DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

PDDL

Fernando Berzal, berzal@acm.org

Lenguajes para planificación

Lenguajes para la descripción de problemas de planificación:

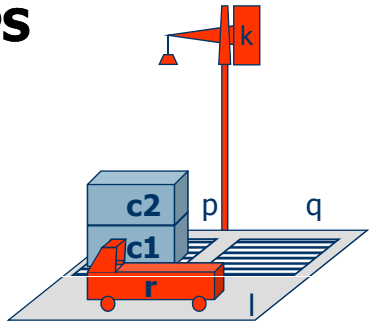
- **STRIPS** [Stanford Research Institute Problem Solver]
  Richard Fikes & Nils Nilsson, SRI International, 1971

- **ADL** [Action Description Language]
  Edwin Pednault, IBM Research, 1987

- **PDDL** [Planning Domain Definition Language]
  Drew McDermott, Yale University, 1998
  International Planning Competition
  @ ICAPS [International Conference on Automated Planning and Scheduling]
  http://ipc.icaps-conference.org/

STRIPS

Representación de estados en STRIPS



Estado inicial = {
  at(p,l), at(q,l), at(r,l), at(k,l),
  in(c1,p), in(c2,p),
  top(c2,p),
  on(c2,c1), on(c1,p),
  empty(k),
  unloaded(r)
}

Representación de objetivos en STRIPS

Objetivo = { on(c1,r) }

STRIPS

Representación de acciones en STRIPS

Acción
Nombre de la acción

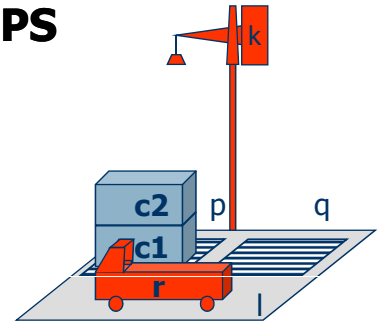
Precondiciones
Proposiciones que deben cumplirse para poder aplicar la acción.

Efectos
Consecuencias de la aplicación de la acción.

- "Add list" (proposiciones que pasan a ser ciertas)
- "Delete list" (proposiciones que pasan a ser falsas)

STRIPS

Representación de acciones en STRIPS



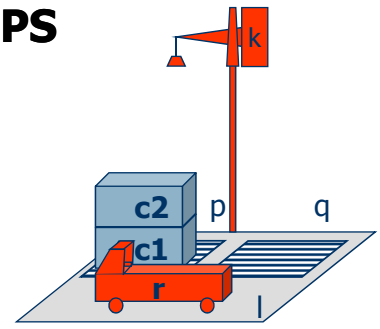
move(r,l,m)
- pre: adjacent(l,m), at(r,l)
- add: at(r,m)
- del: at(r,l)

load(k,l,c,r)
- pre: at(k,l), holding(k,c), at(r,l), unloaded(r)
- add: empty(k), loaded(r,c)
- del: holding(k,c), unloaded(r)

put(k,l,c,d,p)
- pre: at(k,l), at(p,l), holding(k,c), top(d,p)
- add: empty(k), in(c,p), top(c,p), on(c,d)
- del: holding(k,c), top(d,p)

STRIPS

Representación de acciones en STRIPS



move(r,l,m)
- pre: adjacent(l,m), at(r,l)
- eff: at(r,m), ¬at(r,l)

load(k,l,c,r)
- pre: at(k,l), holding(k,c), at(r,l), unloaded(r)
- eff: empty(k), loaded(r,c), ¬holding(k,c), ¬unloaded(r)

put(k,l,c,d,p)
- pre: at(k,l), at(p,l), holding(k,c), top(d,p)
- add: empty(k), in(c,p), top(c,p), on(c,d), ¬holding(k,c), ¬top(d,p)

STRIPS

- Sólo se especifica aquello que cambia.
- **Hipótesis de mundo cerrado**
  (lo que no se menciona es falso).
- Sólo literales positivos en la descripción de estados.
- Sólo conjunciones de literales simples en el objetivo.
- P∧¬Q como efecto añade P y elimina Q
- Las variables no pueden tener tipos.

ADL

- Sólo se especifica aquello que cambia.
- **Hipótesis de mundo abierto**
  (lo que no se menciona no se conoce).
- Literales positivos y negativos en los estados.
- Conjunciones y disjunciones en el objetivo.
- P∧¬Q como efecto añade {P, ¬Q} y elimina {¬P, Q}
- Las variables sí pueden tener tipos.

PDDL

Planning Domain Definition Language

http://cs-www.cs.yale.edu/homes/dvm/

- Estándar para la representación de tareas de planificación "clásica" utilizado desde 1998 en AIPS [International Conference on AI Planning & Scheduling], ahora ICAPS [International Conference on Automated Planning and Scheduling].
- Inspirado en UCPOP, un planificador de orden parcial desarrollado en la Universidad de Washington e implementado inicialmente en Common Lisp, utiliza la sintaxis del lenguaje de programación LISP.

PDDL

Componentes PDDL

- Objetos (entidades de interés).
- Predicados (propiedades de los objetos, true/false).
- Estado inicial
- Especificación de objetivos
- Acciones/operadores (formas de cambiar el estado).

PDDL

Especificación PDDL

Separada en dos ficheros:

- Fichero del dominio
  (predicados y acciones)
- Fichero del problema
  (objetos, estado inicial y especificación de objetivos).

PDDL

Fichero de especificación de dominios PDDL

(define (domain <domain name>**)**
  <PDDL code for predicates>
  <PDDL code for first action>
  [...]
  <PDDL code for last action>
)

PDDL

Fichero de especificación de dominios PDDL

Especificación de predicados (propiedades)

```
(:predicates     (room ?x)
                 (robot-at ?x))
```

- **room(x)** cierto si y sólo si x es una habitación.
- **robot-at(x)** cierto si y sólo si x es una habitación y, además, el robot está en x.

PDDL

Fichero de especificación de dominios PDDL

Especificación de acciones (precondiciones y efectos)

```
(:action move :parameters (?x ?y)
   :precondition   (and   (room ?x) (room ?y)
                          (robot-at ?x))
   :effect         (and   (robot-at ?y)
                          (not (robot-at ?x))))
```

PDDL

Fichero de especificación de problemas PDDL

```
(define (problem <problem name>)
   (:domain <domain name>)
   <PDDL code for objects>
   <PDDL code for initial state>
   <PDDL code for goal specification>
)
```


PDDL

Fichero de especificación de problemas PDDL

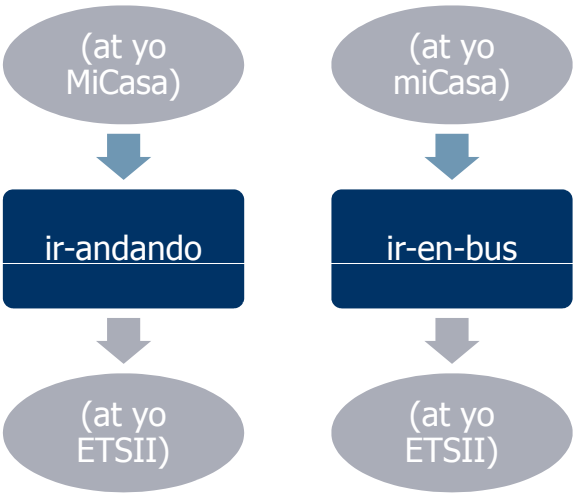
Estado inicial

```
(:init      (room cocina)
            (room comedor)
            (room dormitorio)
            (robot-at comedor))
```

Objetivo

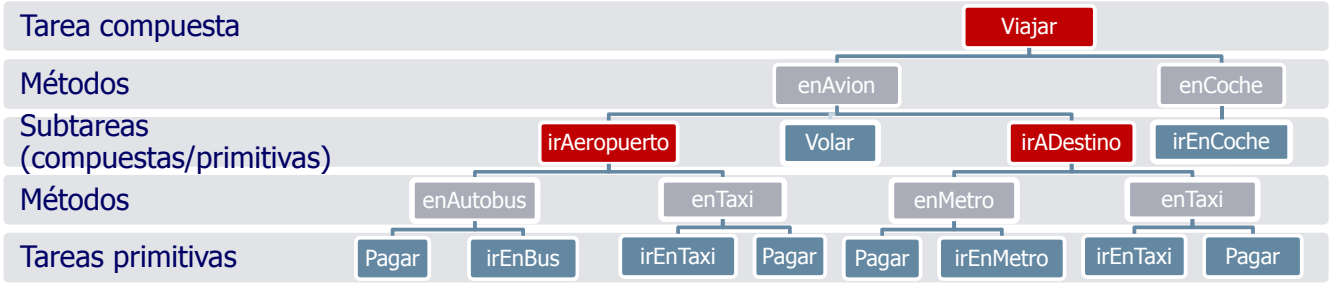
```
(: goal     (robot-at cocina))
```

HTN [Hierarchical Task Network]



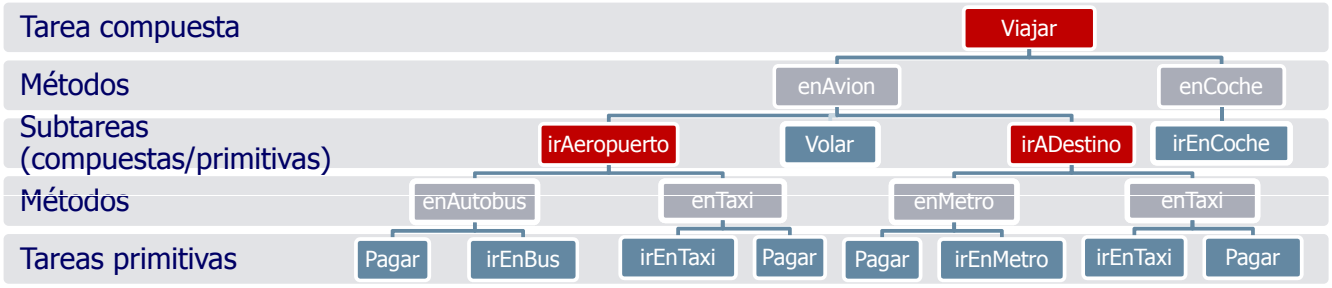
- La **planificación jerárquica** facilita la representación de medios alternativos para la realización de tareas a distintos niveles de abstracción [*abstraction gap*].

HTN [Hierarchical Task Network]



En vez de buscar un plan que consista en una secuencia de acciones individuales, tratamos con un menor número de tareas que pueden corresponder a múltiples acciones cada una de ellas [*refinement planning*]

HTN [Hierarchical Task Network]



- Dominio de planificación basado en la representación de tareas [tasks] a distintos niveles de abstracción.
- Distinción entre tareas compuestas y tareas primitivas (acciones).
- Especificación de métodos alternativos para la realización de tareas.

HTN-PDDL

- Estándar para representar problemas clásicos de planificación

PDDL

- Extensión desarrollada por el Grupo de Sistemas Inteligentes del Departamento de Ciencias de la Computación e I.A. (DECSAI) de la Universidad de Granada para planificación jerárquica.

HTN-PDDL

- IActivePlanner: Planificador desarrollado por el grupo ISG y transferido a la spin-off IActive Intelligent Technologies

Planificador

HTN-PDDL

Especificación de dominios HTN-PDDL

Requisitos, constantes, tipos, predicados y funciones

```
(define (domain viajes)
  (:requirements  :typing  :fluents :derived-predicates
                  :negative-preconditions :htn-expansion)
  (:types      Persona Sitio - object
               Hogar Aeropuerto - Sitio
  (:predicates (en ?p -  Persona ?s – Sitio))
  (:functions (distancia ?x ?y – Sitio))
...
```


HTN-PDDL

Especificación de dominios HTN-PDDL

Tareas y métodos

```
(:task irAeropuerto
   :parameters (?p – Persona ?c – Hogar ?a – Aeropuerto)
   (:method enTaxi …)
   (:method enBus …)
   (:method andando …)  ;; Uff !!
)
```

HTN-PDDL

Especificación de dominios HTN-PDDL

Tareas y métodos

```
(:method enTaxi
   :precondition ()
   :tasks (
         (ir_en_taxi ?p ?c ?a)
         (:inline   (bind ?tarifa
                        (* (distancia ?c ?a) (precio_km))) ())
         (pagar ?p ?tarifa)
   )
)
```


HTN-PDDL

Especificación de dominios HTN-PDDL

```
(:task viajar
   :parameters (?p – Persona ?x ?y – Sitio)
   (:method enAvion
         :precondition (tiene prisa ?p)
         :tasks ( (irAeropuerto ?p ?x GRX)
                  (volar ?p GRX MAD)
                  (irDestino ?p MAD ?y) )
   )
   (:method enCoche
         :precondition (not (tiene_prisa ?p))
         :tasks (irEnCoche ?p ?x ?y)
   )
)
```

HTN-PDDL

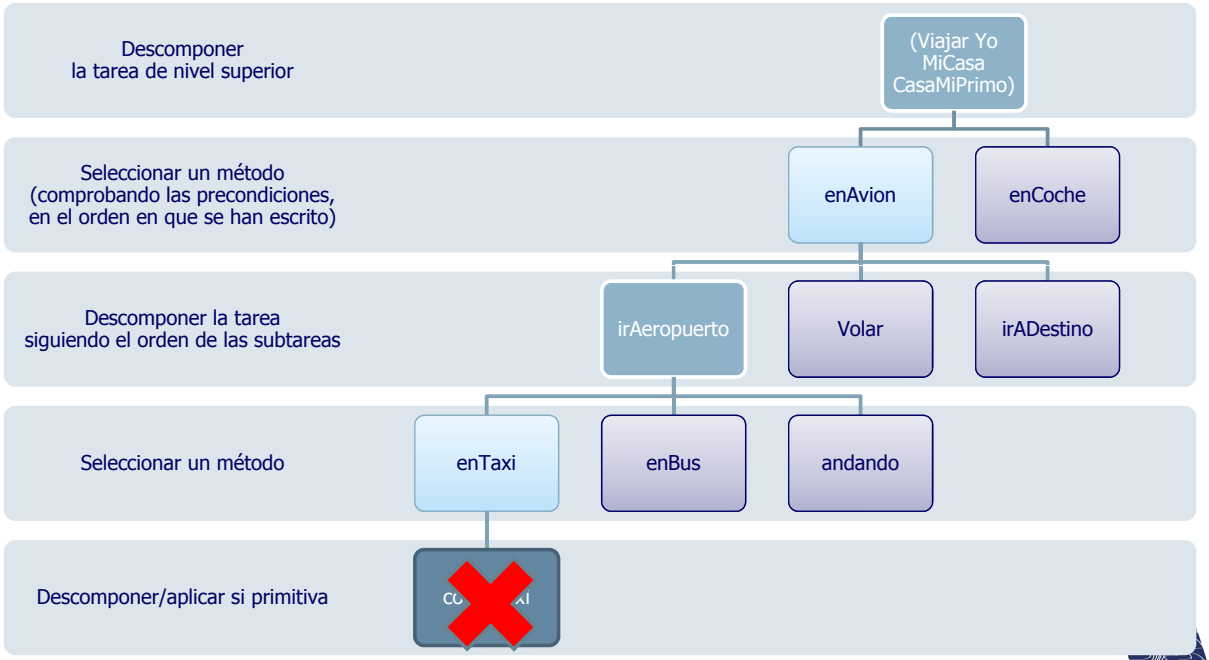
Especificación de problemas HTN-PDDL

Objetos, estado inicial y objetivos

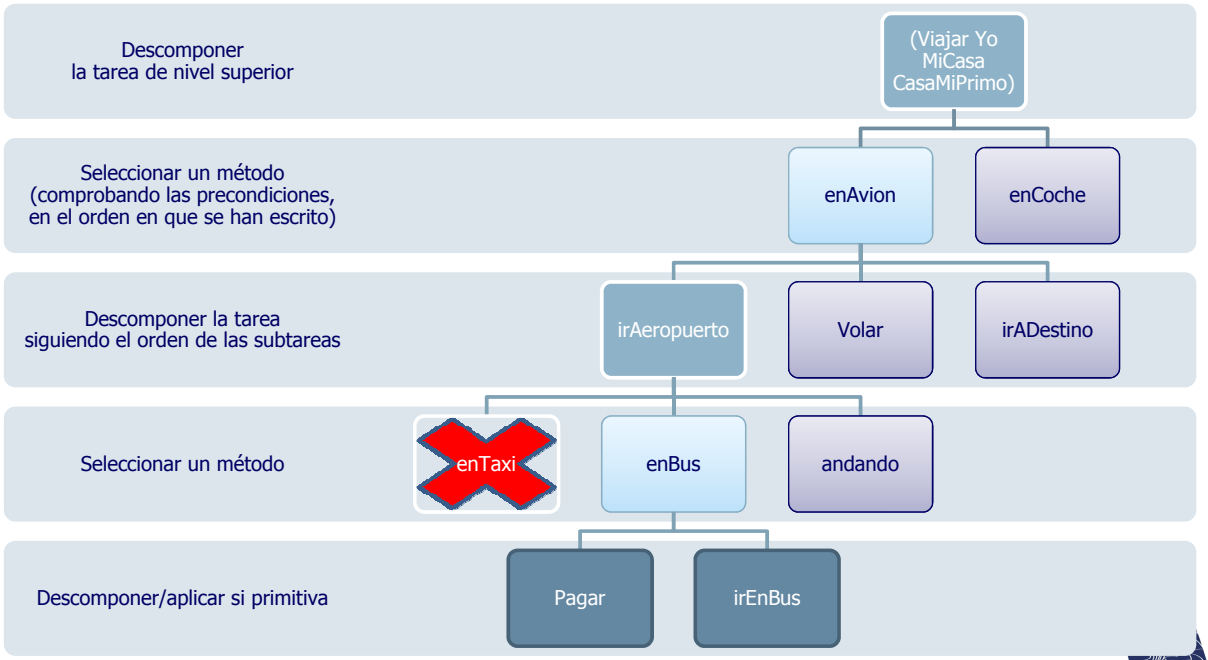
```
(define (problem UnViaje) (:domain Viajes)
    (:objects     MiCasa CasaMiPrimo – Hogar
                  GRX MAD – Aeropuerto
                  Yo – Persona
    )
    (:init        (en Yo MiCasa)
                  (= (distancia MiCasa GRX) 20)
    )
    (:tasks-goal
                  :tasks( (Viajar Yo MiCasa CasaMiPrimo))
    )
)
```


Planificador HTN

Planificador HTN



Planificador HTN



Dominio Depots

Problemas de logística

Dominio Depots

Objetos

(:types place locatable - object
depot distributor - place
truck hoist surface - locatable
pallet crate - surface)

- Cajas [crate]
- Pallets [pallet]
- Almacenes [depot]
- Distribuidores [distributor]
- Grúas [hoist]
- Camiones [truck]

Dominio Depots

Acciones (tareas primitivas)

```
(:action Drive          ; Conducir un camión
   :parameters (?x - truck ?y - place ?z - place)…

(:action Lift           ; Coger una caja con una grúa
   :parameters (?x - hoist ?y - crate ?z - surface ?p - place)…

(:action Drop           ; Dejar una caja con una grúa
   :parameters (?x - hoist ?y - crate ?z - surface ?p - place)…

(:action Load           ; Cargar un camión
   :parameters (?x - hoist ?y - crate ?z - truck ?p - place)..

(:action Unload         ; Descargar un camión
    :parameters (?x - hoist ?y - crate ?z - truck ?p - place)…
```


Dominio Depots

Objetivos (lista de tareas "apilar")

- Tarea objetivo "Main": (:tasks-goal :tasks((Main)))

```
(:task Main :parameters ()
  (!
    (:method itera
       :precondition (AND (apilar ?x ?y))
       :tasks (
          (… ?x ?y )      ; Tarea necesaria para apilar x sobre y
          (:inline () (not (apilar ?x ?y)))  ; Objetivo cumplido
          (Main))         ; Descomposición recursiva
    )
    (:method caso-base  :precondition () :tasks ())
  )
)
```

Acciones primitivas: HTN-PDDL durative-actions

- Objetos del dominio:
  - constantes,
  - tipos
  - predicados,
  - funciones

```
(define (domain viajes)
  (:requirements  :typing  :fluents
    :derived-predicates  :negative-
    preconditions  :htn-expansion)
  (:constants <….>)
  (:types Persona Sitio - object
          Hogar Aeropuerto - Sitio
  (:predicates (en ?p -  Persona ?s - Sitio))
  (:functions (distancia ?x ?y - Sitio)
              (dinero ?p - Persona)
              (velocidad-taxi)
              (precio_km)
)
```

- Acciones primitivas:
  - Representación PDDL
  - Parámetros con tipo,
  - precondiciones,
  - efectos,
  - duración

```
(:durative-action ir_en_taxi
 :parameters (?u - Persona ?o ?d - Sitio)
 :duration (= ?dur (* (distancia ?o ?d)
                      (velocidad_taxi) ))
 :condition (and (en ?u ?o))
 :effect (and (not (en ?u ?o))
              (en ?u ?d)))
```


Otros aspectos útiles de PDDL

- Valores numéricos/funciones
- Especificar cómo calcular duraciones de acciones
- Condiciones con expresiones aritméticas

```
(:durative-action ir_en_taxi
 :parameters (?u - Persona ?o ?d - Sitio)
 :duration (= ?dur (* (distancia ?o ?d)
                      (velocidad_taxi) )
 :condition (and (en ?u ?o)
                 (> (dinero ?u)
                    (* (precio_km)(distancia ?o ?d))
 :effect (and (not (en ?u ?o))
              (en ?u ?d)))
```

Otros aspectos útiles de PDDL

- Valores numéricos/funciones
- Derived literals: reglas de inferencia para "derivar" predicados de las precondiciones

```
(:derived (tiene_dinero ?p - Persona ?org ?dst - Sitio)
          ((> (dinero ?u)
                   (* (precio_km)(distancia ?o ?d)))

(:durative-action ir_en_taxi
 :parameters (?u - Persona ?o ?d - Sitio)
 :duration (= ?dur (* (distancia ?o ?d)
                      (velocidad_taxi) )
 :condition (and (en ?u ?o)
                 (tiene_dinero ?u ?o ?d))
 :effect (and (not (en ?u ?o))
              (en ?u ?d)))
```

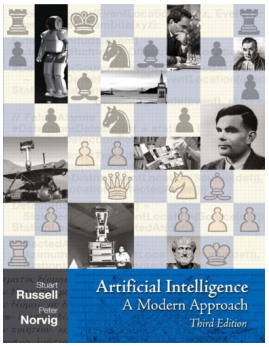

Otros aspectos útiles de PDDL

- Valores numéricos/funciones
- Derived literals: reglas de inferencia para "derivar" predicados de las precondiciones
- Asignación, incremento/decremento de funciones

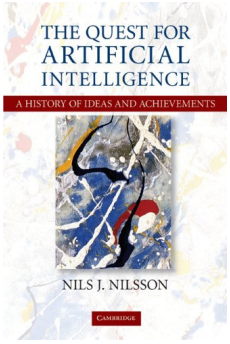
```
(:derived (tiene_dinero ?p - Persona ?org ?dst - Sitio)
          ((> (dinero ?u)
                   (* (precio_km)(distancia ?o ?d)))

(:durative-action ir_en_taxi
 :parameters (?u - Persona ?o ?d - Sitio)
 :duration (= ?dur (* (distancia ?o ?d)
                      (velocidad_taxi) )
 :condition (and (en ?u ?o)
                 (tiene_dinero ?u ?o ?d))
 :effect (and (not (en ?u ?o))
              (en ?u ?d)))

(:durative-action pagar
 :parameters (?u - Persona ?c - number)
 :duration (= ?dur 1)
 :condition (> (- (dinero ?u) ?c) 0)
 :effect (decrease (dinero ?u) ?c)
```

Bibliografía

- Stuart Russell & Peter Norvig:
  Artificial Intelligence:
  A Modern Approach
  Prentice-Hall, 3rd edition, 2009
  ISBN 0136042597
  http://aima.cs.berkeley.edu/



- Nils J. Nilsson
  The Quest for Artificial Intelligence
  Cambridge University Press, 2009
  ISBN 0521122937
  http://ai.stanford.edu/~nilsson/QAI/qai.pdf



Bibliografía

Cursos de planificación

- **MSC Automated Planning**
  School of Informatics
  University of Edinburgh
  http://www.inf.ed.ac.uk/teaching/courses/plan/
  También en Coursera:
  https://www.coursera.org/course/aiplan

- **CS541: Artificial Intelligence Planning**
  USC Viterbi School of Engineering
  University of Southern California
  http://www.isi.edu/~blythe/cs541/