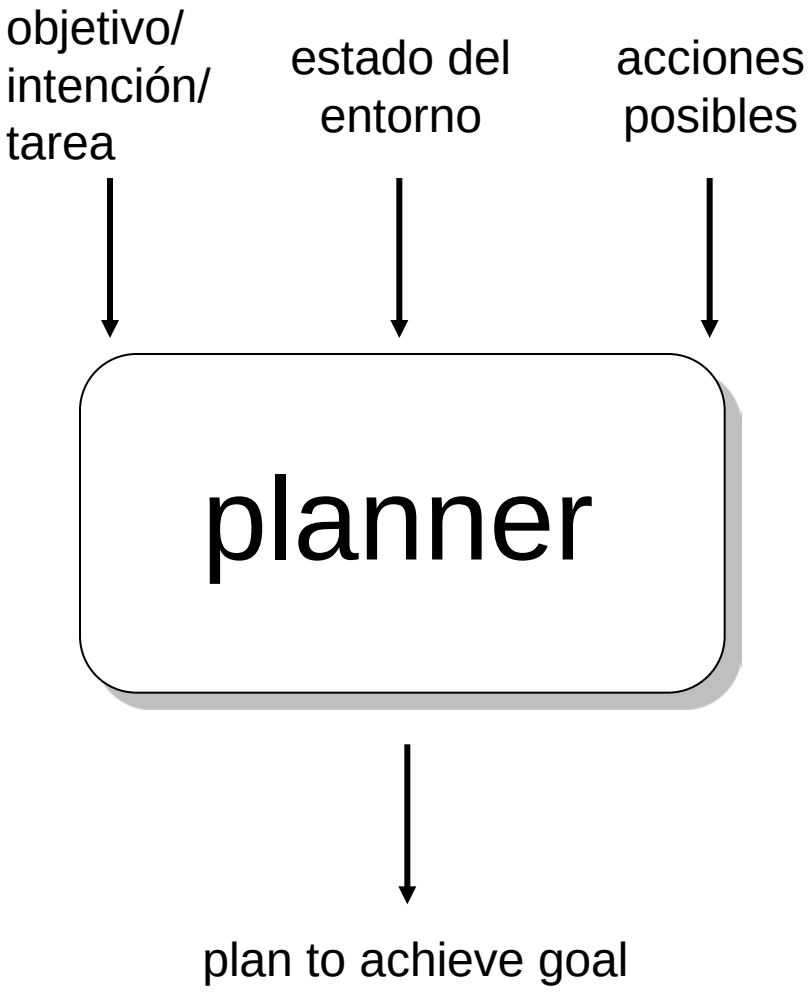
Agentes planificadores (1)

- Desde principios de los '70, la comunidad de IA especializada en planificación se ha preocupado del problema de diseño de agentes artificiales capaces de actuar en un entorno
- La planificación se puede ver como una forma de *programación automática*: el diseño de un curso de acción que satisfará un cierto objetivo
- Dentro de la comunidad de la IA simbolica, se ha asumido desde hace tiempo que algun tipo de sistema planificador debe formar parte de los componentes centrales de cualquier agente artificial

- La idea básica es dotar al agente planificador:
  - representación del objetivo a alcanzar
  - representación de las acciones que puede realizar
  - representación del entorno
  - Capacidad de generar un *plan* para alcanzar el objetivo

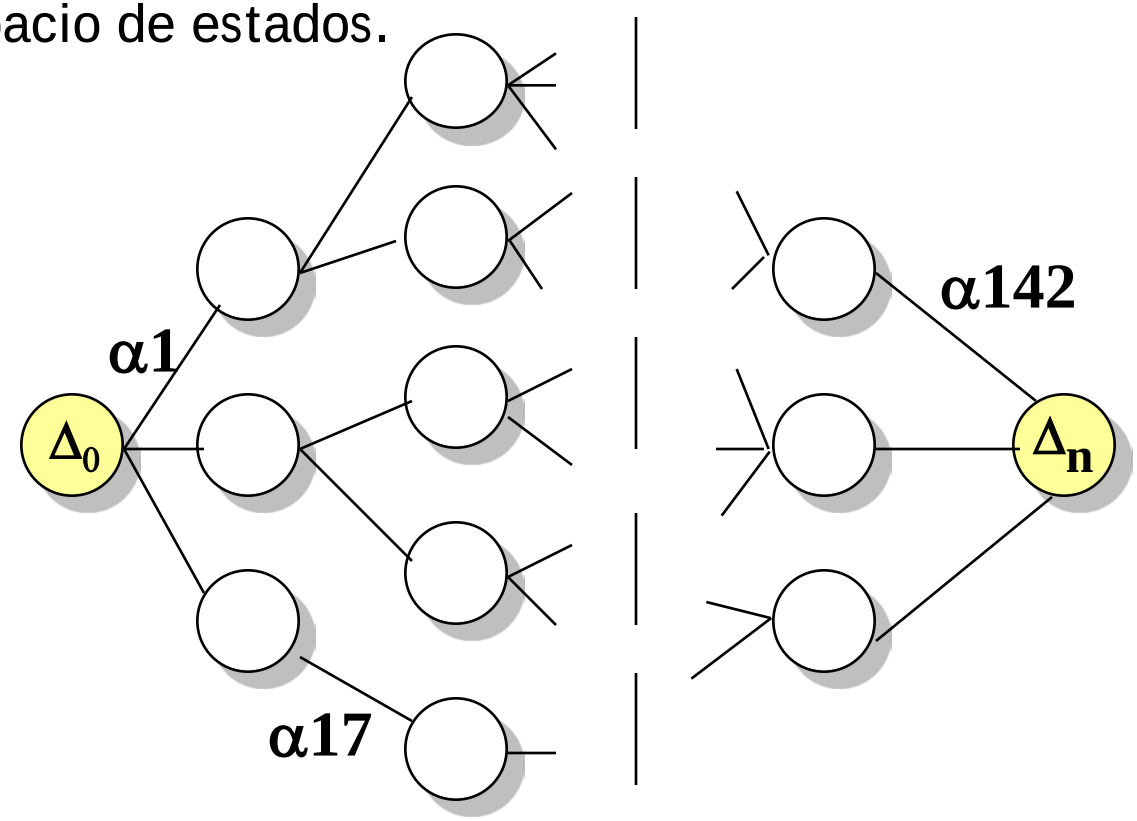
Agentes planificadores (2)



- Pregunta: Como *representar*. . .
  - objetivo a alcanzar
  - estado del entorno
  - acciones disponibles para el agente
  - el propio plan

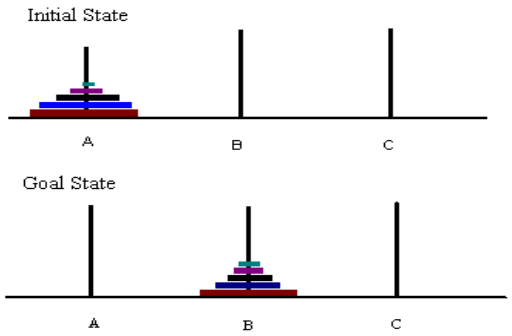
Planificación

- ¿Qué es un plan?
  - Es una secuencia (lista) de acciones, que llevan de un estado inicial a un estado final.
- La planificación se puede ver como un problema de búsqueda en un espacio de estados.

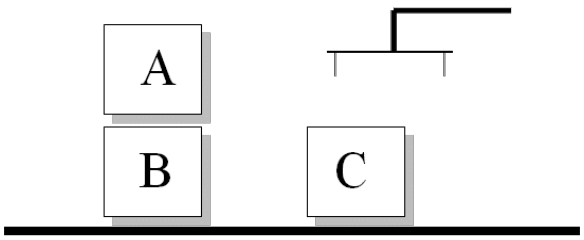


Ejemplos paradigmáticos de planificación

- Hay ejemplos que se repiten de forma reiterada en la literatura de Planificación
  - Torres de Hanoi
  - 8-puzzle, 15-puzzle, ….
  - Mundo de los bloques (Blocks World)

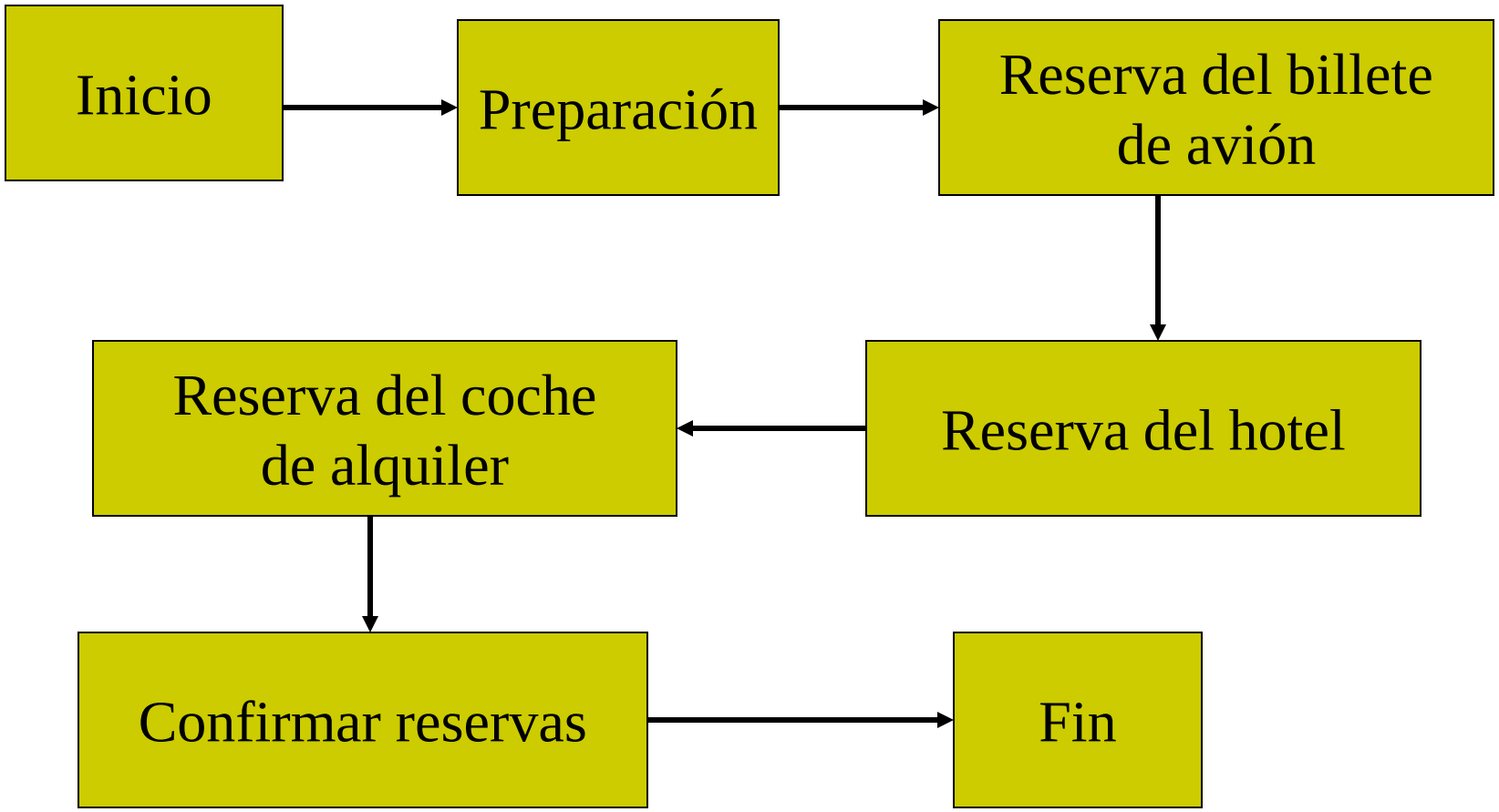


1	3	6	11
9	8	12	
10	4	13	15
2	7	14	5



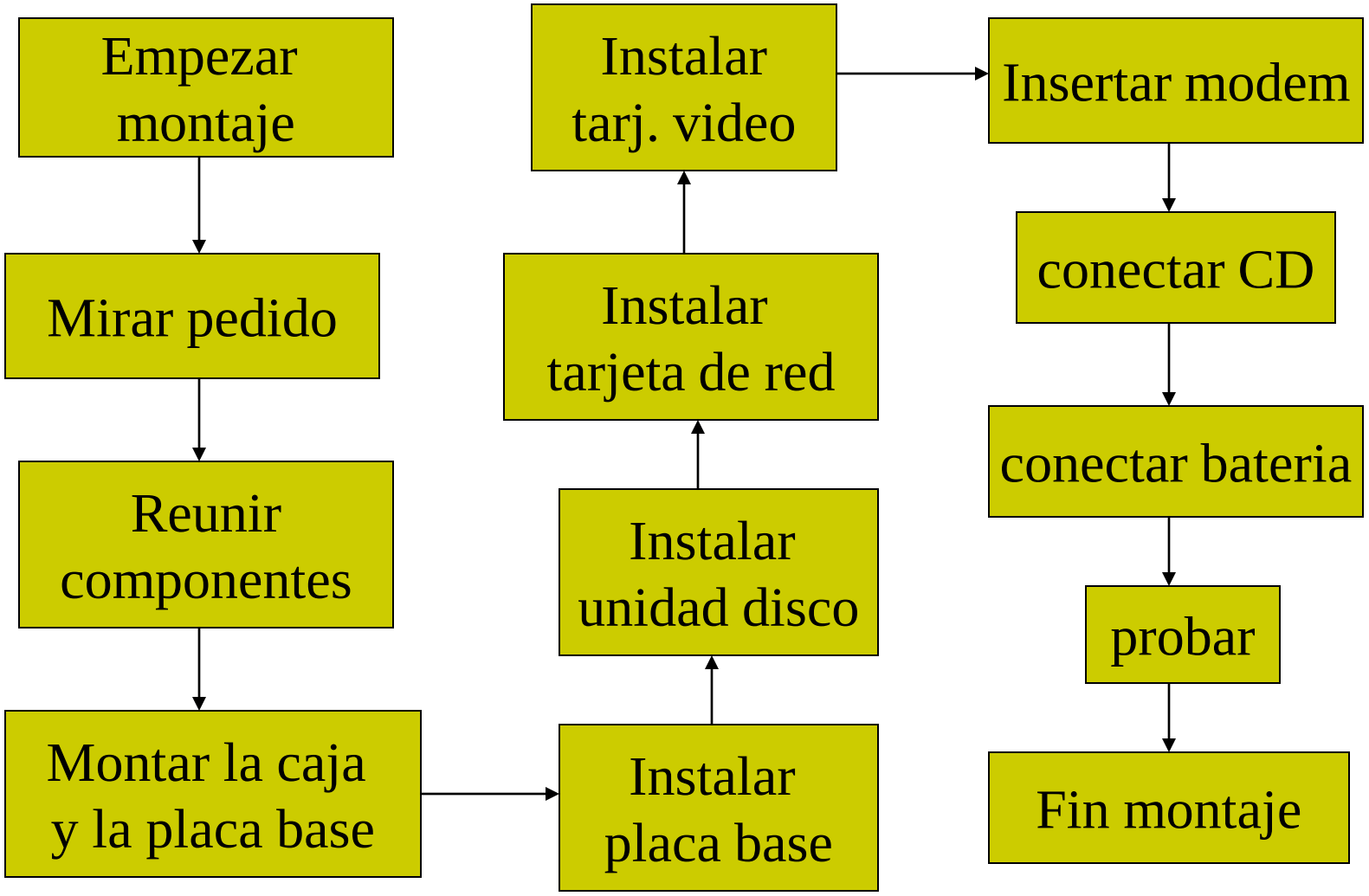
Aplicaciones de la Planificación: en la vida diaria

Ej: planificar un viaje…



[Ejemplo de Han Yu (University of Central Florida)]

Aplicaciones de la Planificación: gestión de workflows



[Ejemplo de Han Yu (University of Central Florida)]

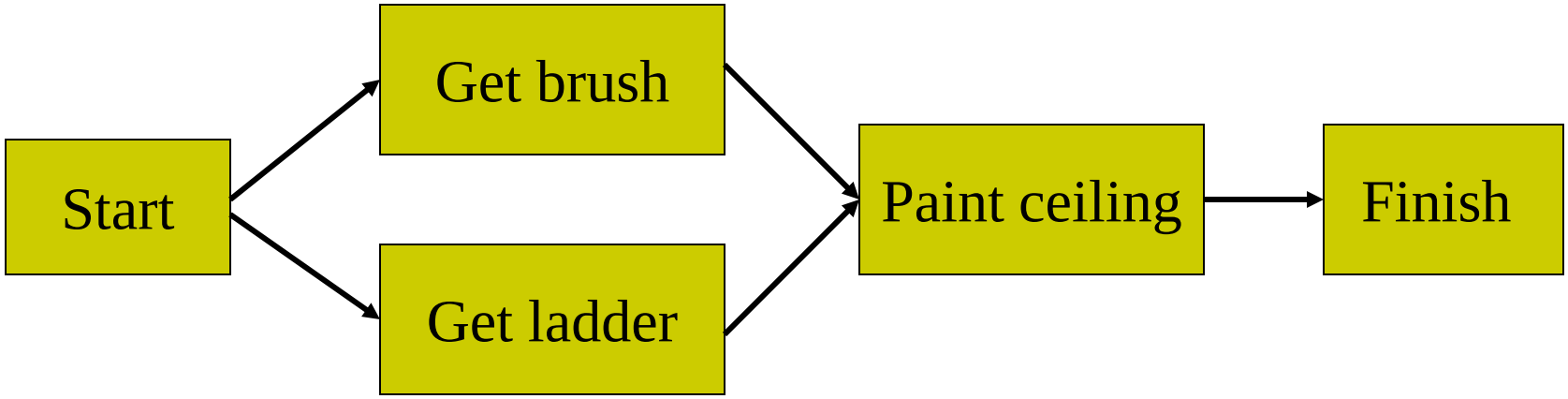
Aplicaciones de la Planificación: exploración espacial

- Mars Exploration Rovers
  - La planificación de las tareas a realizar durante un día marciano se realiza automáticamente por un programa a partir de los objetivos de exploración que fija el personal de misión en la Tierra.

[Fuente: NASA Jet Propulsion Laboratory]

Planes parcialmente ordenados

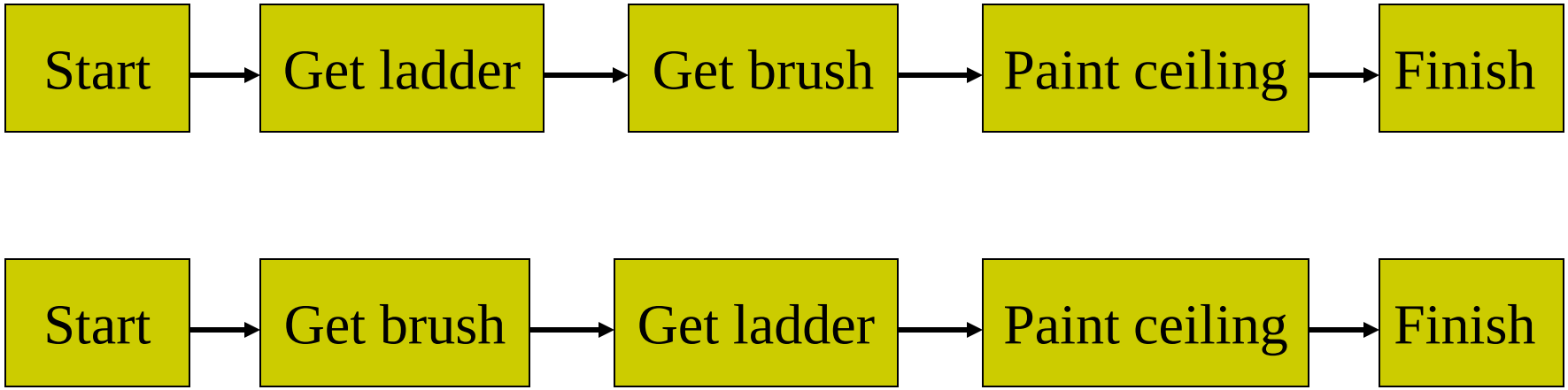
- Plan parcialmente ordenado (Partial-order plan)
  - Compuesto por un conjunto de acciones ordenadas parcialmente
  - Existen restricciones de secuencia en estas acciones
  - Un algoritmo de generación de planes se puede usar para transformar un plan parcialmente ordenado en un plan totalmente ordenado



[Ejemplo de Han Yu (University of Central Florida)]

Planes totalmente ordenados

- Plan totalmente ordenado (Total-order plan)
  - Compuesto por un conjunto de acciones totalmente ordenado



[Ejemplo de Han Yu (University of Central Florida)]

Diferencias entre alg. planificación y alg. búsqueda

- Los algoritmos de búsqueda que hemos visto hasta ahora se interesan sólo en devolver el estado final o estado-solución.
- Los algoritmos de planificación no solo se interesan por encontrar el estado solución, sino en mantener todos los estados intermedios que llevan desde el estado inicial al final.
- Los algoritmos de planificación suelen usar no solo el conocimiento dentro del heurístico, sino también las descripciones de los efectos de las acciones para guiar su búsqueda (utilizan la estructura lógica del problema).
- Muchos algoritmos de planificación reducen la complejidad del problema descomponiendolo en sub-objetivos
  - Esto solo se puede realizar en problemas reales que sean descomponibles o quasi-descomponibles (el planificador descompone el problema y luego resuelve pequeños conflictos al recomponer la solución)

Planificación clásica

- Considera entornos que son
  - completamente observables: el planificador percibe perfectamente el estado del entorno y el efecto de sus acciones en el entorno
  - deterministas: se pueden predecir y predefinir los efectos de todas las acciones
  - finitos: existe un conjunto finito de acciones y de estados
  - estáticos: el entonrno solo cambia cuando el agente planificador actua sobre él.
  - discretos: el entorno se puede describir de forma discreta…
    - Tiempo discreto (se suele medir en ciclos de ejecución)
    - Acciones discretas (las acciones suelen verse como unidades y necesitar un ciclo de ejecución)
    - Objetos discretos: las descripciones de los objetos son discretas
    - Efectos discretos: los efectos de las acciones suelen ser directamente observables una vez se ha ejecutado la acción

Planificación Clásica: teoria formal (I)

- $Ac = \{\alpha_1, \ldots, \alpha_n\}$: un conjunto fijo de acciones.
- $< P_\alpha, D_\alpha, A_\alpha >$ un descriptor para una acción $\alpha \in Ac$
  - P_α es un conjunto de formulas en lógica de primer orden que caracterizan la *precondición* de la acción α
  - D_α es un conjunto de fórmulas en lógica de primer orden que caracterizan aquellos *hechos* que se vuelven *falsos* por la ejecución de α ('delete list')
  - A_α es un conjunto de fórmulas en lógica de primer orden que caracterizan aquellos *hechos* que se vuelven *ciertos* por la ejecución de α ('add list')
- Un **problema de planificación** es una tripleta $<\Delta, O, \gamma>$

Planificación Clásica: teoria formal (II)

- $\pi=(\alpha_1, \ldots, \alpha_n)$: un plan con respecto al problema de planificación $<\Delta, O, \gamma>$ determina una secuencia de $n+1$ modelos:

$$\Delta_0, \Delta_1, \ldots, \Delta_n$$

- donde $\Delta_0 = \Delta$ y

$$\Delta_i = (\Delta_{i-1} \setminus D_{\alpha_i}) \cup A_{\alpha_i} \quad \text{for} 1 \leq i \leq n$$

- Un plan π es aceptable ssi $\Delta_{i-1} \vdash P_{\alpha_i}$,para todo $1 \leq i \leq n$
- Un plan π es correcto ssi
  - π es aceptable, y
  - $\Delta_n \vdash \gamma$

Lenguaje de problemas de planificación (1)

- **Representación de estados**: los planificadores descomponen el mundo en condiciones lógicas, representando un estado como una conjunción de literales positivos:
  - Proposiciones: $Pobre \wedge Desconocido$
  - Literales de 1er orden: $En(Avion1, Melbourne) \wedge En(Avion2, Sydney)$

- **Representación de objetivos**: un objetivo es un estado parcialmente especificado
  - Un estado s **satisface** un objetivo o si s contiene todos los atomos de o (y posiblemente algunos más)
    - Eg: el estado $Rico \wedge Famoso \wedge Miserable$ satisface el objetivo $Rico \wedge Famoso$

Lenguaje de problemas de planificación (2)

- **Representación de acciones**: Las acciones se especifican en terminos de las precondiciones que se han de cumplir antes de que se puedan ejecutar y de los efectos que producen una vez se han ejecutado

$$
\begin{aligned}
&\textit{Acción}(\textit{volar}(av, orig, dest),\\
&\quad \textit{PRECOND: } \textit{En}(av, orig) \wedge \textit{Avion}(av) \wedge \textit{Aeropuerto}(orig) \wedge \textit{Aeropuerto}(dest)\\
&\quad \textit{EFECTO: } \neg\, \textit{En}(av, orig) \wedge \textit{En}(av, dest)\\
&)
\end{aligned}
$$

  - La precondición es una conjunción de literales positivos que especifica que debe de ser verdadero en un estado antes de que la accion se ejecute. Todas las variables en la precondición han de aparecer en la lista de parámetros de la acción.
  - El efecto es una conjunción de literales describiendo como cambia el estado cuando la acción se ejecuta. Todas las variables han de aparecer también en la lista de parámetros de la acción.

Lenguaje de problemas de planificación (3)

- Una acción es aplicable en cualquier estado que satisfaga la precondición
  - En 1er orden: existe una substitución para las variables en la precondición. Por ejemplo, el estado

    $En(A1, JFK) \wedge Avion(A1) \wedge En(A2, SFO) \wedge Avion(A2) \wedge$
    $Aeropuerto(JFK) \wedge Aeropuerto(SFO)$

    satisface la precondición de la acción *volar*:

    $En(a, orig) \wedge Avion(a) \wedge Aeropuerto(orig) \wedge Aeropuerto(dest)$

- El resultado de ejecutar la acción en un estado s es un estado s' al que se añaden los literales positivos del efecto y se eliminan los literales negativos
  - Por ejemplo, el efecto de la acción volar sobre el estado anterior:

    $\mathbf{En(A1, SFO)} \wedge Avion(A1) \wedge En(A2, SFO) \wedge Avion(A2) \wedge$
    $Aeropuerto(JFK) \wedge Aeropuerto(SFO)$

    Se eliminó: En(A1, JFK)

Ejemplo 1: Transporte aereo de carga

- Dos cargas (C1 y C2) estan en 2 aeropuertos (SFO, JFK)
- Tenemos dos aviones (A1 y A2) para transportar las cargas, uno en cada aeropuerto
- Describimos el estado inicial así:

$$Inicio(\ En(C1, SFO) \wedge En(C2, JFK) \wedge En(A1, SFO) \wedge En(A2, JFK) \wedge Carga(C1) \wedge Carga(C2) \wedge Avion(A1) \wedge Avion(A2) \wedge Aeropuerto(SFO) \wedge Aeropuerto(JFK)\ )$$

- El objetivo es que C1 acabe en JFK y C2 en SFO
- Describimos el objetivo así:

$$Objetivo(\ En(C1, JFK) \wedge En(C2, SFO)\ )$$

Ejemplo 1: Transporte aereo de carga

- Describimos las acciones de cargar, descargar y volar:

Acción(carga(c, av, aerop),
PRECOND: En(c, aerop) ∧ En(av, aerop) ∧ Carga(c) ∧ Avion(av) ∧ Aeropuerto(aerop)
EFECTO: ¬En(c, aerop) ∧ Dentro(c, av)
)
Acción(descarga(c, av, aerop),
PRECOND: Dentro(c, av) ∧ En(av, aerop) ∧ Carga(c) ∧ Avion(av) ∧ Aeropuerto(aerop)
EFECTO: En(c, aerop) ∧ ¬Dentro(c, av)
)
Acción(volar(a, orig, dest),
PRECOND: En(a, orig) ∧ Avion(a) ∧ Aeropuerto(orig) ∧ Aeropuerto(dest)
EFECTO: ¬ En(a, orig) ∧ En(a, dest)
)

Ejemplo 1: Transporte aereo de carga

- Solución: el plan lo compone una secuencia de acciones.
- En este caso hay varias soluciones
  - Ej. Solucion 1: usamos los dos aviones para hacer el traslado

    [carga(C1, A1, SFO), vuela(A1, SFO, JFK), descarga(C1, A1, JFK)
    carga(C2, A2, JFK), vuela (A2, JFK, SFO), descarga(C2, A2, SFO)]

  - Ej. Solucion 2: usamos solo un avión

    [carga(C1, A1, SFO), vuela(A1, SFO, JFK), descarga(C1, A1, JFK)
    carga(C2, A1, JFK), vuela (A1, JFK, SFO), descarga(C2, A1, SFO)]

Lenguaje de representación: STRIPS (ejemplo)

Fichero de descripción del dominio

```
pick-up(x):

clear(x),ontable(x),handempty().
clear(x),ontable(x),handempty().
holding(x).


put-down(x):

holding(x).
holding(x).
clear(x),handempty(),ontable(x).


stack(x,y):

holding(x),clear(y).
holding(x),clear(y).
clear(x),handempty(),on(x,y).
```

```
unstack(x,y):

on(x,y),clear(x),handempty().
on(x,y),clear(x),handempty().
holding(x),clear(y).
```

Fichero de descripción del problema

```
clear(C),clear(A),clear(B),clear(D),
ontable(C),ontable(A),ontable(B),
ontable(D),handempty().

on(D,C),on(C,B),on(B,A).
```

Lenguaje de representación: PDDL

- Desde 1998 la comunidad de investigadores en planificación ha desarrollado un lenguaje standard de descripción de planes: Planning Domain Description Language (PDDL)
- Objetivo inicial: lenguaje común para competición mundial de planificadores
- En la actualidad se ha convertido en un estándar de facto
- WARNINGS:
  - existen varias versiones de PDDL, desde la 1.0 a la 3.1, cada una de ellas con diferentes niveles de expresividad
  - No existe ningún planificador que soporte la especificación 3.1 completa, sino subconjuntos de ella. → hay que revisar la documentación (escasa) del plannerque se está usando para saber que soporta y que no.

Lenguaje de representación: PDDL (sintaxis)

Fichero de descripción del dominio

```
(define (domain DOMAIN_NAME)
  (:requirements [:strips] [:equality] [:typing] [:adl])
  (:predicates (PREDICATE_1_NAME [?A1 ?A2 ... ?AN])
               (PREDICATE_2_NAME [?A1 ?A2 ... ?AN])
           ...)

  (:action ACTION_1_NAME
    [:parameters (?P1 ?P2 ... ?PN)]
    [:precondition PRECOND_FORMULA]
    [:effect EFFECT_FORMULA]
   )

  (:action ACTION_2_NAME
    ...)

  ...)
```

Como hay diferentes niveles de expresividad posibles, cada descripción en PDDL dice los requisitos necesarios para procesarla. Los más comunes son:

`:strips` → expresividad como en STRIPS
`:equality` → el dominio usa el predicado =
`:typing` → el dominio define tipos de vars.
`:adl` → expresividad extendida:
1) disyunciones y cuantificadores en precondiciones y objetivos,
2) Efectos cuantificados y condicionales

Lenguaje de representación: PDDL (sintaxis)

Fichero de descripción del problema

```
(define (problem PROBLEM_NAME)
  (:domain DOMAIN_NAME)
  (:objects OBJ1 OBJ2 ... OBJ_N)
  (:init ATOM1 ATOM2 ... ATOM_N)
  (:goal CONDITION_FORMULA) )
```

Lenguaje de representación: PDDL (ejemplo)

Fichero de descripción del dominio

```
(define (domain driverlog)
  (:requirements :strips :typing)
  (:types location locatable - object
          driver truck obj - locatable

  )
  (:predicates
     (at ?obj - locatable ?loc - location)
     (in ?obj1 - obj ?obj - truck)
     (driving ?d - driver ?v - truck)
     (link ?x ?y - location) (path ?x ?y - location)
     (empty ?v - truck)
  )
  (:action LOAD-TRUCK
  :parameters
   (?obj - obj
    ?truck - truck
    ?loc - location)
  :precondition
   (and (at ?truck ?loc) (at ?obj ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?truck)))
```

Fichero de descripción del problema

```
(define (problem DLOG-2-2-2)
    (:domain driverlog)
    (:objects
      driver1 - driver
      truck1 - truck
      package1 - obj
      s0 - location
      s1 - location ...)
    (:init
    (at driver1 s12)
    (at truck1 s0)
    (empty truck1)
    (at package1 s0)
    (path s1 p1-0)
    (path p1-0 s1)
    ...
    (link s0 s1)
    (link s1 s0)
    ... )
    (:goal (and (at driver1 s1)
    (at truck1 s1)
    (at package1 s0)
    )))
```