Intelligent Robotics Lab

Planificación y Sistemas Cognitivos

Bloque 1

Planificación Clásica

Universidad Rey Juan Carlos

Objetivo del Tema

1. Aprenderemos qué es Classic Planning y las aproximaciones típicas
2. Introduciremos PDDL (Panning Definition Domain language)
3. Aprenderemos a plantear problemas que resolveremos aplicando planning
4. Usaremos POPF, una planner de PDDL

¿Qué es Planning?

1. Planning es el **proceso de cálculo de un plan para conseguir uno o varios objetivos**
2. Un plan es una secuencia de acciones que conduce de un estado inicial a un estado que contiene los objetivos a conseguir

¿Cuáles son los elementos que encontramos en el Planning?

1. El estado de un problema lo conforma un conjunto de predicados

$$Perdido \wedge huerfano$$

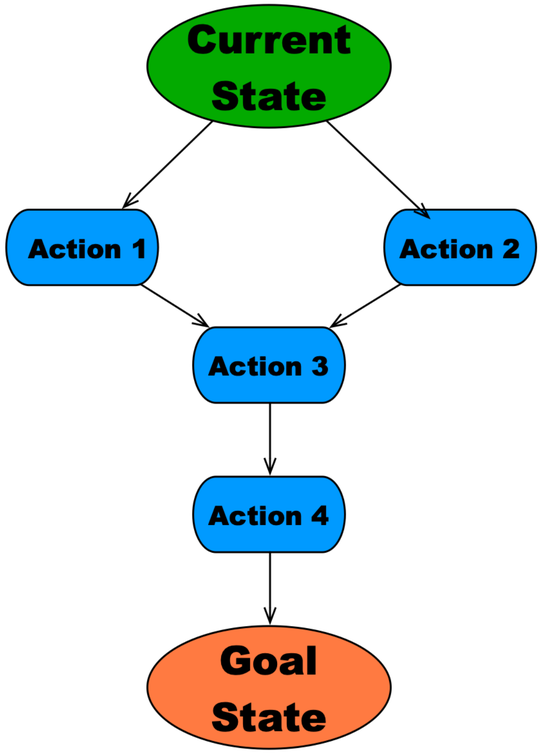
$$robotAt(r2d2, Tatooine) \wedge robotAt(c3po, Devaron)$$

2. No son válidos

$\neg rico$	es negativo
$\neg robotAt(bb8, Dagobah)$	es negativo
$areFighting(owner(bb8), DarthVader)$	No funciones
$robotAt(x, y)$	No grounded

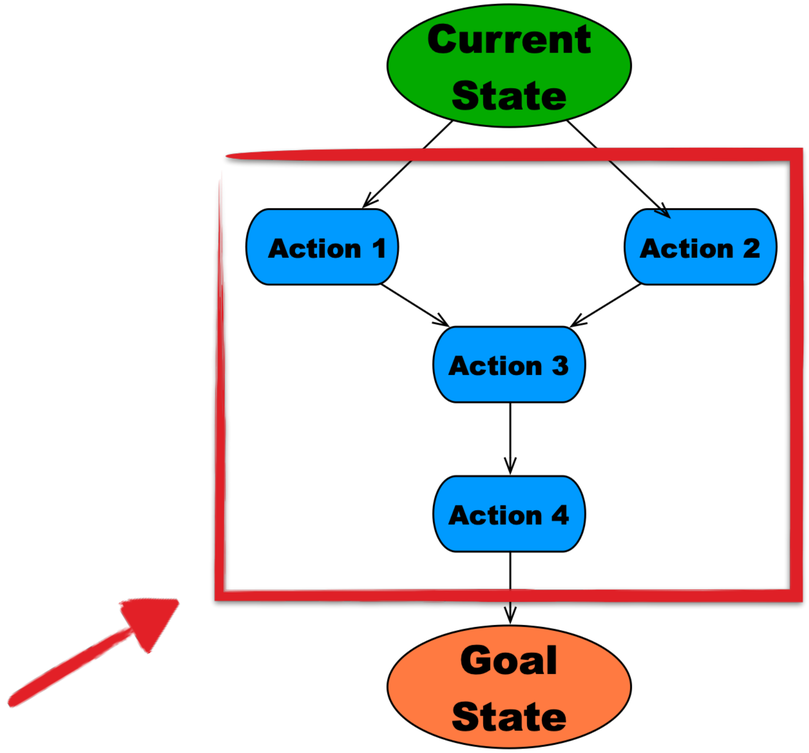
¿Qué es Planning?

1. Planning es el **proceso de cálculo de un plan para conseguir uno o varios objetivos**
2. Un plan es una secuencia de acciones que conduce de un estado inicial a un estado que contiene los objetivos a conseguir



¿Qué es Planning?

1. Planning es el **proceso de cálculo de un plan para conseguir uno o varios objetivos**
2. Un plan es una secuencia de acciones que conduce de un estado inicial a un estado que contiene los objetivos a conseguir



Esto se calcula automáticamente!!!

Más elementos en Planning

1. Una acción permite cambiar el estado del problema
2. Tiene una precondición
3. Tiene un efecto que cambia el estado del problema
4. Un acción es aplicable si su precondición se cumple

$$
\begin{aligned}
&Action(Travel(r, s, from, to),\\
&PRECOND: robotAt(r, from) \wedge spaceshipAt(s, from) \wedge robot(r) \wedge spaceship(s) \wedge\\
&planet(from) \wedge planet(to)\\
&EFFECT: \neg robotAt(r, from) \wedge robotAt(r, to))
\end{aligned}
$$

5. Add List (ADD) es la lista de predicados que añade una acción
6. Delete List (DEL) es la lista de predicados que añade una acción

$$(state, action) = (state - DEL(action)) \cup ADD(action)$$

$$Travel(r2d2, millenniumFalcon, Tatooine, Naboo)$$

$$
\begin{aligned}
&robotAt(r2d2, Tatooine) \wedge\\
&robot(r2d2) \wedge\\
&spaceship(millenniumFalcon) \wedge\\
&planet(Tatooine) \wedge\\
&planet(Naboo)
\end{aligned}
\longrightarrow
\begin{aligned}
&\cancel{robotAt(r2d2, Tatooine) \wedge}\\
&robot(r2d2) \wedge\\
&spaceship(millenniumFalcon) \wedge\\
&planet(Tatooine) \wedge\\
&planet(Naboo) \wedge\\
&robotAt(r2d2, Naboo)
\end{aligned}
$$

Intelligent Robotics Lab

Ejemplo: Viajes estelares

$$
\begin{aligned}
&Init(\\
&robotAt(r2d2, Naboo) \wedge robotAt(c3p0, Tatooine) \wedge spaceshipAt(milleniunFalcon, Alderaan) \wedge\\
&spaceshipAt(Xwing, Kessel) \wedge robot(r2d2) \wedge robot(c3p0) \wedge spaceship(milleniunFalcon) \wedge\\
&spaceship(Xwing) \wedge planet(Alderaan) \wedge planet(Kessel))
\end{aligned}
$$

$$
\begin{aligned}
&Goal(\\
&robotAt(r2d2, Tatooine) \wedge robotAt(c3p0, Naboo))
\end{aligned}
$$

$$
\begin{aligned}
&Action(Load(r, s, p),\\
&PRECOND\ :\ robotAt(r, p) \wedge spaceshipAt(s, p) \wedge robot(r) \wedge spaceship(s) \wedge\\
&planet(p)\\
&EFFECT : \neg robotAt(r, p) \wedge robotIn(r, s)
\end{aligned}
$$

$$
\begin{aligned}
&Action(Unload(r, s, p),\\
&PRECOND\ :\ robotIn(r, s) \wedge spaceshipAt(s, p) \wedge robot(r) \wedge spaceship(s) \wedge\\
&planet(p)\\
&EFFECT : \neg robotIn(r, s) \wedge robotAt(r, p)
\end{aligned}
$$

$$
\begin{aligned}
&Action(Fly(s, from, to),\\
&PRECOND : spaceshipAt(s, from) \wedge spaceship(s) \wedge planet(from) \wedge planet(to)\\
&EFFECT : \neg spaceshipAt(s, from) \wedge spaceshipAt(s, to)
\end{aligned}
$$

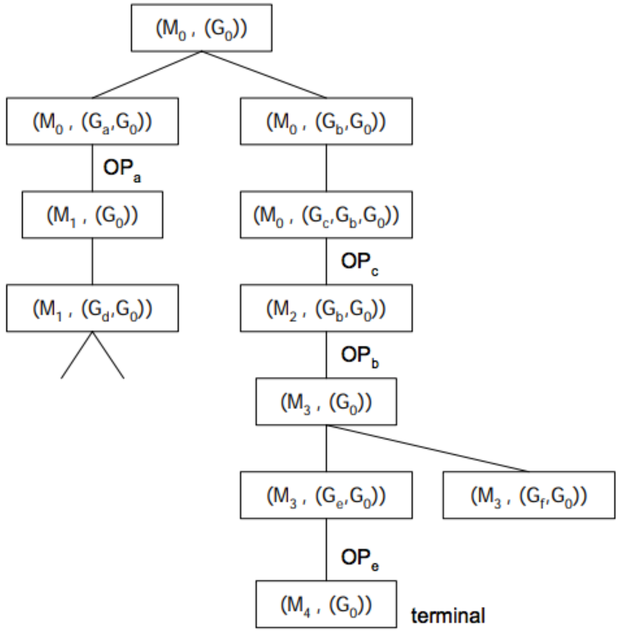
Ejercicio: Viajes estelares

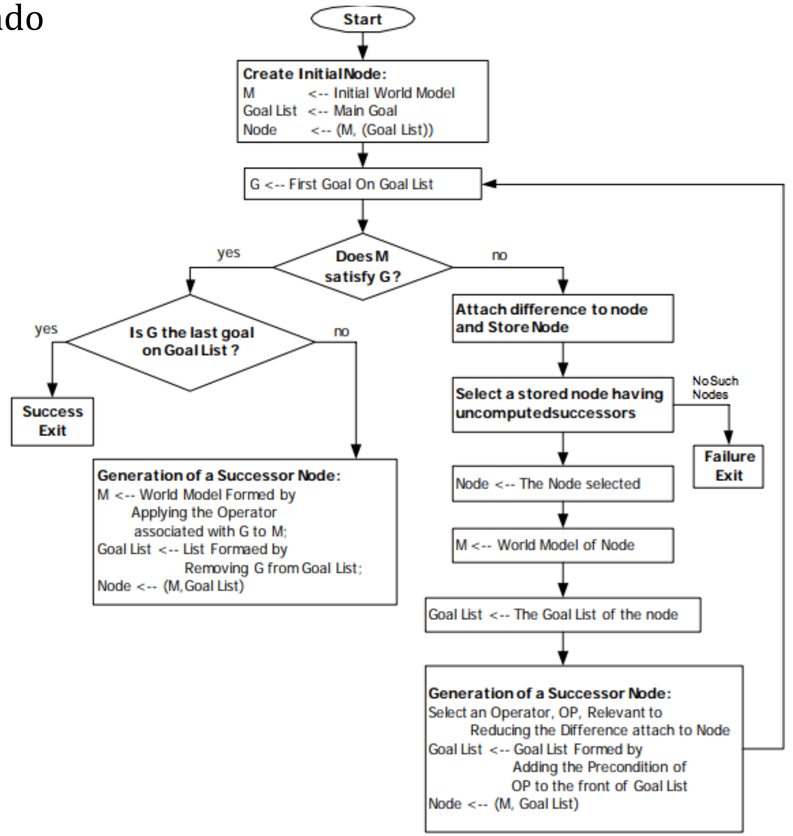
1. En una hoja de papel pon el estado inicial
2. Calcula la secuencia de acciones para obtener el goal
3. Por cada acción, tacha los predicados que se eliminan, y escribe los que se añaden
4. Encuentra el error, y qué efecto produce

10 minutos!!!

STanford Research Institute Problem Solver
STRIPS

- 1971. Imperial College London
- Primer lenguaje de planning ampliamente usado
- Lo usó Shakey en 1984 (SRI)





http://ai.stanford.edu/~nilsson/OnlinePubs-Nils/shakey-the-robot.pdf

http://www.cs.cmu.edu/~mmv/planning/readings/strips.pdf

Intelligent Robotics Lab

Planning Domain Definition Language
PDDL

- PDDL1.2 fue el lenguaje oficial del primer IPC (International Planning Competition) en 1998
- Inspirado en STRIPS
- Un modelo puede aplicarse a muchos problemas distintos
- La versión 2.1 es la más popular, ya que incluye durative actions.
- POPF

http://www.cs.cmu.edu/~mmv/planning/readings/98aips-PDDL.pdf

Planning Domain Definition Language
PDDL

- Establecer las reglas (Domain)
- Presentar una situación y un goal (Problem)
- Usar un plan solver
- Conseguir un plan

```
(define (domain simple)
(:types robot room)
(:predicates
  (robot_at ?r - robot ?ro - room)
  (connected ?ro1 ?ro2 - room))
(:durative-action move
  :parameters (?r - robot ?r1 ?r2 - room)
  :duration ( = ?duration 5)
  :condition (and
    (at start(connected ?r1 ?r2))
    (at start(robot_at ?r ?r1)))
  :effect (and
    (at start(not(robot_at ?r ?r1)))
    (at end(robot_at ?r ?r2))))
)
```

Domain.pddl

Planning Domain Definition Language PDDL

- Establecer las reglas (Domain)
- Presentar una situación y un goal (Problem)
- Usar un plan solver
- Conseguir un plan

```
(define (domain simple)
(:types robot room)
(:predicates
  (robot_at ?r - robot ?ro - room)
  (connected ?ro1 ?ro2 - room))
(:durative-action move
  :parameters (?r - robot ?r1 ?r2 - room)
  :duration ( = ?duration 5)
  :condition (and
    (at start(connected ?r1 ?r2))
    (at start(robot_at ?r ?r1)))
  :effect (and
    (at start(not(robot_at ?r ?r1)))
    (at end(robot_at ?r ?r2))))
)
```

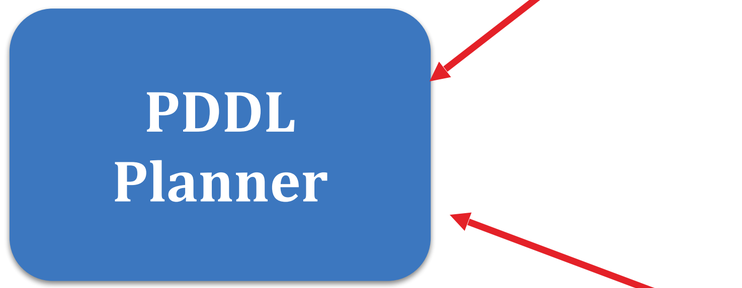
Domain.pddl

```
(define (problem problem_1)
(:domain simple)
(:objects
  r2d2 - robot
  bedroom living kitchen - room
)
(:init
  (robot_at r2d2 bedroom)
  (connected living bedroom)
  (connected bedroom living)
  (connected living kitchen)
  (connected kitchen living))
(:goal (and(robot_at r2d2 kitchen)))
)
```

Problem1.pddl

Planning Domain Definition Language PDDL

- Establecer las reglas (Domain)
- Presentar una situación y un goal (Problem)
- Usar un plan solver
- Conseguir un plan




```
(define (domain simple)
(:types robot room)
(:predicates
  (robot_at ?r - robot ?ro - room)
  (connected ?ro1 ?ro2 - room))
(:durative-action move
  :parameters (?r - robot ?r1 ?r2 - room)
  :duration ( = ?duration 5)
  :condition (and
    (at start(connected ?r1 ?r2))
    (at start(robot_at ?r ?r1)))
  :effect (and
    (at start(not(robot_at ?r ?r1)))
    (at end(robot_at ?r ?r2))))
)
```

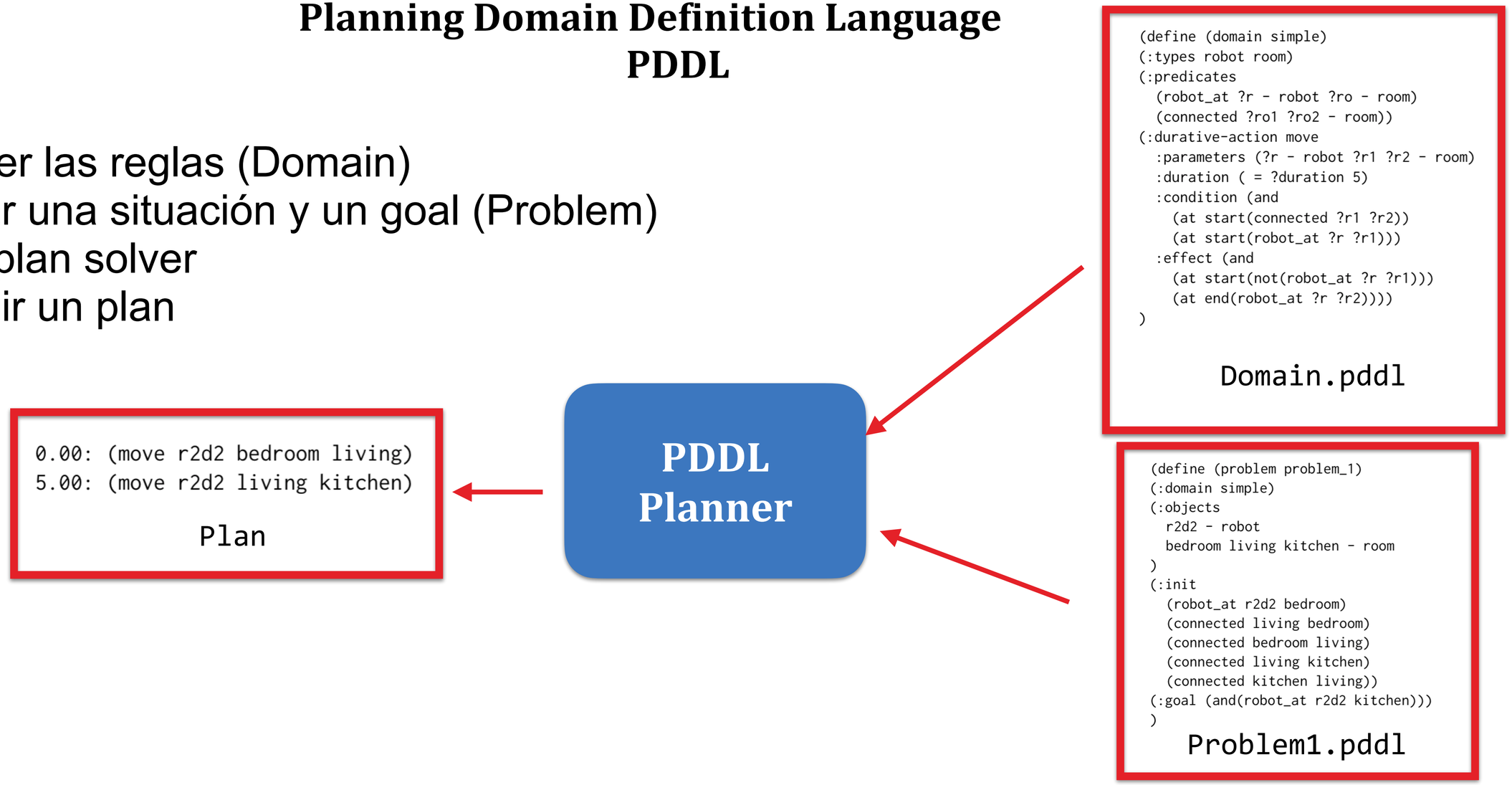
Domain.pddl

```
(define (problem problem_1)
(:domain simple)
(:objects
  r2d2 - robot
  bedroom living kitchen - room
)
(:init
  (robot_at r2d2 bedroom)
  (connected living bedroom)
  (connected bedroom living)
  (connected living kitchen)
  (connected kitchen living))
(:goal (and(robot_at r2d2 kitchen)))
)
```

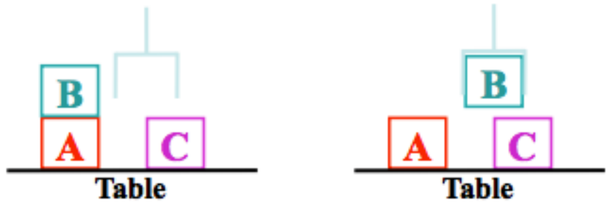
Problem1.pddl

Planning Domain Definition Language PDDL

- Establecer las reglas (Domain)
- Presentar una situación y un goal (Problem)
- Usar un plan solver
- Conseguir un plan



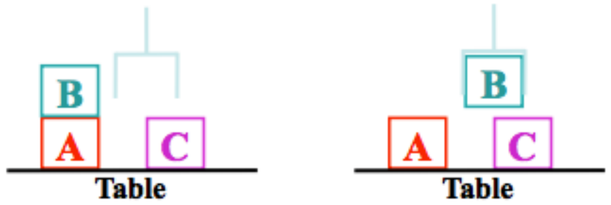
El mundo de los bloques




```
(define (domain blocks)
(:requirements :strips :equality)
(:predicates
  (on ?x ?y)
  (clear ?x)
  (table ?t)
  (block ?b)
)

(:constants Table)
```

https://github.com/fmrico/planning_cognitve_systems_course/tree/main/pddl

El mundo de los bloques

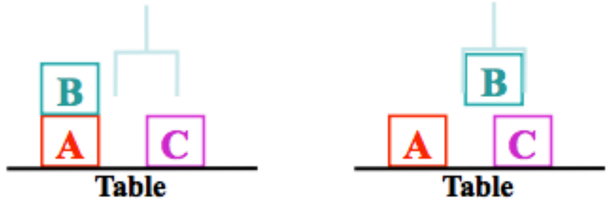



```
(:action move_to_table
  :parameters (?b ?x)
  :precondition
    (and
      (block ?b)
      (block ?x)
      (on ?b ?x)
      (clear ?b)
      (not (= ?b ?x))
    )
  :effect
    (and
      (on ?b Table)
      (clear ?x)
      (not (on ?b ?x))
    )
)
)
```

```
(:action move
  :parameters (?b ?x ?y)
  :precondition
    (and
      (block ?b)
      (block ?y)
      (clear ?b)
      (clear ?y)
      (on ?b ?x)
      (not (= ?b ?x))
      (not (= ?b ?y))
      (not (= ?x ?y))
    )
  :effect
    (and
      (on ?b ?y)
      (clear ?x)
      (not (on ?b ?x))
      (not (clear ?y))
    )
)
```

https://github.com/fmrico/planning_cognitve_systems_course/tree/main/pddl

El mundo de los bloques



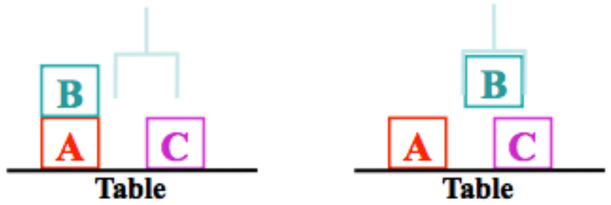

```
(define (problem blocks1)
(:domain blocks)
(:objects
  a b c
)
(:init
  (block a)
  (block b)
  (block c)
  (clear b)
  (clear c)
  (on b Table)
  (on a Table)
  (on c a)
)
```

```
(:goal (and
  (on b c)(on a b))
)

)
```

- Ejercicio 1: Prueba a resolverlo con popf
- Ejercicio 2: Añade otro bloque y prueba una configuración inicial más compleja

https://github.com/fmrico/planning_cognitve_systems_course/tree/main/pddl

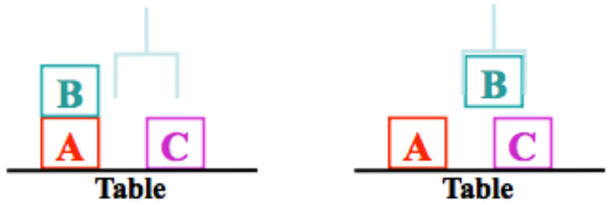
El mundo de los bloques con tipos




```
(define (domain blocks)
(:requirements :strips :equality :typing)

(:types
  table block - object
)

(:predicates
  (on ?x ?y - object)
  (clear ?x - object)
)

(:constants Table - table)
```

https://github.com/fmrico/planning_cognitve_systems_course/tree/main/pddl

El mundo de los bloques con tipos

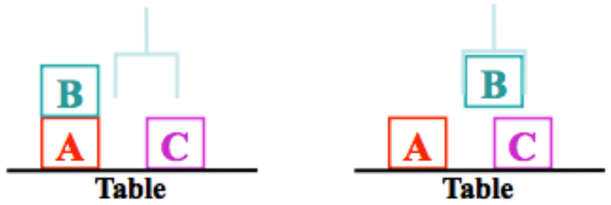



```
(:action move_to_table
  :parameters (?b ?x - block)
  :precondition
    (and
     (on ?b ?x)
      (clear ?b)
      (not (= ?b ?x))
    )
  :effect
    (and
      (on ?b Table)
      (clear ?x)
      (not (on ?b ?x))
    )
)
)
```

```
(:action move
  :parameters (?b - block ?x - object ?y - block)
  :precondition
    (and
      (clear ?b)
      (clear ?y)
      (not (= ?b ?y))
   )
  :effect
    (and
      (on ?b ?y)
      (clear ?x)
      (not (on ?b ?x))
      (not (clear ?y))
    )
)
```

https://github.com/fmrico/planning_cognitve_systems_course/tree/main/pddl

El mundo de los bloques con tipos




```
(define (problem blocks1)
(:domain blocks)
(:objects
  a b c d - block
)
(:init
  (clear b)
  (clear d)
  (on c Table)
  (on a Table)
  (on b a)
  (on d c)
)
```

```
(:goal (and
  (on b a)
  (on c b)
  (on d c)
  (on a Table)
)
)
```

https://github.com/fmrico/planning_cognitve_systems_course/tree/main/pddl

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

`(domain <name>)`

- Un dominio siempre empieza por `(define` y lo siguiente es la especificación del nombre del dominio del dominio.
- La mayor parte de los planificadores usan el nombre del fichero en lugar de este nombre

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

`(domain <name>)`

- Un dominio siempre empieza por `(define` y lo siguiente es la especificación del nombre del dominio del dominio.
- La mayor parte de los planificadores usan el nombre del fichero en lugar de este nombre

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- No está muy soportado
- “Hereda” de otro dominio “padre” la mayor parte de sus componentes

```
(:extends <domain_name>)
```

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Simular a `#include` o `import`
- “Hereda” de otro dominio “padre” la mayor parte de sus componentes

```
(:requirements <requirement_name>)
```

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Simular a `#include` o `import`
- "Hereda" de otro dominio "padre" la mayor parte de sus componentes

```
(:requirements <requirement_name>)
```

- `:strips`
- `:typing`
- `:disjunctive-preconditions`
- `:equality`
- `:existential-preconditions`
- `:universal-preconditions`
- `:quantified-preconditions`
- `:conditional-effects`
- `:action-expansions`
- `:foreach-expansions`
- `:dag-expansions`
- `:domain-axioms`
- `:subgoal-through-axioms`
- `:safety-constraints`
- `:expression-evaluation`
- `:fluents`
- `:open-world`
- `:true-negation`
- `:adl`
- `:ucpop`
- `:numeric-fluents`
- `:durative-actions`
- `:continuous-effects`
- `:negative-preconditions`

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite añadir o borrar efectos

```
:effect (walls-built ?s)
:effect (not (walls-built ?s))
```

- **:strips**
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite usar tipos

```
(:types
    site material - object
    bricks cables windows - material
)
```

- :strips
- **:typing**
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite usar or en las precondiciones

```
(or
    (walls-built ?s)
    (windows-fitted ?s)
)
```

- :strips
- :typing
- **:disjunctive-preconditions**
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- :strips
- :typing
- :disjunctive-preconditions
- **:equality**
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

- Permite comparar si dos objetivos son el mismo

```
(not (= ?s1 ?s2))
```

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite usar `exist` en goals y precondiciones

```
(exists (?c - crane)
    (crane-is-free ?c)
)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- **:existential-preconditions**
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite usar `forall` es goals y precondiciones

```
(forall (?c - crane)
    (crane-is-free ?c)
)
```

- `:strips`
- `:typing`
- `:disjunctive-preconditions`
- `:equality`
- `:existential-preconditions`
- **`:universal-preconditions`**
- `:quantified-preconditions`
- `:conditional-effects`
- `:action-expansions`
- `:foreach-expansions`
- `:dag-expansions`
- `:domain-axioms`
- `:subgoal-through-axioms`
- `:safety-constraints`
- `:expression-evaluation`
- `:fluents`
- `:open-world`
- `:true-negation`
- `:adl`
- `:ucpop`
- `:numeric-fluents`
- `:durative-actions`
- `:continuous-effects`
- `:negative-preconditions`

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Es equivalente a

```
(:requirements :existential-preconditions :universal-preconditions)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- **:quantified-preconditions**
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Intelligent Robotics Lab

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite definir la misma acción con diferentes tipos

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- **:action-expansions**
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- **:foreach-expansions**
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

- Permite usar `foreach` en expansiones de acciones
- Equivalente a

```
(:requirements :action-expansions :foreach-expansions)
```

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- **:domain-axioms**
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

- Permite usar axiomas

```
(:derived (clear ?x)
   (and (not (holding ?x))
   (forall (?y) (not (on ?y ?x))))))
```

https://planning.wiki/

Intelligent Robotics Lab

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite usar axiomas como subgoals

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- **:subgoal-through-axioms**
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite definir predicados que deben ser válidos al final de la ejecución de un plan

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- **:safety-constraints**
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- **:expression-evaluation**
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

- Permite usar `eval` en axiomas
- Si dos predicados son equivalentes, devuelve true

```
(eval (im-not-true ?a) (im-true ?b))
```

https://planning.wiki/

Intelligent Robotics Lab

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite usar `(fluents t)` en axiomas
- Cambia en PDDL 2.1, y no está claro el uso

- `:strips`
- `:typing`
- `:disjunctive-preconditions`
- `:equality`
- `:existential-preconditions`
- `:universal-preconditions`
- `:quantified-preconditions`
- `:conditional-effects`
- `:action-expansions`
- `:foreach-expansions`
- `:dag-expansions`
- `:domain-axioms`
- `:subgoal-through-axioms`
- `:safety-constraints`
- `:expression-evaluation`
- **`:fluents`**
- `:open-world`
- `:true-negation`
- `:adl`
- `:ucpop`
- `:numeric-fluents`
- `:durative-actions`
- `:continuous-effects`
- `:negative-preconditions`

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- En planning, el predicado que no existe se considera falso (*closed-world assumption*)
- Esto permite que no sea así

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- **:open-world**
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- No considera negación como fallo

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- **:true-negation**
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Implica otros requirements

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- **:adl**
- :ucpop
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Implica otros requirements

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- **:ucpop**
- :numeric-fluents
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permiten usar funciones, que representan valores numéricos

```
(:functions
    (battery-amount ?r - rover)
)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- **:numeric-fluents**
- :durative-actions
- :continuous-effects
- :negative-preconditions

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- **:durative-actions**
- :continuous-effects
- :negative-preconditions

- Permiten usar `durative-action`

```
(:durative-action move
    :parameters (<arguments>)
    :duration (= ?duration 5)
    :condition (logical_expression)
    :effect (logical_expression)
)
```

https://planning.wiki/

Intelligent Robotics Lab

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- **:durative-inequalities**
- :continuous-effects
- :negative-preconditions

- Permiten usar desigualdades para expresar duración

```
(= ?duration
 (/ (- 80 (battery-amount ?rover))
    (recharge-rate ?rover)))
```

https://planning.wiki/

Intelligent Robotics Lab

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- :strips
- :typing
- :disjunctive-preconditions
- :equality
- :existential-preconditions
- :universal-preconditions
- :quantified-preconditions
- :conditional-effects
- :action-expansions
- :foreach-expansions
- :dag-expansions
- :domain-axioms
- :subgoal-through-axioms
- :safety-constraints
- :expression-evaluation
- :fluents
- :open-world
- :true-negation
- :adl
- :ucpop
- :numeric-fluents
- :durative-actions
- :durative-inequalities
- **:continuous-effects**
- :negative-preconditions

- Permite el uso de efectos continuos sobre números dentro de acciones durativas.

```
:effect
            (at end
                (increase (battery-amount ?rover)
                    (* ?duration (recharge-rate ?rover))))
        )
```

https://planning.wiki/

Intelligent Robotics Lab

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Permite usar `not` en precondiciones

- `:strips`
- `:typing`
- `:disjunctive-preconditions`
- `:equality`
- `:existential-preconditions`
- `:universal-preconditions`
- `:quantified-preconditions`
- `:conditional-effects`
- `:action-expansions`
- `:foreach-expansions`
- `:dag-expansions`
- `:domain-axioms`
- `:subgoal-through-axioms`
- `:safety-constraints`
- `:expression-evaluation`
- `:fluents`
- `:open-world`
- `:true-negation`
- `:adl`
- `:ucpop`
- `:numeric-fluents`
- `:durative-actions`
- `:durative-inequalities`
- `:continuous-effects`
- **`:negative-preconditions`**

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Definición de tipos

```
(:types
    <type_name_1> ... <type_name_n> - object
    <sub_name_1> ... <sub_name_n> - <type_name_1>
)
```

Planning Domain Definition Language
PDDL

```
(define
    (domain construction)
    (:extends building)
    (:requirements :strips :typing)
    (:types
        site material - object
        bricks cables windows - material
    )
    (:constants mainsite - site)
```

- Definición constantes que se pueden usar en el dominio

Planning Domain Definition Language
PDDL

```
(:predicates
        (walls-built ?s - site)
        (windows-fitted ?s - site)
        (foundations-set ?s - site)
        (cables-installed ?s - site)
        (site-built ?s - site)
        (on-site ?m - material ?s - site)
        (material-used ?m - material)
    )

    (:timeless (foundations-set mainsite))
```

- Definición de predicados

```
(:predicates
    (<predicate_name> <argument_1> ... <argument_n>)
)
```

Planning Domain Definition Language
PDDL

```
(:predicates
        (walls-built ?s - site)
        (windows-fitted ?s - site)
        (foundations-set ?s - site)
        (cables-installed ?s - site)
        (site-built ?s - site)
        (on-site ?m - material ?s - site)
        (material-used ?m - material)
    )

    (:timeless (foundations-set mainsite))
```

- Un predicado que siempre es verdadero

Planning Domain Definition Language
PDDL

```
(:action BUILD-WALL
        :parameters (?s - site ?b - bricks)
        :precondition (and
            (on-site ?b ?s)
            (foundations-set ?s)
            (not (walls-built ?s))
            (not (material-used ?b))
        )
        :effect (and
            (walls-built ?s)
            (material-used ?b)
        )
        ; :expansion ;deprecated
    )

    (:axiom
        :vars (?s - site)
        :context (and
            (walls-built ?s)
            (windows-fitted ?s)
            (cables-installed ?s)
        )
        :implies (site-built ?s)
    )

    ;Actions omitted for brevity
)
```

- Una acción cons sus componentes

https://planning.wiki/

Planning Domain Definition Language
PDDL

```
(:action BUILD-WALL
        :parameters (?s - site ?b - bricks)
        :precondition (and
            (on-site ?b ?s)
            (foundations-set ?s)
            (not (walls-built ?s))
            (not (material-used ?b))
        )
        :effect (and
            (walls-built ?s)
            (material-used ?b)
        )
        ; :expansion ;deprecated
    )

    (:axiom
        :vars (?s - site)
        :context (and
            (walls-built ?s)
            (windows-fitted ?s)
            (cables-installed ?s)
        )
        :implies (site-built ?s)
    )

    ;Actions omitted for brevity
)
```

- Un axioma es un predicado que se deriva de una condición

https://planning.wiki/

El dominio del brazo

```
(define (domain gripper-strips)
(:predicates
  (room ?r)
  (ball ?b)
  (gripper ?g)
  (at-robby ?r)
  (at ?b ?r)
  (free ?g)
  (carry ?o ?g)
)
```

```
 (:action move
   :parameters (?from ?to)
   :precondition
     (and
       (room ?from)
       (room ?to)
       (at-robby ?from)
     )
   :effect
     (and
       (at-robby ?to)
       (not (at-robby ?from)))
     )
```

```
(:action pick
  :parameters (?obj ?room ?gripper)
  :precondition
    (and
      (ball ?obj)
      (room ?room)
      (gripper ?gripper)
      (at ?obj ?room)
      (at-robby ?room)
      (free ?gripper)
    )
:effect
  (and
    (carry ?obj ?gripper)
    (not (at ?obj ?room))
    (not (free ?gripper)))
  )
```

```
(:action drop
:parameters (?obj ?room ?gripper)
:precondition
  (and
    (ball ?obj)
    (room ?room)
    (gripper ?gripper)
    (carry ?obj ?gripper)
    (at-robby ?room)
  )
:effect
  (and
    (at ?obj ?room)
    (free ?gripper)
    (not (carry ?obj ?gripper))
  )
)
)
```

https://github.com/fmrico/planning_cognitve_systems_course/blob/main/pddl/gripper_domain.pddl