AI Planning for Autonomy

Problem Set IV: PDDL and General Heuristics

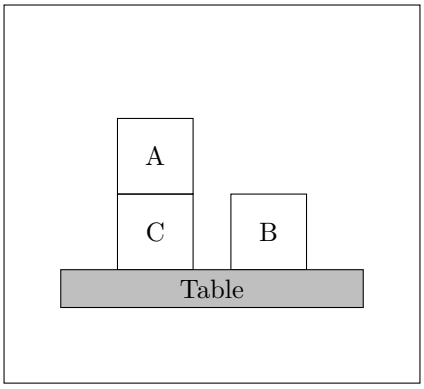

Figure 1: A blocks-world problem.

In blocks-world, the agent's aim is to stack the blocks in one tower with A on B and B on C. The agent can hold up to one block at a time and can put blocks down on the table, or another block.

1. Model Blocks-World as a STRIPS problem $P = \langle F, O, I, G \rangle$: define the set of facts F, the set of operators O, the goal facts G and the initial facts I. You must also define the *pre*, *add*, and *del* functions.

2. There are several important classes of domain-independent heuristics. Recall the delete-relaxed based heuristics from Lectures:

- What is a (relaxed) planning graph?
- Compute $h^{add}(s_0)$ for this blocks-world problem.
- Compute $h^{max}(s_0)$ for this blocks-world problem.

3. Implement your STRIPS model in PDDL. Use http://editor.planning.domains to test your model.

A PDDL implementation is split between two files: a domain file (sometimes an "operator" file) and a problem file (sometimes a "fact" file).

The example TSP of Australia from Nir's lectures is implemented in PDDL below.

See http://www.hakank.org/pddl/ for more examples.

```
(define (domain tsp)
   (:requirements :typing)
   (:types node)
   ;; Define the facts in the problem
   ;; "?" denotes a variable, "-" a type
   (:predicates (move ?from ?to - node)
                (at ?pos - node)
                (connected ?start ?end - node)
                (visited ?end - node))
   ;; Define the action(s)
   (:action move
             :parameters (?start ?end - node)
             :precondition (and (at ?start)
                                (connected ?start ?end))
             :effect  (and (at ?end)
                           (visited ?end)
                           (not (at ?start)))))
```

Figure 2: tsp-domain.pddl

```
(define (problem tsp-01)
   (:domain tsp)
   (:objects Sydney Adelade Brisbane Perth Darwin - node)
   ;; Define the initial situation
   (:init (connected Sydney Brisbane)
          (connected Brisbane Sydney)
          (connected Adelade Sydney)
          (connected Sydney Adelade)
          (connected Adelade Perth)
          (connected Perth Adelade)
          (connected Adelade Darwin)
          (connected Darwin Adelade)
          (at Sydney))
   (:goal
       (and (at Sydney)
            (visited Sydney)
            (visited Adelade)
            (visited Brisbane)
            (visited Perth)
            (visited Darwin))))
```

Figure 3: tsp-problem.pddl