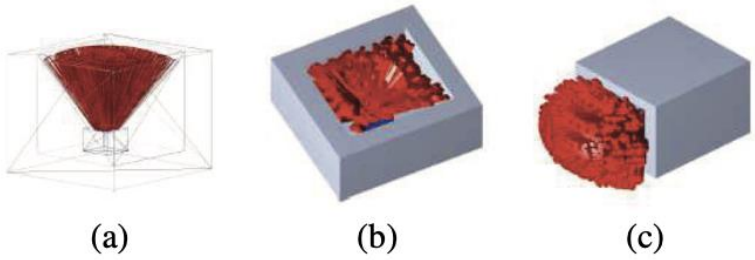
Semantic attachment are implemented as condition-checker module *canLoad* in order to fit a truck with the possible packages (correct but non optimal solution)

#	Trucks	Packs	Nodes	Runtime
01	2	2	5	0.01
02	2	4	10	0.36
03	3	6	15	0.81
04	3	8	20	1.70
05	3	10	25	33.86
06	4	12	30	27.47
07	4	14	35	146.62
08	4	18	45	244.45

Experiments

EXPERIMENT 3

Plan based robot manipulation.
Each pose is defined as a tuple
p = (rotx (p), roty (p), rotz (p)), where for i ∈ (x, y, z), roti(p) ∈ [0, 2Π] is the rotation of the robot gripper around the i axis. The module check all the possible grasp poses.



(a) (b) (c)

#	Cubes	Boxes	Tables	Runtime
01	1	1	1	0.20
02	4	1	1	1.20
03	8	1	1	5.00
04	12	1	1	12.80
05	1	4	4	0.58
06	1	8	8	1.06
07	1	50	50	6.27
08	4	4	1	5.30
09	6	6	1	36.55
10	12	12	1	1444.90

The task consists of multiple cubes that have to be moved out of boxes onto a single shelf. Tables serve as temporary placement possibilities where grasp poses could be changed if necessary. Using FF/M solve the problem in reasonable time.

Conclusions

Using planning in real word problems could be difficult. In particular in robotics application the application domain may be very hard to describe declaratively but instead be computed when needed.

The presented approach integrates external reasoning mechanisms, so-called *semantic attachments*, directly into a planner.

This method provides the planner with better information reducing future execution failure.