Visual design of planning domains

Jindřich Vodrážka
Charles University in Prague
Faculty of Mathematics and Physics
Malostranské náměstí 25
118 00 Praha 1, Czech Republic
vodrazka@ufal.mff.cuni.cz

Lukáš Chrpa
Czech Technical University in Prague
Agent Technology Center
Karlovo náměstí 13
121 35 Praha 2, Czech Republic
chrpa@agents.felk.cvut.cz

Introduction

Description of planning domains and problems is the first critical task when using planning technology. It naturally belongs to the area of Knowledge Engineering as it involves knowledge extraction (from the user) and schematic formulation of problems.

To make the task more comprehensive for non-experts in planning we propose to use graphical representation for planning domains. This method has been already used in systems GIPO (Simpson *et al.* 2007) and itSIMPLE (Vaquero *et al.* 2007). In this paper we will describe system VIZ inspired by them. Unlike GIPO or itSimple, VIZ is a lightweight system which uses straightforward approach to model a planning domain. Users do not need to be familiar with PDDL syntax. VIZ provides a graphical user interface for description of planning domains and problems. The interface uses collection of simple diagrams which can be exported directly into PDDL.

Categorization

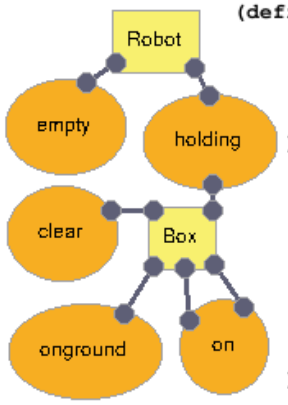
Planning domain designers usually start with informal description of some system. Available pieces of information need to be categorized with respect to their meaning. Concept of object oriented programming and formalism of first order logic (FOL) is incorporated in the following categories which are used in VIZ:

- *class* determines common properties for all *objects* which belong to it. It can be understood as a set of *objects*.
- *object* is a specific instance of some *class*
- *variable* can refer to any *object* from particular *class*
- *predicate* denotes an atomic statement of FOL language for a given planning domain

Design levels

It is convenient to split complex task of planning domain design into pieces. We can consider three levels of abstraction:

1. declaration of *classes* and *predicates*
2. definition of planning operators using *variables* and previously declared *predicates*
3. definition of planning problem using *objects* and *predicates*




```
(define (domain Blockworld )
    (:requirements :strips
                   :typing)
    (:types
       Box Robot - object
    )
    (:predicates
        (empty ?a - Robot)
        (clear ?a - Box)
        (onground ?a - Box)
        (holding ?a - Robot
                  ?b - Box)
        (on ?a - Box
            ?b - Box)
    )
```

Figure 1: Declaration of language

Planning domain is described at the first two levels. Planning problems are described at the third level.

Program VIZ

Three types of diagrams are sufficient to describe simple planning domain with VIZ. All diagram types share the same idea. Semantics of diagrams correspond with three levels of abstraction described earlier. Number of diagrams depends on number of planning operators and problems.

Blockworld domain (Slaney and Thiébaux 2001) is used as an example in following Figures. Corresponding PDDL code is shown as well.

Declaration of language

In Figure 1 we can see rectangular nodes labeled `Box` and `Robot` representing *classes* used in the Blockworld domain. There are also eliptic nodes as a representation for *predicates*. By connecting e.g. *predicate* `empty` and *class* `Robot`, we can declare type of `empty`'s only argument. Generally we start declaration of *predicates* with zero arguments. Then we add new arguments by connecting the node which represents the *predicate* to some *class*. In this manner types of predicate arguments will be defined as well. Various information e.g. the order of predicate arguments can be displayed on demand in the VIZ's property editor.

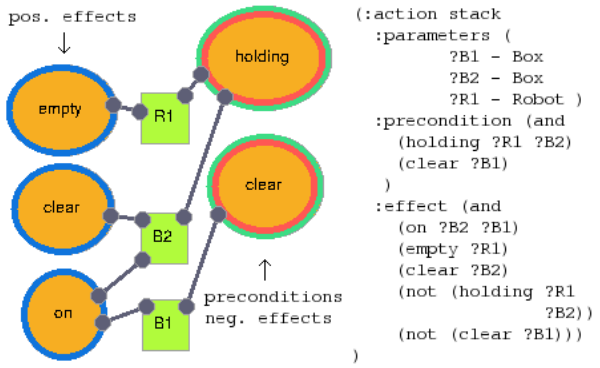


Figure 2: Planning operator example

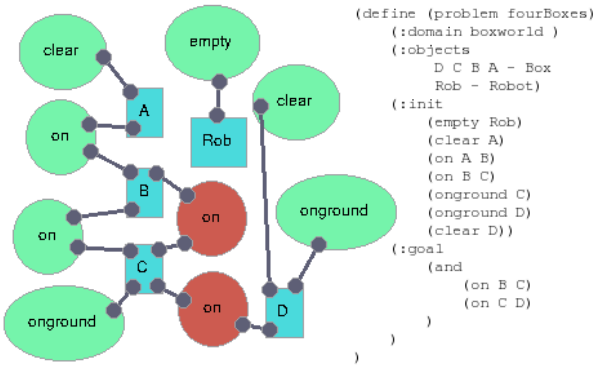


Figure 3: Planning problem example

Definition of planning operators

In Figure 2 we can see a diagram representing planning operator `stack`. Rectangular nodes in this diagram represent *variables*. We can see only their names, but VIZ allows to set their *class* as well. Notable difference is in visualisation of *predicates*. Predicates in operator Op are divided into three disjoint sets ($effect^+$,$effect^-$,$precond$). This is marked with three different colours. In the first set, there are predicates appearing only in $effect^+$ (we assume that such predicates do not appear in $precond$ neither in $effect^-$). The second set consists of predicates which appear in $precond$ but not in $effect^-$ (not shown in Figure 2). Finally, there is a set of predicates from $precond$ that also appear in $effect^-$.

Description of planning problems

The diagram for describing planning problems can be seen in Figure 3. Rectangle nodes represent *objects* and eliptic nodes (*predicates*) help to describe the state of the world. Distinct colors of *predicate* nodes are used to distinguish whether the described state is an initial state of the world or a condition which has to be fulfilled in the goal state.

Features and restrictions

Program VIZ is restricted to simple STRIPS planning domains with typing. It covers the following key features:

- class inheritance in the language declaration
- n-ary predicates with possibility of overloading
- basic consistency checking (e.g. missing predicate arguments, inconsistencies caused by changes in the language declaration)
- export of diagrams as `.png` images
- export and import from/to XML (in special format)
- export into PDDL (import from PDDL is not yet supported)

Conclusions

The presented system provides a comprehensive graphical interface which can assist users when designing simple planning domains through their visual representation. It can be also used for educational purposes. The proposed concept can be extended to allow design of more complex domains (e.g. functional symbols, conditional effects). Future development will be focused on the process of knowledge extraction from the informal problem description. VIZ is available from *http://clp.mff.cuni.cz/Viz.html*.

Acknowledgements

The research is supported by the Czech Science Foundation under the contract P103/10/1287.

We would like to thank Roman Barták for help with proof reading of this paper.

References

Simpson, R.M.; Kitchin, D.E.; McCluskey, T.L.:*Planning domain definition using GIPO*. Knowledge Engineering Review, 22 (2): 117-134 (2007)

Vaquero, T. S.; Romero, V. M. C.; Tonidandel, F.; Silva, J. R.:*itSIMPLE 2.0: An Integrated Tool for Designing Planning Domains.* In Proceedings of International Conference on Automated Planning & Scheduling (ICAPS 2007), pp. 336-343, AAAI Press (2007)

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; Wilkins, D.: *PDDL - the planning domain definition language.* Technical report, Yale Center for Computational Vision and Control (1998)

Slaney, J.K.; Thiébaux, S.:*Blocks World revisited.* Artif. Intell. (AI) 125(1-2):119-153 (2001)