Planning Representation: PDDL2 & 3

Dra. Mª Dolores Rodríguez Moreno

Objectives

Specific Objectives

- Model in PDDL 2 & PDDL 3

Source

- PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains. JAIR, 20:61-124, 2003
- Plan Constraints and Preferences in PDDL3. Proc. of ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning, 2006

Outline

- **PDDL 2.X & 3.X syntax**
- Gripper domain
- Conclusions

PDDL 2.X syntax: Domain

```
(define (domain name)
  (:requirements <require-key> :durative-actions :fluents)
  (:types <typed_list (name)>)
<PDDL list of predicates in the domain>
<PDDL list of functions in the domain>

<PDDL code for first action>
...
<PDDL code for last action>
)
```

PDDL 2.X syntax: Actions (I)

(:durative-action <action name>
 :parameters (<list>)
 :duration (= ?duration <number> or (<predicate list>))
 :condition (and (at start/at end/over all (<predicate list>))
 :effect (and (at start/at end/over all (<predicate list>))
)

PDDL 2.X syntax: Actions (II)

- To assign
  - assign (not =)
- To add
  - increase
- To subtract
  - decrease

PDDL 2.X syntax: Problem

```
(define (problem <problem name>)
  (:domain <domain name>)
  <PDDL code for objects>
  <PDDL code for initial state>
  (= (predicate <parameter list>) number)
  <PDDL code for goal specification>
(:metric minimize (predicate))
)
```

PDDL 3.1 syntax: Domain

```
(define (domain name)
  (:requirements <require-key> :constraints :preferences)
  (:types <typed_list (name)>)
<PDDL list of predicates in the domain>
<PDDL list of functions in the domain>
<PDDL code for first action>
...
<PDDL code for last action>
)
```

PDDL 3.1 syntax: Problem

(define (problem <problem name>)

...

(:goal (and ...

(preference [name] <GD>)

(:constraints

(at end <GD>) | (always <GD>) | (sometime <GD>) | (within <num> <GD>) | (at- most-once <GD>) | (sometime-after <GD> <GD>) | (sometime-before <GD> <GD>) | (always-within <num> <GD> <GD>) | (hold-during <num> <num> <GD> | (hold-after <num> <GD> | ...

(:metric

(is-violated <preference-name>)

))

PDDL syntax: Tip

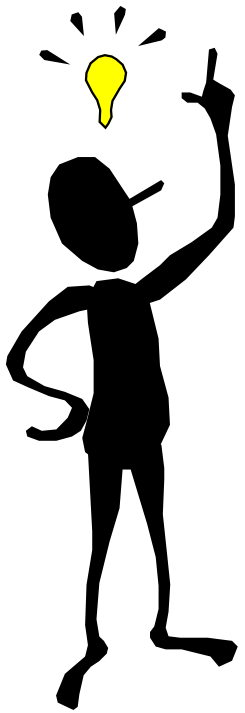
- Use the extensión .pddl for your files
- Name the files as the domain and problem files

d-prueba.pddl

(define (domain d-prueba) → **Domain**

p1.pddl

(define (problem p1) → **Problem**



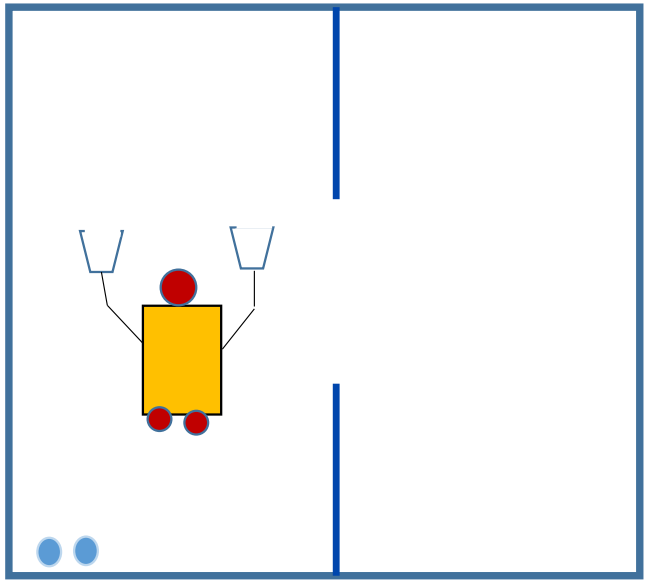
Outline

- PDDL 2.X & 3.X syntax
- **Gripper domain**
- Conclusions

Gripper domain (I)

- There is a robot that can move between two rooms and pick up or drop balls (2) with either of his two arms. Initially, all balls and the robot are in the kitchen. We want the balls to be in the living room.
- As a preference, at the end

  the robot is in the living room



Gripper domain (II)

- **Requirements**
  - Durative-actions
  - Fluents
  - Preferences

```
(define (domain gripper-tiempos)
  (:requirements :typing :fluents :durative-actions :constraints :preferences)
  (:types room ball)
  (:predicates  (at-robby ?r - room )
                (at-ball ?b - ball ?r - room)
                (free ?g - gripper)
                (carry ?g - gripper ?b - ball))
 (:functions
   (dur-ball ?ball - ball)
 )
```

Gripper domain: actions (I)

- **Operator *Move* (as in the previous domain)**
  - Description: The robot can move from x to y
  - Precondition: ROOM(x), ROOM(y) and at-robby(x) are true
  - Effect: at-robby(y) becomes true
    at-robby(x) becomes false
    Everything else doesn't change

Gripper domain: actions (II)

- Operator *Move* in PDDL (domain file)

```
11  (:action move
12      :parameters (?x ?y)
13      :precondition (and (ROOM ?x)
14                         (ROOM ?y)
15                         (at-robby ?x)
16                    )
17      :effect (and (at-robby ?y)
18                   (not (at-robby ?x))
19              )
20  )
21
```

Gripper domain: actions (III)

- **Operator *Pick-up***
  - Description: The robot can pick up b in r with g
  - Duration
  - Precondition: BALL(b), ROOM(r), GRIPPER(g), at-ball(b, r), at-robby(r) and free(g) are true
  - At start/at end/over all
  - Effect: carry(g, b) becomes true
    at-ball(b, r) and free(g) become false
    Everything else doesn't change
    At start/at end/over all

Gripper domain: actions (IV)

```
(:durative-action pick-up
    :parameters (?ball - ball ?room - room ?gripper - gripper)
    :duration (= ?duration (dur-ball  ?ball))
    :condition (and   (at start (at-ball ?ball ?room))
                      (over all  (at-robby ?room))
                      (at start  (free ?gripper))
               )
    :effect (and (at end (carry ?gripper ?ball))
                 (at start (not (at-ball ?ball ?room)))
                 (at start (not (free ?gripper)))
            )
)
```

Gripper domain: actions (V)

- **Operator *Drop***
  - Description: The robot can drop b in r from g
  - Duration
  - Precondition: BALL(b), ROOM(r), GRIPPER(g), carry(g, b), at-robby(r) are true
  - At start/at end/over all
  - Effect: carry(g, b) becomes false
    at-ball(b, r) and free(g) become true
    Everything else doesn't change
    At start/at end/over all

Gripper domain: actions (VI)

```
(:durative-action drop
    :parameters (?ball - ball ?room - room ?gripper - gripper)
     :duration (= ?duration (dur-ball  ?ball))
    :condition (and (at start (carry ?gripper ?ball))
                    (over all(at-robby ?room))
                )
    :effect (and (at end (at-ball ?ball ?room))
                 (at end (free ?gripper))
                 (at start  (not (carry ?gripper ?ball)))
            )
)
```

Gripper domain: Initial state

```
1  ;; Problem
2  (define (problem gripper-tiempos-p1)
3     (:domain gripper-tiempos)
4     (:objects kitchen living - room
5               Z1 Z2 - ball
6               left right)
7     (:init (at-robby kitchen)
8            (free left)           (free right)
9            (at-ball Z1 kitchen)  (at-ball Z2 kitchen)
10           (= (dur-ball Z1) 10)
11           (= (dur-ball Z2) 20)
12    )
13
```

Gripper domain: goals

```
15      (:goal (and (at-ball Z1 living)
16                  (at-ball Z2 living)
17             )
18      )
19      (:constraints (and
20        (preference PLACED (at end (at-robby living)))))
21
```

Outline

- PDDL 2.X & 3.X syntax
- Gripper domain
- **Conclusions**

Conclusions

- PDDL 2.1 follows non-conservative time model
  - What (actions) + When (execution time)
- PDDL 3.1 represents plan with soft constraints & goals
  - Best quality plan satisfy “as much as possible” the soft constraints & goals