Inf 2D Coursework 2

Planning in PDDL
Important Dates

• Deadline: 3pm Thursday 29th March 2018
• Drop-in Labs: Fridays 11am - 1pm Appleton Tower 6.06
Coursework Goals

- Formalize a reasonably sized planning problem
- Balance trade-offs in model design
- Actually implement and debug some PDDL
Assignment Outline

- Part 1a - Formalize problem in PDDL
- Part 1b - Backward State Space Search
- Parts 2 and 3 - Implement model in PDDL editor
Action(\textit{Move}(b, x, y)) :

\textbf{PRECOND} : \textit{On}(b, x) \land \textit{Clear}(b) \land \textit{Clear}(y) \land \textit{Block}(b) \land \textit{Block}(y) \land \\
(b \neq x) \land (b \neq y) \land (x \neq y)

\textbf{EFFECT} : \textit{On}(b, y) \land \textit{Clear}(x) \land \neg \textit{On}(b, x) \land \neg \textit{Clear}(y)
Backwards Search

Initial
\[ \text{On}(A, \text{Table}) \land \text{On}(B, \text{Table}) \land \text{On}(C, \text{Table}) \land \text{Block}(A) \land \text{Block}(B) \land \text{Block}(C) \land \text{Clear}(A) \land \text{Clear}(B) \land \text{Clear}(C) \]

Goal
\[ \text{On}(A, B) \land \text{On}(B, C) \]
Backwards Search

- \( g_1 = \text{On}(A, B) \land \text{On}(B, C) \)
Backwards Search

- $g_1 = \text{On}(A, B) \land \text{On}(B, C)$
- Available Actions: Move
Backwards Search

- $g_1 = \text{On}(A, B) \land \text{On}(B, C)$
- Available Actions: $Move$
- Choose: $Move(B, x, C)$
Backwards Search

- $g_1 = \text{On}(A, B) \land \text{On}(B, C)$
- Available Actions: Move
- Choose: $\text{Move}(B, x, C)$
- $g_2 = \text{On}(B, x) \land \text{Clear}(B) \land \text{Clear}(C) \land \text{Block}(B) \land \text{Block}(C) \land B \neq x \land B \neq C \land x \neq C$
Backwards Search

- $g_1 = On(A, B) \land On(B, C)$
- Available Actions: Move
- Choose: $Move(B, x, C)$
- $g_2 = On(B, x) \land Clear(B) \land Clear(C) \land Block(B) \land Block(C) \land B \neq x \land B \neq C \land x \neq C$
- Available actions: Move
Backwards Search

1.  $g_1 = \text{On}(A, B) \land \text{On}(B, C)$
2.  Available Actions: Move
3.  Choose: Move($B, x, C$)
4.  $g_2 = \text{On}(B, x) \land \text{Clear}(B) \land \text{Clear}(C) \land \text{Block}(B) \land \text{Block}(C) \land B \neq x \land B \neq C \land x \neq C$
5.  Available actions: Move
6.  Choose: Move($A, y, B$)
• \( g_1 = \text{On}(A, B) \land \text{On}(B, C) \)

• Available Actions: Move

• Choose : \( \text{Move}(B, x, C) \)

• \( g_2 = \text{On}(B, x) \land \text{Clear}(B) \land \text{Clear}(C) \land \text{Block}(B) \land \text{Block}(C) \land B \neq x \land B \neq C \land x \neq C \)

• Available actions : Move

• Choose : \( \text{Move}(A, y, B) \)

• \( g_3 = \text{On}(B, x) \land \text{Clear}(B) \land \text{Clear}(C) \land \text{Block}(B) \land \text{Block}(C) \land B \neq x \land B \neq C \land x \neq C \land \text{On}(A, y) \land \text{Clear}(A) \land \text{Block}(A) \land A \neq y \land A \neq B \land B \neq y \)
Backwards Search

• $g_1 = On(A, B) \land On(B, C)$

• Available Actions: Move

• Choose : $Move(B, x, C)$

• $g_2 = On(B, x) \land Clear(B) \land Clear(C) \land Block(B) \land Block(C) \land B \neq x \land B \neq C \land x \neq C$

• Available actions : Move

• Choose : $Move(A, y, B)$

• $g_3 = On(B, x) \land Clear(B) \land Clear(C) \land Block(B) \land Block(C) \land B \neq x \land B \neq C \land x \neq\n\hfill \neg C \land On(A, y) \land Clear(A) \land Block(A) \land A \neq y \land A \neq B \land B \neq y$

• $g_3$ satisfies initial state by substituting $\{x = Table, y = Table\}$
Backwards Search

- $g_1 = \text{On}(A, B) \land \text{On}(B, C)$
- Available Actions: Move
- Choose: Move($B, x, C$)
- $g_2 = \text{On}(B, x) \land \text{Clear}(B) \land \text{Clear}(C) \land \text{Block}(B) \land \text{Block}(C) \land B \neq x \land B \neq C \land x \neq C$
- Available actions: Move
- Choose: Move($A, y, B$)
- $g_3 = \text{On}(B, x) \land \text{Clear}(B) \land \text{Clear}(C) \land \text{Block}(B) \land \text{Block}(C) \land B \neq x \land B \neq C \land x \neq C \land \text{On}(A, y) \land \text{Clear}(A) \land \text{Block}(A) \land A \neq y \land A \neq B \land B \neq y$
- $g_3$ satisfies initial state by substituting \{\(x = \text{Table}\), \(y = \text{Table}\)\}
- Done!
editor.planning.domains
(define (domain blocks-world)
  (:requirements :adl)
  (:types table block)
  (:predicates
    (On ?x - block ?y - object)
    (Clear ?b - object)
  )
  (:constants Table - table)

  (:action MOVE
    :parameters (?b - block ?x - object ?y - block)
    :effect (and (On ?b ?y) (Clear ?x) (not (On ?b ?x)) (not (Clear ?y)))
  )

  (:action MOVE-TO-TABLE
    :parameters (?b - block ?x - block)
    :precondition (and (On ?b ?x) (Clear ?b) (not (= ?b ?x)))
    :effect (and (On ?b Table) (Clear ?x) (not (On ?b ?x)))
  )
)
```
(define (problem block-problem)
  (:domain blocks-world)
  (:objects
    A - block
    B - block
    C - block
  )
  (:init
    (On A Table)
    (On B Table)
    (On C Table)
    (Clear A)
    (Clear B)
    (Clear C)
  )
  (:goal (and
    (On A B)
    (On B C)
  ))
)```
Found Plan (output)

(move b table c)

(move a table b)

(:action move
 :parameters (b table c)
 :precondition
  (and
   (on b c)
   (clear b)
   (clear ?y)
   (not
    (= b c)
   )
   (not
    (= b ?y)
   )
   (not
    (= c ?y)
   )
  )
 :effect
  (and
   (on b ?y)
   (clear ?y)
   (not
    (on b c)
   )
   (not
    (clear ?y)
   )
  )
)
Action(Move(b, x, y)):

PRECOND : On(b, x) ∧ Clear(b) ∧ Clear(y) ∧ Block(b) ∧ Block(y) ∧ (b ≠ x) ∧ (b ≠ y) ∧ (x ≠ y)

EFFECT : On(b, y) ∧ Clear(x) ∧ ¬On(b, x) ∧ ¬Clear(y)

(:action MOVE
 :parameters ( ?b - block
       ?x - object
       ?y - block
 )
 :precondition (and
       (On ?b ?x)
       (Clear ?b)
       (Clear ?y)
       (not (= ?b ?x))
       (not (= ?b ?y))
       (not (= ?x ?y))
 )
 :effect (and
       (On ?b ?y)
       (Clear ?x)
       (not (On ?b ?x))
       (not (Clear ?y))
 ))
examples/blocks-world-domain.pddl
examples/blocks-world-problem.pddl
(define (domain bartending)
  (:requirements :adl)
  (:types
    bartender
    location
    ;; Fill in additional types here
  )
  (:constants
    ;; You should not need to add any additional constants
    Agent - bartender
    BAR - location
  )
  (:predicates
    ;; Example:
    ;; (Contains ?x - object ?c - container)
  )
  ;;;; Action Template - Delete and fill in own actions ;;;;
  (:action dummy-action
    ; :parameters (?obj - object)
    ; :precondition (and
    ;   (dummy-pred-1 ?obj)
    ;   (dummy-pred-2 ?obj)
    ; )
    ; :effect (and
    ;   (not (dummy-pred-1 ?obj))
    ;   (dummy-pred-3 ?obj)
    ; )
  )
)
Inf2d-ass2-s0929508/
answers.txt
domain-solution21.pddl
problem-solution22.pddl
problem-solution23.pddl
domain-solution31.pddl
problem-solution32.pddl
Compress

tar cvzf Inf2d-ass2-s0929508.tar.gz Inf2d-ass2-s0929508

Submit

submit inf2d cw2 Inf2d-ass2-s0929508.tar.gz
Questions?