

Planning Representation: Time & preferences

Dra. M^a Dolores Rodríguez Moreno

Objectives

Specific Objectives

- Model in PDDL 2.1 & PDDL 3.1
- Run SoA planners

Source

- Fox & Long. PDDL2.1: An Extension to PDDL for Expressing Temporal Problems. Journal of AI Research 20 (2003) 61-124
- Gerevini & Long. Plan Constraints and Preferences in PDDL₃ (2005) Tech. Report, Dpt. of Electronics Automation, University of Brescia
- Eva Onaindia De La Rivaherrera. Planificación Automática. Videos. UPV. <https://media.upv.es/>

Outline

- **Introduction**
- Temporal Planning
- PDDL 2.X syntax
- Airport problem
- Preferences in Planning
- PDDL 3.X syntax
- PDDL₃.X examples
- Conclusions

Introduction (I)

- Classical planning is restrictive
 - Implicit time assumption
 - Actions no duration
- Need new features for real problems
 - Time
 - Resources
 - Multi-objective
- PDDL is extended: PDDL 2.1

Introduction (II)

- PDDL 2.1 Levels:
 - Level 1: STRIPS version
 - Level 2: the numeric extensions
 - Level 3: the addition of discretised durative actions
 - Level 4: continuous durative actions
 - Level 5: comprised all of the extensions of pddl2.1 and additional components to support the modelling of spontaneous events and physical processes

Introduction (III)

- New
 - Numeric expressions
 - Durative actions
 - Metrics: evaluate the quality of a plan
 - Continuous changes

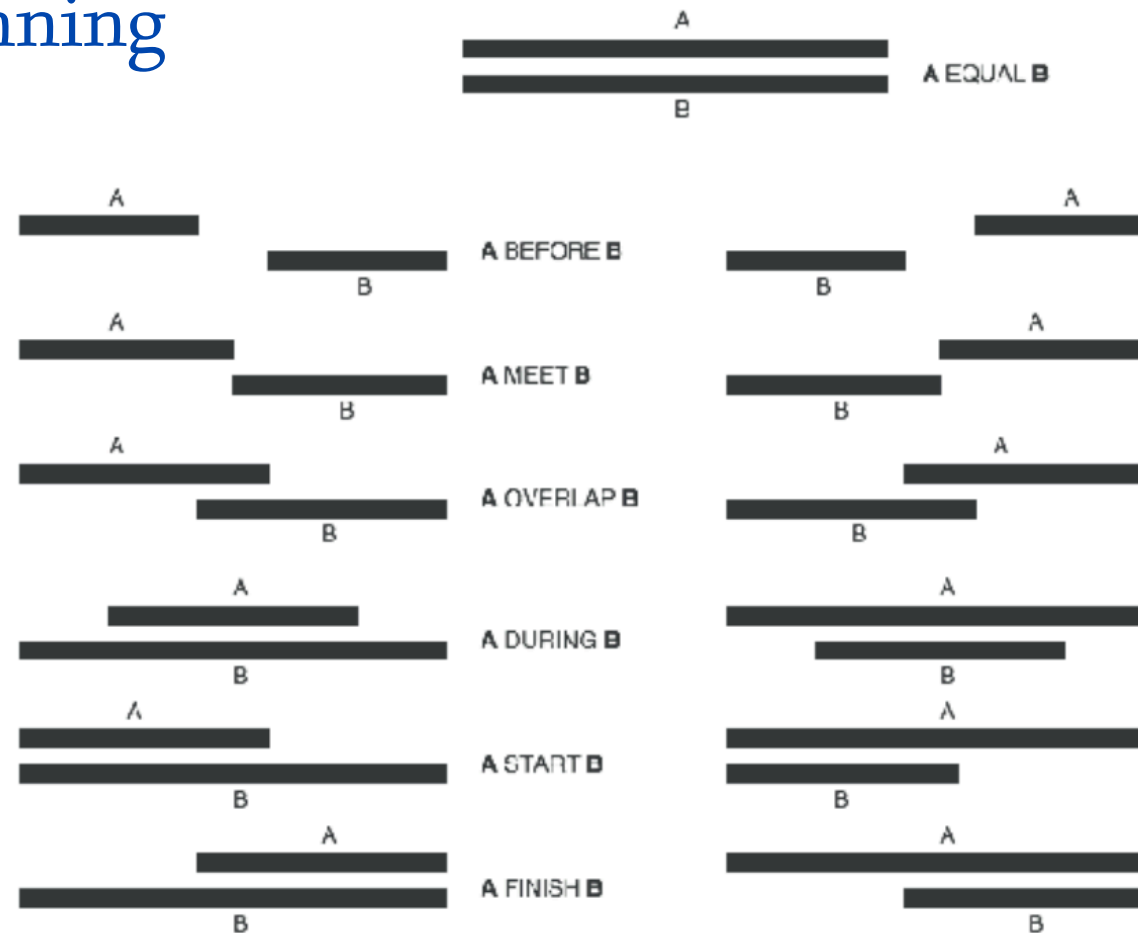
Outline

- Introduction
- **Temporal Planning**
- PDDL 2.X syntax
- Airport problem
- Preferences in Planning
- PDDL 3.X syntax
- PDDL₃.X examples
- Conclusions

Temporal planning

- Sequential planning is not adequate
- What (actions) + When (execution time)
 - Actions synchronization
 - Actions overlapping
 - New optimization criteria
 - Planning steps vs. plan duration (*makespan*)

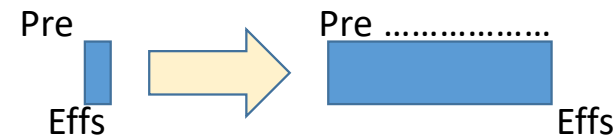
Temporal planning



Source: Flavio S. Corrêa da Silva

Temporal planning: conservative model (I)

- Easiest way: convert the existing model in a duration model (conservative time model)
- How
 - Preconditions are true at the beginning
 - Effects are true at the end



- Actions are not atomic, allow concurrency (iff no conflicts)
- We cannot know the state of variables in the problem

Temporal planning: conservative model (II)

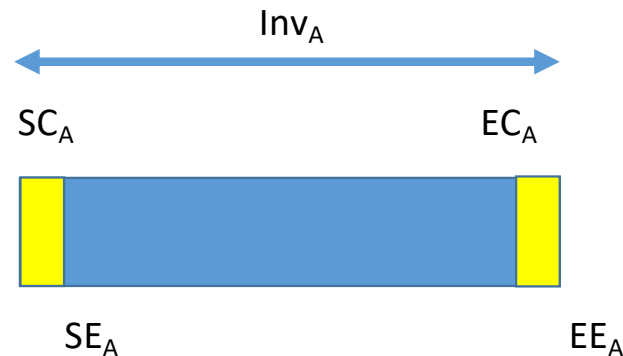
- What happen with pick-up action? Where is A during the process?



- The planner can assume that is all the time on the table (is NOT correct!!!)

Temporal planning: non-conservative model

- Consider when each predicate holds
 - StartCond: (SC_A)
 - EndCond: (EC_A)
 - StartEff: (SE_A)
 - EndEff: (EE_A)
 - Invariant: (Inv_A)



Outline

- Introduction
- Temporal Planning
- **PDDL 2.X syntax**
- Airport problem
- Preferences in Planning
- PDDL 3 syntax
- PDDL₃ examples
- Conclusions

PDDL 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 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 syntax: Actions (II)

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

PDDL 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))  
)
```

Outline

- Introduction
- Temporal Planning
- PDDL 2.X syntax
- **Airport problem**
- Preferences in Planning
- PDDL 3.X syntax
- PDDL₃.X examples
- Conclusions

Airport domain

- The domain consists of planes and passengers that travel from one city to another
- Model the duration of the actions and the fuel consumption
- Model 3 actions:
 - Board: a person on a plane that is in a city. As a result the person is not in the city and is on the plane
 - Debark: a person from an airplane. As a result the person is in the city, and is not on the plane
 - Fly: from one city to another. The fuel of the plane depends on the distance between the cities and the fuel ratio consumption of the plane. As a precondition, it should be verified $\text{Fuel} \geq (\text{distance-between-cities}) \times (\text{burn-fuel-ratio of the plane})$

Outline

- Introduction
- Temporal Planning
- PDDL 2.X syntax
- Airport problem
- **Preferences in Planning**
- PDDL 3.X syntax
- PDDL₃.X examples
- Conclusions

Introduction

- PDDL₂.X is still restrictive
 - Plan quality measured by plan size
 - Hard constraints on actions
 - Hard constraints on goals
- If not satisfied, NO plan!!
- Plan with soft constraints & goals
 - Best quality plan satisfy “as much as possible” the soft constraints & goals
- PDDL₂.X is extended: PDDL₃

Preferences in planning (I)

- With soft constraints and goals, can be useful to give priorities
 - Numerical weight representing the cost of its violation in a plan (metric)
- Transportation example
 - We would like that every airplane is used (instead of using only a few airplanes, because it is better to distribute the workload among the available resources and limit heavy usage)
 - Whenever a ship is ready at a port to load the containers it has to transport, all such containers should be ready at that port
 - We would like that at the end of the plan all trucks are clean and at their source location
 - We would like no truck to visit any destination more than once

Preferences in planning (II)

- Can express that certain plans are more preferred than others
 - *I prefer a plan where every airplane is used, rather than a plan using 100 units of fuel less,* which could be expressed by weighting a failure to use all the planes by a number 100 times bigger than the weight associated with the fuel use in the plan metric
 - *I prefer a plan where each city is visited at most once, rather than a plan with a shorter makespan,* which could be expressed by using constraint violation costs penalising a failure to visit each city at most once very heavily
 - *I prefer a plan where at the end each truck is at its start location, rather than a plan where every city is visited by at most one truck,* which could be expressed by using goal costs penalising a goal failure of having every truck at its start location more heavily than a failure of having in the plan every city visited by at most one truck

Outline

- Introduction
- Temporal Planning
- PDDL 2.X syntax
- Airport problem
- Preferences in Planning
- **PDDL 3.X syntax**
- PDDL₃.X examples
- Conclusions

PDDL 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 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>)
```

```
))
```

Outline

- Introduction
- Temporal Planning
- PDDL 2.X syntax
- Airport problem
- Preferences in Planning
- PDDL 3.X syntax
- **PDDL₃.X examples**
- Conclusions

PDDL₃.X examples: Preferences

- (preference VisitParis (forall (?x - tourist) (sometime (at ?x Paris))))
 - yields a violation count of 1 for (is-violated VisitParis), if at least one tourist fails to visit Paris
- (forall (?x - tourist) (preference VisitParis (sometime (at ?x Paris))))
 - yields a violation count equal to the number of people who failed to visit Paris
- (:goal (and (at package₁ London) (preference p₁ (clean truck₁))))

PDDL₃.X examples: Constraints

- Constraints can be used to weighted expressions in metrics
`(:metric minimize (+ (* 10 (fuel-used)) (is-violated VisitParis)))`
would weight fuel use as ten times more significant than violations of the VisitParis constraint
- Another example of multiple ones:
`(:constraints (and (preference p1 (always (clean truck1))) (preference p2 (and (at end (at package2 Paris)) (sometime (clean truck1)))) (preference p3 (...)) ...))`
- Combine metrics and preferences
`(:metric (+ (* 10 (is-violated p1)) (* 5 (is-violated p2)) (is-violated p3)))`

PDDL₃.X examples

- We want three jobs completed. We would prefer to take a coffee-break and that we take it when everyone else takes it (at coffee-time) rather than at any time. We would also like to finish reviewing a paper, but it is less important than taking a break. Finally, we would like to be finished so that we can get home at a reasonable time, and this matters more than finishing the review or having a sociable coffee break

Outline

- Introduction
- Temporal Planning
- PDDL 2.X syntax
- Airport problem
- Preferences in Planning
- PDDL 3.X syntax
- PDDL₃.X examples
- **Conclusions**

Conclusions

- No model of time in classical planning
- PDDL 2.1 follows non-conservative time model
- What (actions) + When (execution time)
 - Actions synchronization
 - Actions overlapping
 - New optimization criteria
 - Planning steps vs. plan duration (*makespan*)
- PDDL2.X is extended: PDDL3.X
- Represent plan with soft constraints & goals
 - Best quality plan satisfy “as much as possible” the soft constraints & goals