PDDL 2.1 – Numbers and Optimization

Dr. Adrián Domínguez Díaz

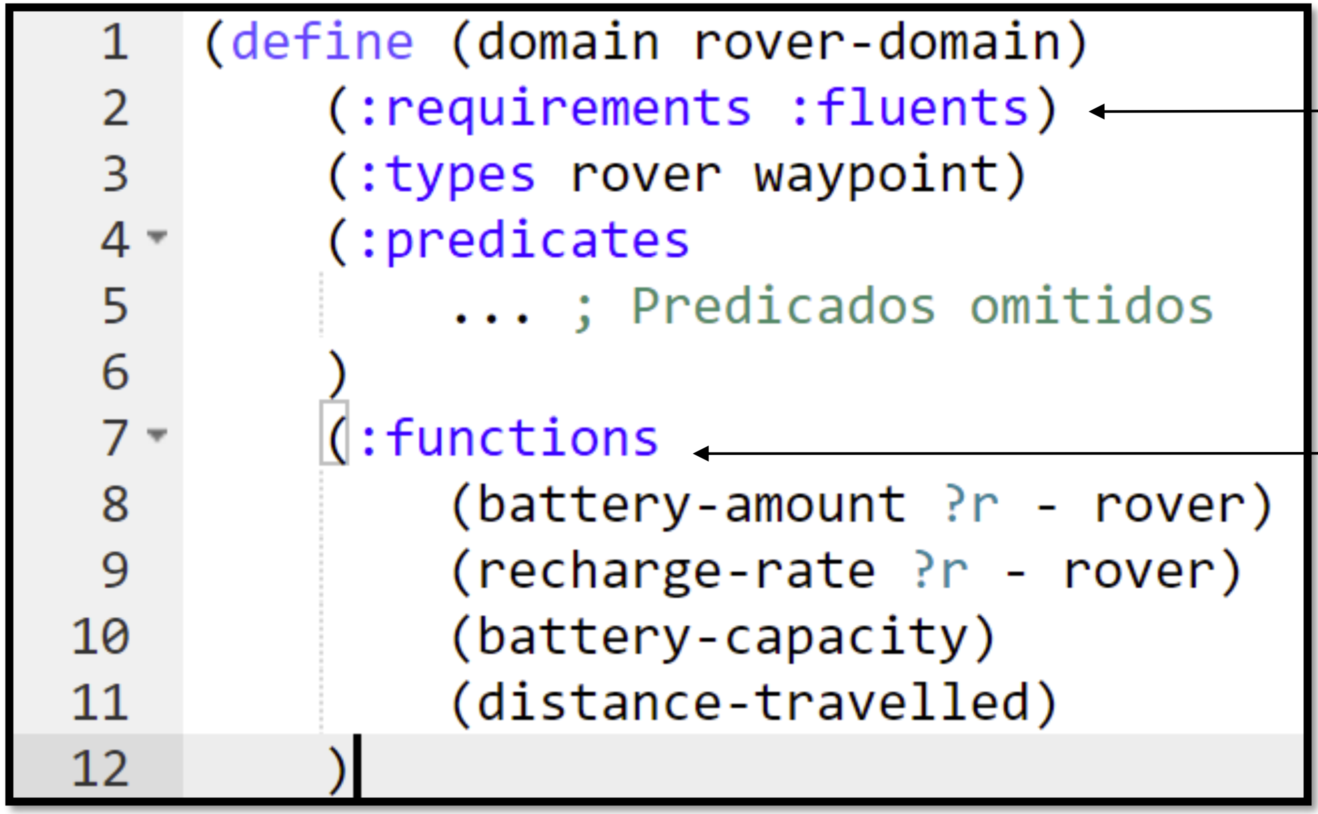
Planificación Automática

"Numeric fluents” and “metrics”

- PDDL 2.1 adds the ability to create "numeric fluents"
  - State variables with a numeric value assigned to them
  - They are declared with the tag (:functions...) similar to predicates
    - They must be declared between predicates and actions
  - They can then be used in actions
    - Comparison functions (>, <, =) can be used in preconditions
    - As an effect, its value can be assigned, increased or decreased

- Metrics that define the total cost of a plan are also added
  - A metric returns a fluent or an operation with fluents
  - Planners who support it will try to optimize that metric
  - Optimization can be defined as maximizing or minimizing the final value

Numeric fluents - Domain

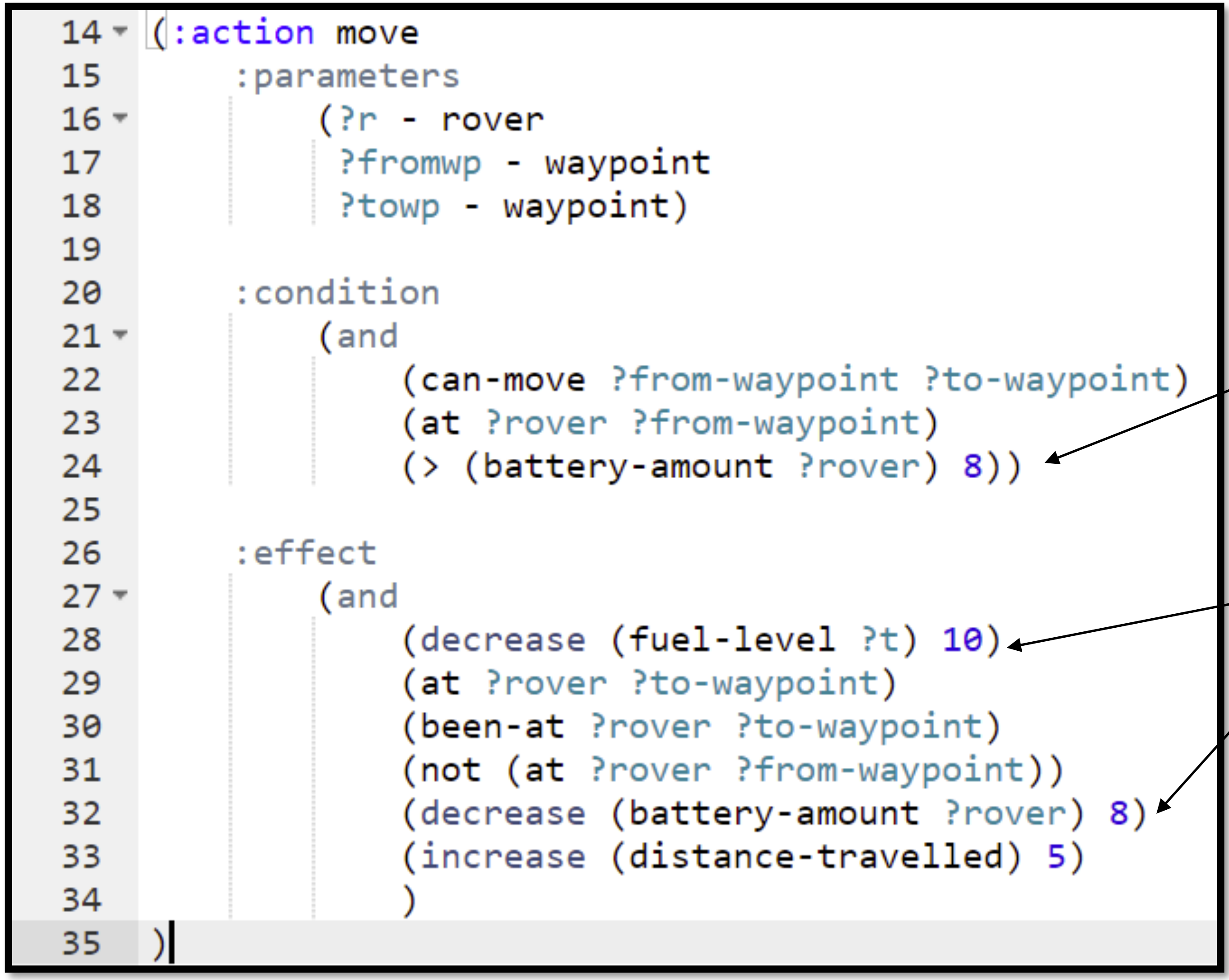


```
(define (domain rover-domain)
    (:requirements :fluents)
    (:types rover waypoint)
    (:predicates
        ... ; Predicados omitidos
    )
    (:functions
        (battery-amount ?r - rover)
        (recharge-rate ?r - rover)
        (battery-capacity)
        (distance-travelled)
    )
```

- Need to add requirement
- We'll use :numeric-fluents better

- Numeric status variables
  - Battery of a given robot
  - Reload rate of a given robot
  - Battery Capacity (General)
  - Distance Traveled (Overall)

Numeric fluents - Domain



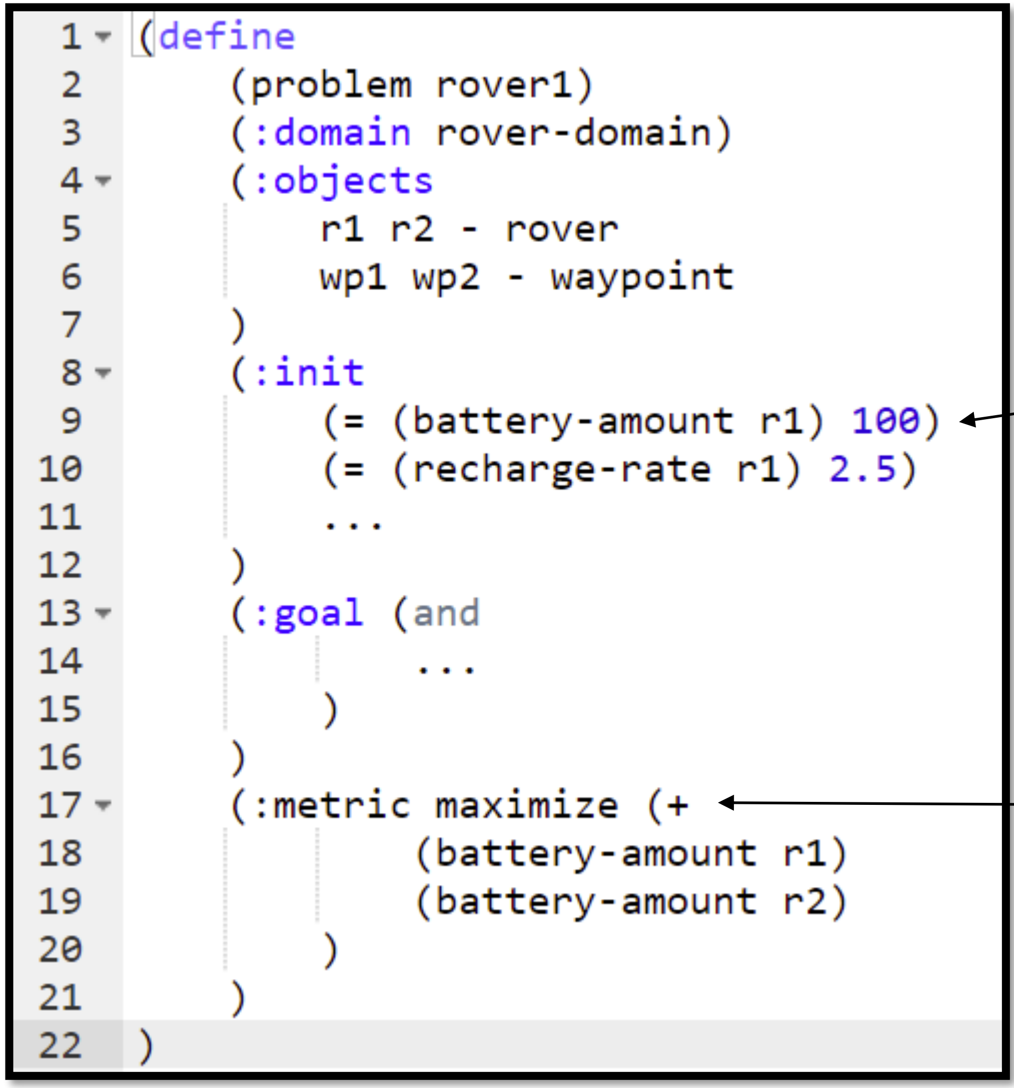
```
(:action move
    :parameters
        (?r - rover
         ?fromwp - waypoint
         ?towp - waypoint)

    :condition
        (and
            (can-move ?from-waypoint ?to-waypoint)
            (at ?rover ?from-waypoint)
            (> (battery-amount ?rover) 8))

    :effect
        (and
            (decrease (fuel-level ?t) 10)
            (at ?rover ?to-waypoint)
            (been-at ?rover ?to-waypoint)
            (not (at ?rover ?from-waypoint))
            (decrease (battery-amount ?rover) 8)
            (increase (distance-travelled) 5)
            )
)
```

- Can be used in action preconditions using comparison functions (<, >, <=, >=, =)

- They can be modified in the effects of the actions (`assign, increase, decrease`)

- They can be used in basic mathematical operations with prefixed notation. E.g.: `(increase (distance-travelled) (* 5 5))`

Numeric fluents - Problem



```
(define
    (problem rover1)
    (:domain rover-domain)
    (:objects
        r1 r2 - rover
        wp1 wp2 - waypoint
    )
    (:init
        (= (battery-amount r1) 100)
        (= (recharge-rate r1) 2.5)
        ...
    )
    (:goal (and
            ...
        )
    )
    (:metric maximize (+
            (battery-amount r1)
            (battery-amount r2)
        )
    )
)
```

- They should be assigned an initial value during initialization

- In case of initializing route costs, a value would be assigned to the *fluent* for each source -> destination pair (2 arguments)

- The ":metric" metrics define whether the planner should try to maximize or minimize a certain calculated value

- The value to be optimized can simply be a *fluent*, or a basic operation with *fluents*

Planners & Optimization

- Many classic planners do not support *numeric fluents*
  - Some give an error if these features are included
    - E.g.: Basic FF Planner
  - Others don't give errors, but ignore their use completely
    - For example, the default scheduler in http://editor.planning.domains

- Although they support *fluents*, *metric* support varies
  - Many are looking to get a plan quickly, not for it to be optimal
  - Some return a first plan and then look for ways to improve it
    - They are called *anytime* planners and are set up with a maximum time
    - E.g.: FF Anytime planner (PDDL4J) or OPTIC
  - Others just return an optimal plan, even if it takes longer to calculate
    - They are much slower than the rest, they must make sure that there is no better plan
    - E.g.: Fast Downward Planner, used as a basis in many other planners

Optimization with "action costs"

- action-costs: Limited version of “numeric-fluents” extension that is supported by more planners
  - *Fluents* cannot be used as a condition in actions or goals
  - In the effects of actions, only one *fluent* can be modified, which must be named *(total-cost)* and must not have parameters.
  - The *(total-cost) fluent* can only be increased, it cannot be decreased.
  - There can be no *fluents* initialized with negative values.
  - The only *metric* of the problem can be to minimize *(total-cost)*

- In laboratory practice we will use this extension.

Exercises – Dominio Airport

- There are people who are in one city and want to travel to another by plane.
- There is a certain distance between every two cities.
- Each aircraft has a fuel tank. When traveling, the plane consumes fuel proportional to the distance traveled.
- Before flying, you need to have enough fuel.
- You want everyone to be able to fly to their destination while consuming the minimum amount of fuel in total.