

Planning Representation: PDDL₂ & 3

Dra. M^a Dolores Rodríguez Moreno

Objectives

Specific Objectives

- Model in PDDL₂ & PDDL₃

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/overall (<predicate list>))  
:effect (and ( at start/at end/overall (<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

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



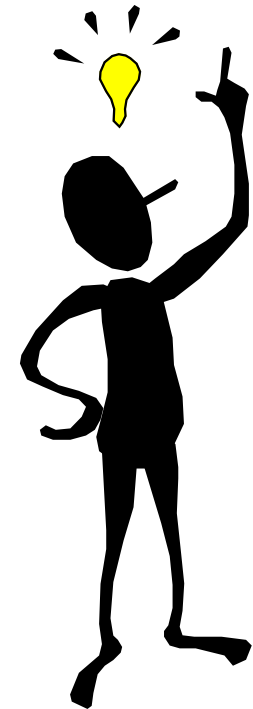
d-prueba.pddl



(define (problem **p1**)



p1.pddl

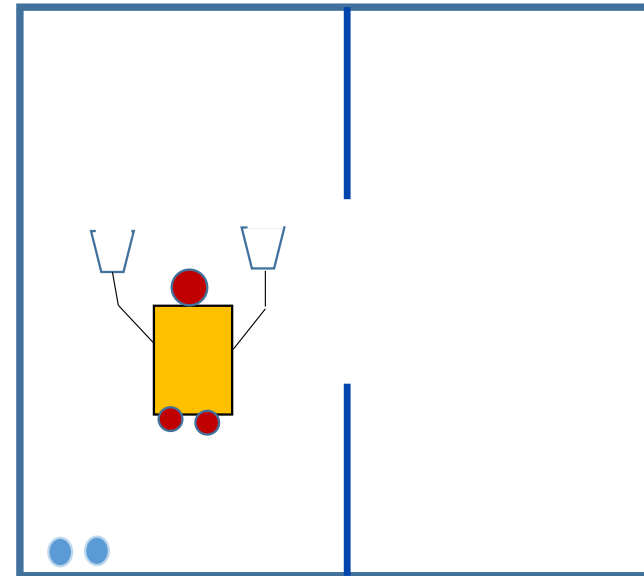


Outline

- 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

```
1  ;;; Gripper with time
2  (define (domain gripper-tiempos)
3    (:requirements :typing :fluents :durative-actions)
4    (:types room ball)
5    (:predicates (at-robby ?r)
6                  (at-ball ?b ?r)
7                  (free ?g)
8                  (carry ?g ?b))
9  ( :functions
10    (dur-ball ?ball - ball)
11  )
```

Gripper domain: actions (I)

- Operator *Move* (as in the previous domain)
 - Description: The robot can move from x to y
 - Precondition: $\text{ROOM}(x)$, $\text{ROOM}(y)$ and $\text{at-robby}(x)$ are true
 - Effect: $\text{at-robby}(y)$ becomes true
 $\text{at-robby}(x)$ 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: $BALL(b)$, $ROOM(r)$, $GRIPPER(g)$, $at-ball(b, r)$, $at-robot(r)$ and $free(g)$ are true
- At start/at end/overall
- Effect: $carry(g, b)$ becomes true
 $at-ball(b, r)$ and $free(g)$ become false
 Everything else doesn't change
 At start/at end/overall

Gripper domain: actions (IV)

```
22 ▾ (:durative-action pick-up
23     :parameters (?ball - ball ?room - room ?gripper)
24     :duration (= ?duration (dur-ball ?ball))
25     :condition (and (at start (at-ball ?ball ?room))
26                     (at start (at-robby ?room))
27                     (at start (free ?gripper))
28                    )
29     :effect (and (at end (carry ?gripper ?ball))
30                 (at start (not (at-ball ?ball ?room)))
31                 (at start (not (free ?gripper))))
32     )
33 )
```

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/overall
- Effect: $carry(g, b)$ becomes false
 $at-ball(b, r)$ and $free(g)$ become true
 Everything else doesn't change
 At start/at end/overall

Gripper domain: actions (VI)

```
34 ▾ (:durative-action drop
35     :parameters (?ball - ball ?room - room ?gripper)
36     :duration (= ?duration (dur-ball ?ball))
37     :condition (and (at start (carry ?gripper ?ball))
38                    (at start (at-roby ?room))
39                    )
40     :effect (and (at end (at-ball ?ball ?room))
41                (at end (free ?gripper))
42                (at start (not (carry ?gripper ?ball))))
43     )
44 )
```

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