Massachusetts Institute of Technology

16.410-13 Principles of Autonomy and Decision Making

Problem Set #6 (distributed 10/22/04, revised 10/26/04)

Paper solutions due no later than 5pm on Friday, 10/29/04. Please give solutions to the course secretary, Brian O' Conaill, at his desk outside of 33-330.

Note: Problem solutions will be posted on the web at 5pm on Friday, 10/29/04, hence, no late problem sets will be accepted.

Note Problem Set Revision:

In order to simplify the solution of this problem, this revision replaces the original pair of goals:

```
(rocket-fired)
(off-rocket)
```

with:

```
(rocket-fired)
(fuel-flowing)
```

The plan graphs in your solution should not need to contain more than two action layers and three proposition layers.

Objective

To exercise your understanding of the plan graph representation and plan extraction process, based on the GraphPlan algorithm.

Background

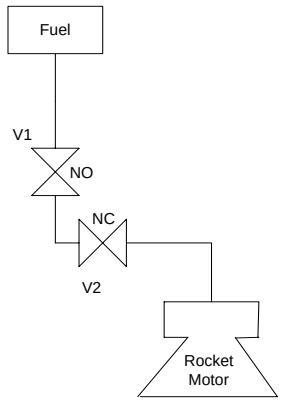
Handouts from lecture and from AIMA Chapter 11: "Planning." AIMA denotes "Artificial Intelligence: A Modern Approach: 2nd Edition" by Russell and Norvig.

Planning For Space Operations

A critical stage of many deep space probe missions is orbital insertion. One of the most ambitious autonomy demonstrations to date has been robust mission planning, execution and failure recovery for Saturn Orbital Insertion (SOI). During SOI it is essential that the main engine be commanded reliably under failure. In this problem we consider the problem of automatically planning a control sequence for a simple rocket engine system. We address execution and failure recovery later in this course.

Consider an extremely simple rocket engine system, shown at the top of the next page. To fire the engine, fuel must be flowing to it. This flow is controlled by valves V1 and V2, which are pyro valves. Pyro valves are initially in one particular state (open or closed). An explosive bolt can be fired that switches a pyro valve to its other state. Thus,

an important disadvantage of a pyro valve (with respect to a typical electrically activated on-off valve) is that a pyro valve can switch states only once. The advantage of using a pyro valve is that it is extremely reliable. It will stay in its initial state until it is fired. When it is fired, it is guaranteed to switch to the opposite state, where it will remain. In the following diagram, valve V1 is initially open (indicated by NO for normally open). Firing closes it. Valve V2 is initially closed (NC for normally closed). Firing opens it.



We formulate the problem using the STRIPS plan representation (the STRIPS representation is introduced in the lecture notes and Ch. 11 of AIMA). Our problem is to generate a command sequence that fires and turns off (powers down) the rocket engine, given that initially v2 is closed, v1 is open and the rocket is off. The initial and goal conditions in STRIPS are:

Initial Conditions:

```
(closed-v2)
(open-v1)
(off-rocket)
```

Goal:

```
(rocket-fired)
(fuel-flowing)          [note: goal changed to simplify problem]

~~(off-rocket)~~
```

We define the operations of firing pyro valves v1 and v2 through the plan operators fire-v1 and fire-v2, given below. Note that fire-v1 moves v1 from open to closed, and can only be executed when v2 is open. Likewise, fire-v2 moves v2 from closed to open, and can only be executed when v1 is open.

Operators:

```
(:operator fire-v1
 (:precondition
  (open-v2) (open-v1))
 (:effect
  (not open-v1)
  (closed-v1)
  (not fuel-flowing)
  (fuel-not-flowing)))

(:operator fire-v2
 (:precondition
  (closed-v2) (open-v1))
 (:effect
  (not closed-v2)
  (open-v2)
  (not fuel-not-flowing)
  (fuel-flowing)))
```

In addition, we introduce operators for firing and shutting down the rocket. The engine can only be fired if it is off and fuel is flowing. The engine can only be shut off if the rocket is on, but fuel is not flowing.

```
(:operator fire-rocket
  (:precondition
    (fuel-flowing) (off-rocket))
  (:effect
    (not off-rocket)
    (on-rocket)
    (rocket-fired)))

(:operator shut-off-rocket
  (:precondition
    (on-rocket) (fuel-not-flowing))
  (:effect
    (not on-rocket)
    (off-rocket)))
```

Problem 1.

Draw the plan graph for this problem, beginning with the initial state, and expanding levels until a level appears that contains all goal variables. Ignore mutex relations for now.

Problem 2.

Draw a plan graph that includes mutex relations for both actions and variables. The last level for this graph must include all goal variables with no mutex relations between any of them.

Problem 3.

Extract a plan solution from the plan graph of Problem 2 using the backward search algorithm described in lecture. Indicate each step including backtracking, if any is necessary. Note that, if your plan graph is correct, you should only be exploring a plan graph with two action layers.