

# Planning Representation: Preferences in PDDL

Dra. M<sup>a</sup> Dolores Rodríguez Moreno

# Objectives

## Specific Objectives

- Model in PDDL 3.0
- Comparison with NASA planning language

## Source

- Gerevini & Long. Plan Constraints and Preferences in PDDL<sub>3</sub>. Technical Report, Department of Electronics for Automation, University of Brescia, Italy, August 2005

# Outline

- **Introduction**
- Preferences in Planning
- PDDL 3 syntax
- PDDL<sub>3</sub> examples
- XIDDL language
- XIDDL example
- Conclusions

# Introduction

- PDDL<sub>2</sub>.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
- PDDL<sub>2</sub>.X is extended: PDDL<sub>3</sub>

# Outline

- Introduction
- **Preferences in Planning**
- PDDL<sub>3</sub> syntax
- PDDL<sub>3</sub> examples
- XIDDL language
- XIDDL example
- Conclusions

# 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
- Preferences in Planning
- **PDDL 3 syntax**
- PDDL<sub>3</sub> examples
- XIDDL language
- XIDDL example
- 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>)
```

```
...
```

```
(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>) | ...
```

```
(is-violated <preference-name>)
```

```
)
```

# Outline

- Introduction
- Preferences in Planning
- PDDL 3 syntax
- **PDDL<sub>3</sub> examples**
- XIDDL language
- XIDDL example
- Conclusions

## PDDL<sub>3</sub> 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 package<sub>1</sub> London) (preference p<sub>1</sub> (clean truck<sub>1</sub>))))

# PDDL<sub>3</sub> 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)))

## PDDL<sub>3</sub> 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
- Preferences in Planning
- PDDL 3 syntax
- PDDL<sub>3</sub> examples
- **XIDDL language**
- XIDDL example
- Conclusions

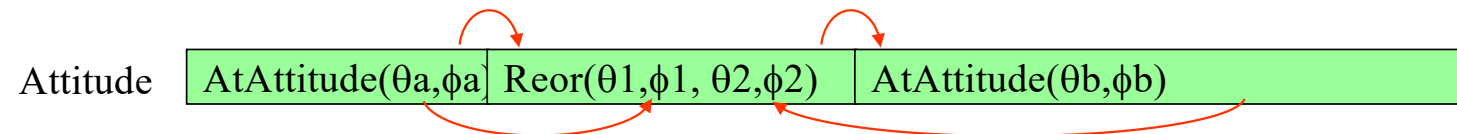
# XIDDL

- Declarative descriptions of legal operations of a system
- Models define:
  - Timelines: discrete state variables, populated over all time by predicates
  - Predicates/Tokens: parameterized state values that remain fixed over time intervals
  - Constraints: temporal and parameter restrictions between tokens/predicates



# XIDDL

- What do we mean by a model?
  - For example, a spacecraft reorientation maneuver
    - MET\_BY(Reor, AtAttitude) w/  $\theta_I = \theta_a, \phi_I = \phi_a$
    - MEETS(Reor, AtAttitude) w/  $\theta_2 = \theta_b, \phi_2 = \phi_b$



- Models written in XML-based language
- Enables automatic syntactic and semantic checking of the model

# XIDDL: domain (I)

- Timeline (TL) is a logical structure used to represent states over time
- Class is a set of TLs
- Types:
  - *Internal modes*: describes the internal state of the controlled subsystem. State transitions are not communicated outside
  - *Goal*: represents a system which exerts control over the agent itself
  - *Executable*: describes the state that will be communicated to the outside world when the state transition is at the current time

## XIDDL: domain (II)

- Tokens/predicates: define the possible values that TLs can have:

$$P(i_1, \dots, i_n \rightarrow m_1, \dots, m_k \rightarrow o_1, \dots, o_m; s)$$

Each  $i_i$ ,  $m_i$  y  $o_i$  represents: input arguments, mode and output argument

- Types:
  - *call\_args*: are passed to the external system
  - *internal\_modes*: set for internal reasoning and not to communicate out
  - *return\_args*: returned by the external subsystem
  - *return\_status*: returned by the external subsystem and used for close loop
- Constraints: temporal and parameter restrictions between tokens

## XIDDL: problem

- Specify what token exists in the initial state in each TL
- Generally the goals are loaded in a different module

# Outline

- Introduction
- Preferences in Planning
- PDDL 3 syntax
- PDDL<sub>3</sub> examples
- XIDDL language
- **XIDDL example**
- Conclusions

# Space Operator domain (PDDL)

```
(:action take_image
  :parameters (?s - satellite ?d - direction ?i - instrument ?m - mode)
  :precondition (and (calibrated ?i)
                     (on_board ?i ?s)
                     (supports ?i ?m)
                     (power_on ?i)
                     (pointing ?s ?d)
                     (power_on ?i))
  :effect (have_image ?d ?m))
```

# Space Operator problem (PDDL)

```
(:objects
  satelliteo satellite1 satellite2- satellite
  instrumento instrument1 instrument2 - instrument
  ...
  Phenomenon4 Star5 Phenomenon6 - direction)

(:init
  (supports instrumento thermographo)
  (calibration_target instrumento GroundStation2)
  ...
  (pointing satelliteo Phenomenon6))

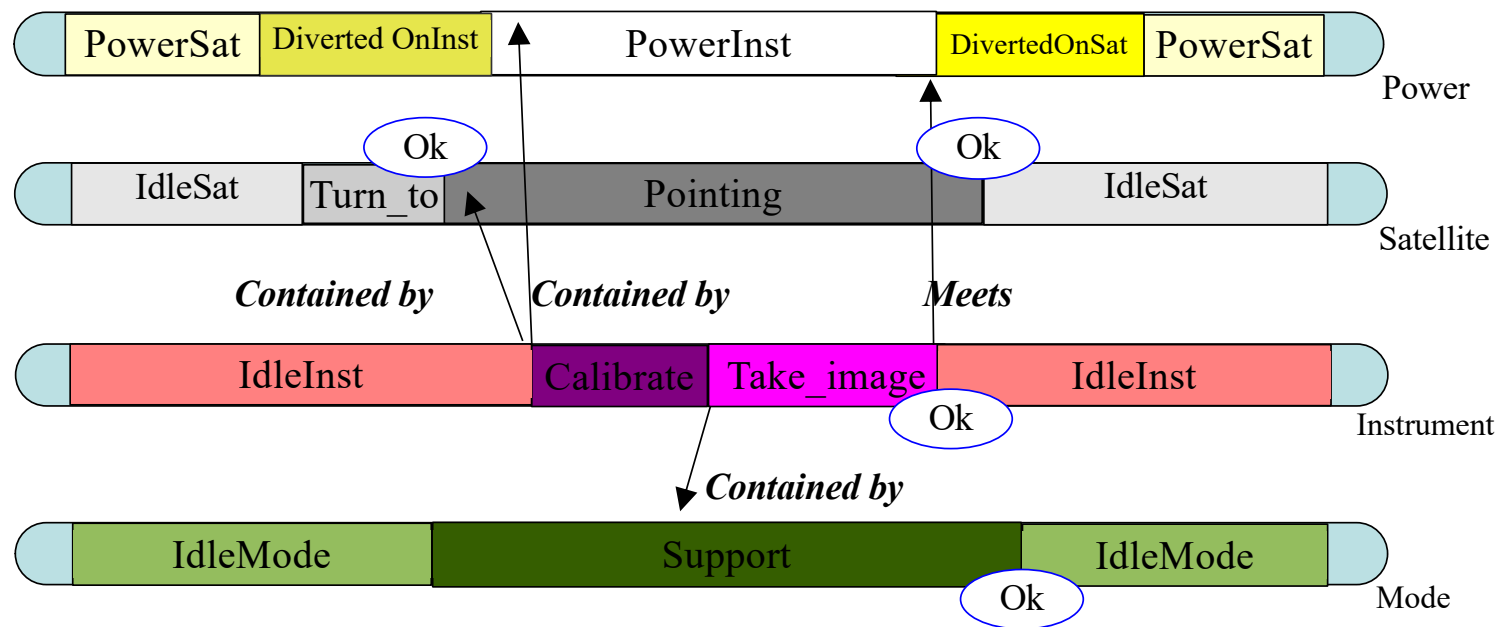
(:goal  (have_image Phenomenon4 thermographo)
        (have_image Star5 thermographo)
        (have_image Phenomenon6 thermographo))
```

# XIDDL example

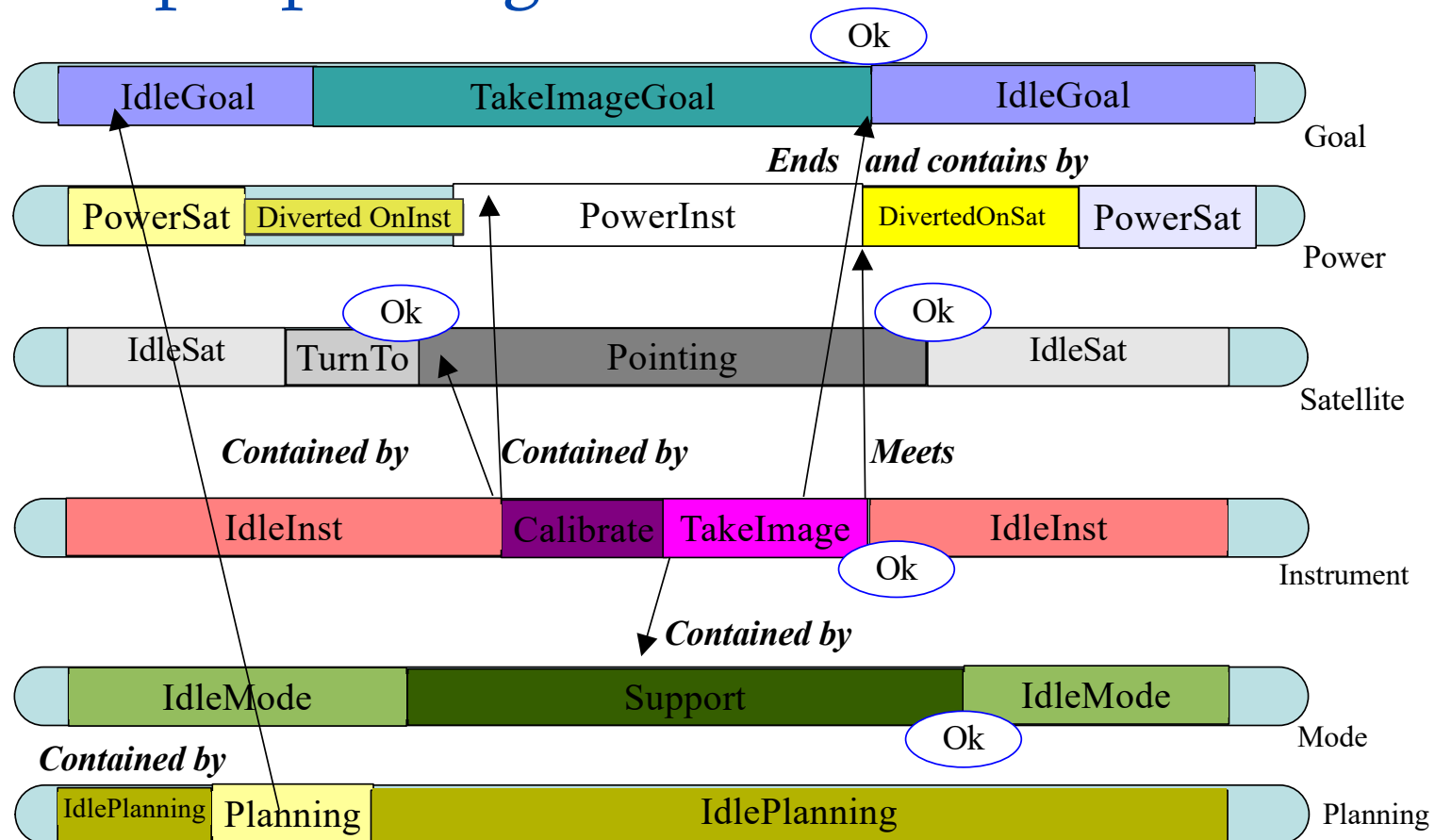
- How to represent this information?
- What system do I want to know its evolution?
- How many TLs do I need?



## XIDDL example: domain



# XIDDL example: plan & goals



# XIDDL: Domain

- TLs
  - satellite\_TL
  - instrument\_TL
  - power\_TL
  - mode\_TL

# XIDDL: Domain

- Procedures
  - satellite\_TL
    - Pointing(?s – satellite, ?d - direction)
    - TurnTo(?s – satellite, ?ds – direction, ?de - direction)
  - instrument\_TL
    - Calibrate(?s – satellite, ?d – direction, ?i - instrument)
    - TakeImage(?s – satellite, ?d – direction, ?i -instrument, ?m – mode)
  - power\_TL
    - DivertedOnSat(?s – satellite, ?i - instrument)
    - DivertedOnInst(?s – satellite, ?i - instrument)
    - PowerInst(?i – instrument)
    - PowerSat(?s – satellite)
  - mode\_TL
    - Support(?i -instrument, ?m – mode)

```
<!-- -->
<!-- TURN TO (Sat, Dir, NewDir) -->
<!-- -->

<define_procedure>
  <name>TurnTo</name>
  <call_args>
    <arg>
      <type>SatelliteValues</type>
      <name>satellite</name>
    </arg>
    <arg>
      <type>DirectionValues</type>
      <name>prevDir</name>
    </arg>
    <arg>
      <type>DirectionValues</type>
      <name>newDir</name>
    </arg>
  </call_args>
  <return_status>
    <type>ReturnResult</type>
    <name>resultTT</name>
    <flag>resultTTflag</flag>
  </return_status>
</define_procedure>
```

# XIDDL: Domain

- Compatibilities

```
<!--                                     -->
<!-- IdleSat meets TurnTo(s,pd,nd)    -->
<!--                                     -->

<define_compatibility>
  <master type="single">
    <class>SatelliteDomain</class>
    <attr>SatelliteTL</attr>
    <pred>IdleSat</pred>
  </master>
  <duration_bounds>
    <range>
      <lb>*latency*</lb>
      <ub>_plus_infinity_</ub>
      <!-- <ub>*dur*</ub> -->
    </range>
  </duration_bounds>
  <subgoals>
    <meets>
      <class>SatelliteDomain</class>
      <attr>SatelliteTL</attr>
      <pred>TurnTo</pred>
    </meets>
  </subgoals>
</define_compatibility>
```

# XIDDL: Objects (I)

```
<define_label_set>  
  <name>SatelliteValues</name>  
  <value>satelliteo</value>  
  <value>satellitei</value>  
</define_label_set>
```

```
<define_label_set>  
  <name>InstrumentValues</name>  
  <value>instrumento</value>  
  <value>instrumenti</value>  
</define_label_set>
```

## XIDDL: Objects (II)

```
<define_label_set>
  <name>DirectionValues</name>
  <value>GroundStation1</value>
  ...
  <value>Phenomenon6</value>
</define_label_set>
<define_label_set>
  <name>ModeValues</name>
  <value>image1</value>
  <value>spectrograph2</value>
  <value>thermographo</value>
</define_label_set>
```

## XIDDL: Initial State

```
Object_Timelines SpaceOpDomain SpaceOpInstance  
(  
  SatelliteTL ([o] IdleSat() ...)  
  InstrumentTL ([o] IdleInst() ...)  
  PowerTL ([o] PowerSat ( satelliteo) ...)  
  ModeTL ([o] IdleMode() ...)  
)
```



# XIDDL: IS & Goal

```
Object_Timelines SatelliteDomain SatelliteInstance (  
  SatelliteTL ([o] IdleSat() ...) ...  
  ModeTL ([o] IdleMode() ...))
```

```
Object_Timelines GoalClass GoalInstance (  
  GoalTL([o] IdleGoal() [io] TakeImageGoal(satelliteo instrumento  
  Phenomenon4 thermographo * *) [6o] ...))
```

```
Object_Timelines PlanningClass PlanningInstance (  
  PlanningSV ([o] IdlePlanning() [io] Planning() ... ))
```

# Outline

- Introduction
- Preferences in Planning
- PDDL<sub>3</sub> syntax
- PDDL<sub>3</sub> examples
- XIDDL language
- **Conclusions**

# Conclusions

- Represent plan with soft constraints & goals
  - Best quality plan satisfy “as much as possible” the soft constraints & goals
- PDDL<sub>2</sub>.X is extended: PDDL<sub>3</sub>
- XIDDL based on NDDL (EUROPA planner)
  - Evolution of system along time
  - Basic structure TL