Tema-3-Sistemas-de-Planificacion...

ParmigianoReg

Tecnologias de los Sistemas Inteligentes

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Tema 3 - Sistemas de Planificación en IA

Profesor Antonio González Muñoz, Curso 20-21

Razonamiento sobre Acciones y Planificación Clásica:

- ¿Qué es planificar?
  - Es razonar sobre acciones. El eje central de la planificación son las acciones, conocer como son las acciones, representarlas y razonar sobre las acciones.
- ¿Qué es un plan?
  - Es una secuencia de acciones que resuelve un problema en particular.
- Búsqueda y Planificación
  - El planteamiento de Planificación se asemeja mucho al de búsqueda.
  - La diferencia entre ambos es los tipos de problemas en los que se aplica, y lo que requiere. Cuando se trata de búsqueda, esta va analizando como ordenar las acciones para conseguir un fin determinado y eso lo requiere la planificación, pero la planificación va un paso más allá porque como trata de problemas reales necesita un modelo de representación del conocimiento, algo que la búsqueda no lo utiliza.

Problemas de Planificación:

- Complejidad del mundo real:
  - Frente a un dominio que es real, la búsqueda por sí sola no puede resolver ciertos problemas más complejos; se deben utilizar los modelos basados en la lógica para capturar esa complejidad del mundo real.
- Otros problemas:
  - **Problema del Marco (Frame Problem)**
    - Problema más importante de la planificación, que solamente se puede mitigar pero no evitar del todo.
    - Es un problema esencialmente asociado a la representación del conocimiento porque lo que se plantea es como se puede representar una acción para poder razonar eficientemente como es el mundo antes y después de realizar una acción, ya que las acciones cambian el mundo.
    - Puede parecer un problema sencillo, pero el problema es que primero se tiene un estado antes y un estado después de la acción en el que habrá cambiado algo por la acción; aquí la esencia de la acción es que debe indicar como se modifica ese estado.
    - La dificultad es dual; cuando se describe la acción es normal que se describa que cambia pero el problema surge en poder describir que es lo que no cambia luego de la acción y luego como se razona con eso que no cambia. Por eso se llama del "marco", es del contexto.
    - Una solución sería describir todo, lo que cambia y que cambia pero esto es mucho más complejo y se vuelve inviable.
  - **Efectos Dependientes del Contexto**
    - Este problema tiene que ver, obviamente, con los efectos: los cambios que se tienen que describir
    - Por ejemplo, describir el tiempo es complicado ya que el tiempo que se toma realizar una acción, ir de la casa a la universidad, varía por la hora y el día. Puede que a veces tome más y a veces menos tiempo, entonces, es un efecto que depende del contexto.
    - Se tienen que considerar las acciones raras que pueden que sucedan muy pocas veces para que el razonador sepa cómo lidiar con ellas si llegan a suceder.
  - **Problema de la Cualificación**
    - Tiene que ver con la representación de una acción, es sobre qué condiciones se puede aplicar una acción
    - Todas las acciones tienen precondiciones, los requisitos por los cuales se puede aplicar una acción.
    - El problema es ¿cómo definir las precondiciones?, porque a veces no es sencillo saber cuáles podrían ser y cómo representarlo.
    - **Ejemplo**: Se tiene un coche que se conduce solo, ¿qué precondiciones necesita para que ponga

en marcha el coche el agente? Se pueden añadir las usuales, que el motor esté encendido, que esté en neutral, que haya gasolina. Pero, puede suceder que en un día se cumplen todas las precondiciones y no enciende el coche porque alguien le metió un plátano al tubo de escape, entonces, ¿también debería añadirse otra precondición para verificar que no hayan plátanos?
    - El problema es cuando existen precondiciones que son muy infrecuentes y que es absurdo comprobarlas siempre, entonces, pueden existir situaciones en la que las acciones no puedan realizarse aunque las precondiciones originales se cumplan.

Modelos previos a la Planificación
Son modelos anteriores a la Planificación como tal pero que pretendían resolver la misma clase de problemas; no se consideran como Planificación porque solo eran capaces de resolver problemas muy simples pero aportaron mucha información para llegar a la Planificación.

- **El cálculo de situaciones (de Green)**
    - Utiliza la lógica de predicados para modelar el problema en vez de los modelos icónicos, ya que se tenía un modelo deductivo muy potente que era independiente del problema; se intentó adaptar esto para planificar sobre un problema.
    - Se intenta "demostrar" llegar al estado que deseo por medio de las acciones que serían ahora axiomas, pero entendiendo que los cambios que pasan en el mundo tienen que representarse.
    - Para evitar el problema de la monotonía, o sea, que el valor de verdad de los predicados cambie cuando cambia el estado del problema. Se introduce una variable a los predicados de estado, por lo tanto se puede tener que los predicados sean ciertos en un estado pero no en otro.
    - Las acciones se representan como una función $hacer$(acción, estado) la cual genera un nuevo estado dependiendo de la acción, los efectos se expresan como fbfs, ya sean efectos positivos o negativos.
    - El problema es que surge el **problema del marco**, porque los efectos son locales y por eso no se sabe de lo que no cambia. Esto se intentó mitigar con Axiomas de Marco que son acciones para lo que no cambia
    - Sucede el problema de la cualificación, de las precondiciones, si se intentan añadir también predicados para todo lo que puede suceder se vuelve inviable la inferencia lógica.
    - A pesar de esto, este sistema puede funcionar en problemas reducidos; ya que era muy costosa computacionalmente y que no realmente se resuelve el problema del marco. Es teóricamente correcto.
    - Lo más **importante** que **aportó** a la planificación es que la **representación por lógica de predicados** es muy potente.
- **Reducción de diferencias**
    - Diseñado en paralelo al sistema anterior, se quería diseñar un sistema que resolviera los problemas de manera similar a como lo hace el Humano. Era el General Problem Solver y la Reducción de Diferencias era como el GPS funcionaba.
    - Se desea reducir las diferencias entre el estado actual y el estado objetivo, si no hay diferencias entonces el problema está resuelto.
    - Es un proceso que se va descomponiendo el problema, se busca la acción más relevante para el problema, se aplica y se repite el proceso con los trozos restantes del problema.
    - La idea es encontrar una diferencia entre el objeto inicial y el objeto final. Si no hay diferencias, el problema está resuelto. El objeto es el estado del problema.
    - Si hay diferencias, se considera la más importante (las diferencias están ordenadas), se encuentra un operador para reducir esa diferencia y ahora se comparan las precondiciones del operador con el estado actual, si no hay diferencias es que se puede aplicar y se aplica. Si hay diferencias, se tratará de reducirlas.
    - Se repite el proceso tomando como objeto inicial el objeto producido por la aplicación del operador y el objeto final.
    - Uno de los problemas de este sistema es que requiere de mucha infraestructura para realizar el análisis de diferencias, era muy difícil representar problemas más complejos con listas.
    - También no se tenían métodos de búsquedas eficientes, se utilizaban métodos retroactivos. Aun así era capaz de resolver problemas simples.

- **Lecciones aprendidas**: Es muy útil utilizar lógica de predicados para representar esta clase de problemas y también es bueno utilizar heurísticas para dirigir la búsqueda y tener un esquema más sencillo de diferencias.

quieres trabajar en Wuolah??

TE BUSCAMOS

sin ánimo de lucro, chequea esto:

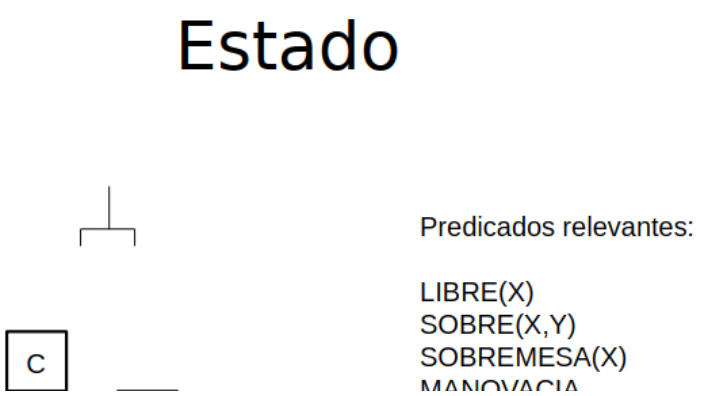
tú puedes ayudarnos a llevar
WUOLAH al siguiente nivel
(o alguien que conozcas)

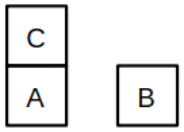
Planificación Clásica

- Surge como un descendiente directo de los dos sistemas anteriores. En primer lugar la PC adopta una serie de suposiciones:
  - El sistema tiene un número finito de estados o situaciones.
  - El sistema es perfectamente observable, es decir, se tiene conocimiento completo del estado del sistema.
  - El sistema es determinista, es decir, la aplicación de una acción a un estado conduce siempre a un mismo estado, no hay incertidumbre.
  - El sistema es estático, es decir, el sistema permanece en el mismo estado hasta que se aplique una acción
  - Los objetivos se conocen antes de empezar la planificación y no cambian.
  - Un plan solución de un problema de planificación es una secuencia de acciones finita y linealmente ordenada.
  - No se contempla el razonamiento temporal y numérico, por lo que la calidad del plan se determina por el número de acciones del mismo.
  - La tarea de planificación consiste en construir un plan completo que satisface el objetivo antes de la ejecución de cualquier parte del mismo.
- Como se puede ver, tiene unas cuantas restricciones. Existen otros tipos de planificación que relajan estas suposiciones.
- Un problema en Planificación Clásica se formula a través de los siguientes elementos:
  - Un conjunto de fórmulas atómicas, denominadas hechos o literales, que representan la información relevante del problema.
  - Un conjunto de operadores definidos en el dominio del problema.
  - Un conjunto inicial de hechos que forman la situación inicial del problema.
  - Un conjunto final de hechos que deben formar parte de la situación final del problema.
- El primer sistema de planificación se llamó STRIPS y condicionó la mayoría de trabajos de planificación desde comienzos de los años 70. Es la base de los planificadores actuales pero ahora está en desuso.
- STRIPS se utilizó para planificar las acciones del robot Shakey.

STRIPS: Standford Research Institute Problem Solver (1971)

- Representación del conocimiento, heredada del Cálculo de Situaciones.
  - Describe **estados** con lógica de predicados, pero sabiendo los problemas que daba anteriormente, se restringe para evitarlos, para ello se utiliza la Hipótesis del Mundo Cerrado.
    - Todo predicado que no está descrito en un estado se supone que es falso, esto simplifica muchísimo más describir los estados, esto aumenta la eficiencia.
    - Esto impide que se maneje incertidumbre.
    - Los estados se describen como una **conjunción** de literales básicos, o sea, son predicados completamente instanciados. El trabajar con solamente con conjunciones, para saber si una acción se puede aplicar en un estado es mucho más eficiente y rápido hacerlo con conjunciones.
    - **No se admiten disyunciones**, porque esto implica incertidumbre. Tampoco se admiten implicaciones (axiomas).

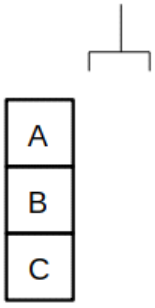




SOBRE(X,Y)
SOBREMESA(X)
MANOVACIA
COGIDO(X)

LIBRE(B) ∧ SOBRE(C,A) ∧SOBREMESA(A) ∧ LIBRE(C) ∧ MANOVACIA ∧ SOBREMESA(B)

- Describe **objetivos**
  - El objetivo es una orden, lo que se desea que sea cierto. No necesariamente corresponde a un estado.
  - En el ejemplo coincide con un estado pero no tiene por qué ser siempre así.
  - El objetivo se puede formar por conjunciones de literales básicos (que no tienen variables) y también órdenes que tengan variables pero deben estar cuantificadas existencialmente (∃), por ejemplo:
    - Que exista un bloque que esté sobre A, $\exists x\ sobre(A,x)$
    - Que exista un bloque que esté sobre A y B, $\exists x\ sobre(A,x) \wedge sobre(B,x)$.
  - No se pueden resolver cuantificadores universales (∀)



SOBRE(A,B) ∧ SOBRE(B,C)

- Describe **operaciones**
  - Componente más importante de STRIPS, los estados y objetivos meramente restringen la lógica de predicados.
  - Las reglas tienen un nombre y unas variables que utiliza.
  - No utiliza como tal la lógica de predicados como lo hace el sistema de Green, la utiliza parcialmente.
  - Se describe en base a tres estructuras:
    - Fórmula de **precondición**, que es una fórmula de lógica totalmente.
    - Lista de **supresión**, lista de predicados y con variables libres sin cuantificación. Son aquellos predicados que se eliminan de la conjunción.
    - Lista de **adición**, lista de predicados y con variables libres sin cuantificación. Son aquellos predicados que se añaden a la conjunción.
  - Se podrá aplicar una acción sobre un estado siempre que la precondición sea verdad en ese estado.
    - *"Si se tiene una acción a que se quiere aplicar al estado S, se podrá hacer si y solo si la precondición de la acción es cierto en ese estado"*

Nombre de la acción y variables usadas
Fórmula de Precondición
Lista de supresión
Lista de Adición

Lista de Adición

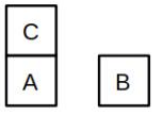
$$(a \in \text{ACTIONS}(s)) \Leftrightarrow s \models \text{PRECOND}(a)$$

$$\text{RESULT}(s,a) = (s - \text{DEL}(a)) \cup \text{ADD}(a)$$

- Para demostrar que una fórmula parte de otra fórmula, se utilizan mecanismos deductivos, pero como los estados y la precondición son una conjunción de literales el mecanismo de deducción es muy eficiente. Se transforma en un esquema simple de emparejamiento del estado con la precondición.
- ¿Cómo se obtiene de una acción y un estado, otro estado nuevo? Para eso se tienen las listas de supresión y de adición.

Modelo de regla tipo STRIPS

- Coger(x)
- FP: ∃x tal que Sobremesa(x) ∧ Libre(x) ∧ Manovacia
- LA: COGIDO(x)
- LS: SOBREMESA(x), LIBRE(x), MANOVACIA



LIBRE(B) ∧ SOBRE(C,A) ∧SOBREMESA(A) ∧ LIBRE(C) ∧ MANOVACIA ∧ SOBREMESA(B)

~~LIBRE(B)~~ ∧ SOBRE(C,A) ∧SOBREMESA(A) ∧ LIBRE(C) ∧ ~~MANOVACIA~~ ∧ ~~SOBREMESA(B)~~

SOBRE(C,A) ∧SOBREMESA(A) ∧ LIBRE(C) ∧ COGIDO(B)

- STRIPS está hecho de esta manera para poder mitigar el problema del marco, con las listas de supresión y adición y supone que el resto de las cosas que no están en ninguna de esas dos listas no se ve afectado. No hay necesidad de deducir las cosas que no cambian en las acciones.
- Esta es una de las aportaciones más importantes de STRIPS, hace que el razonamiento sea mucho más eficiente ya que no usa como tal la lógica de predicados sino acciones que quitan y ponen cosas en el estado.
- STRIPS está muy limitado y por ello había necesidad de extenderlo.

Lenguaje de Planificación PDDL

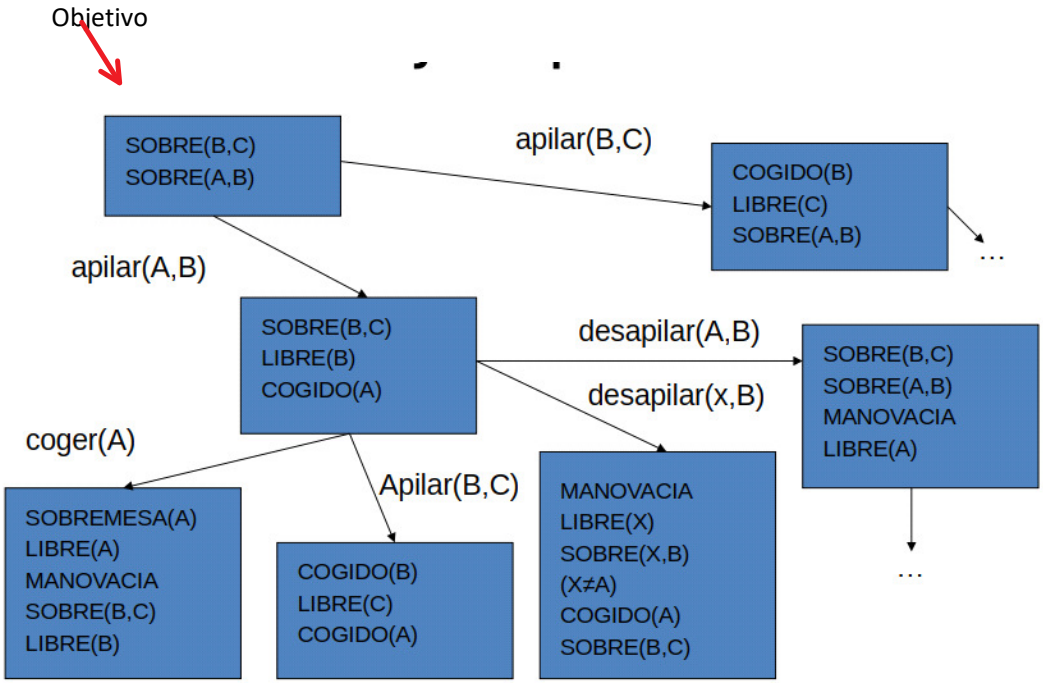
- El lenguaje que utiliza STRIPS es otra de las aportaciones más importantes, ya que ha influido en todos los planificadores que se han diseñado a partir de él. El PDDL contiene el propio lenguaje de STRIPS con ADL y extensiones.
- Es un intento de estandarizar los lenguajes para describir los problemas y dominios de planificación, desarrollado para permitir competiciones entre planificadores.
- Una definición en PDDL consiste en:
  - Una definición del dominio.
  - Una definición del problema.
- Requerimientos:

- :strips, :equality, :typing, :adl (Para usar disyunciones, cuantificadores en precondiciones y objetivos, efectos condicionales, etc)

- Otras características:
  - Acciones con duración.
  - Expresiones y variables numéricas.
  - Métricas del problema.
  - Ventanas temporales.
  - Restricciones duras y blandas sobre el plan.

Planificación como búsqueda en un espacio de estados

- STRIPS usa un modelo de planificación particular, aunque existían dos maneras de realizar esta búsqueda: Por Progresión y Regresión, como algoritmos A* o similares, ¿por qué no se utilizaron?
  - Existen algoritmos de **progresión** válidos pero no se utilizó por el problema de la complejidad del mundo, al factor de ramificación de un estado puede ser muy muy alto, cuando se desarrolló STRIPS no se había encontrado manera de paliar esto: no se logró encontrar una **heurística potente**, por lo que no se utiliza la **progresión**.
  - Por **regresión**, el factor de ramificación no crece tanto como la progresión, lo cual es muy útil. STRIPS se plantea utilizar regresión, pero no es directo ya que los objetivos no son necesariamente estados entonces, ¿cómo se transforma un objetivo a un estado al que retroceder?
    - Se desea tener un estado donde se verifique el objetivo que se tiene, entonces, lo que se busca es un estado donde si se aplica un operador se obtiene el objetivo o lo que es lo mismo: se aplica el operador a la inversa; ya que esto genera un estado donde el operador original es aplicable y si se aplica se satisface al objetivo.
    - ¿Cómo se construye? Dado el objetivo y un literal del mismo, si se quiere conseguir ese literal se debe buscar un operador que en su lista de adición lo contenga o algo que con una substitución lo genere.
    - Para los antecesores se le añade al objetivo la fórmula de precondición del estado actual.
    - Si Q_u es un literal de la lista de adición del operador particularizado la regresión es V.
    - Si está en la lista de supresión del operador, la regresión es F, cuando se tiene una conjunción con Falso, es imposible.
    - En otro caso, la regresión es Q_u.

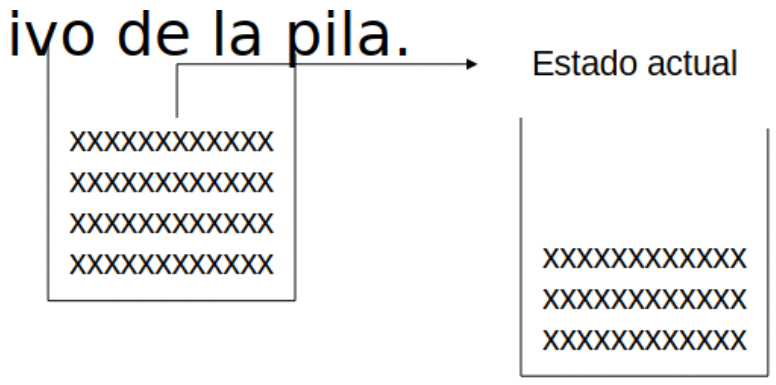


  - De nuevo, STRIPS no utiliza la regresión como tal, ya que aunque se pensaba que iba a generar menos

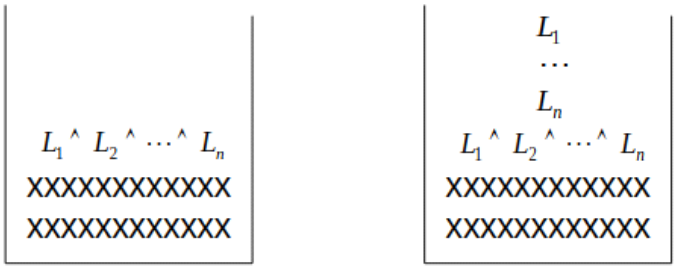
ramificación, se han generado una cantidad considerable y además los estados que se obtienen son bastante más complejos. Se están analizando todos los órdenes posibles para resolver los literales: se decide descomponer el problema para que no intervenga el orden.
    - STRIPS maneja un modelo por **regresión basado en descomposición**.
- Ni los métodos hacia delante ni hacia atrás no podrían ser eficientes sin una buena heurística generalista; posteriormente si se han podido encontrar haciendo uso de la técnica de modelos simplificados.
    - Basadas en ignorar precondiciones o ignorar la lista de supresión.

Cómo funciona STRIPS

- Regresión por descomposición, como ya se mencionó. Cuando hay varios literales se descompone.
- Resolución de problemas, heredada de la Reducción de Diferencias y el General Problem Solver
    - Incorporan operadores al plan que sean relevantes aunque no se puedan aplicar en el estado actual, lo que le permitirá razonar hacia delante o atrás.
- Utiliza la representación simbólica del conocimiento sobre el dominio del problema: STRIPS no trabajará directamente con el estado y las acciones, se tienen dos estructuras…
    - Utiliza una pila de objetivos: que queda por resolver, que ya se ha resuelto, que acciones se han resuelto.
    - El estado actual del problema.
- Es una especie de metaestado, donde se tiene una pila junto con un estado. La pila se utiliza para dar una descripción de cómo se está resolviendo el problema en cada momento y con esto se realizará la búsqueda. Estos son los nodos del proceso de búsqueda.
- Se realizará un proceso de búsqueda con una serie de operaciones, estas operaciones no son acciones como tal, sino son acciones que trabajan con la pila y el estado actual del problema.
- El nodo inicial contiene en la pila de objetivos el objetivo completo del problema, y el estado actual es estado inicial del problema.
- **Operadores**: (Transforman la pila y/o el estado actual)
    - **Emparejar**:
        - Si el objetivo (literales simples o compuestos) de la parte superior de la pila empareja con el estado actual, se suprime ese objetivo de la pila, o sea que es cierto en ese estado.
        - Puede suceder que sea válido con varias substituciones diferentes, se anota la sustitución asociada a la pila, entonces recordará la sustitución aplicada.
        - Este operador **solo toca la pila de objetivos.**



    - **Descomponer**:
        - Si el objetivo en la parte superior de la pila es compuesto que no es cierto (si lo fuera se aplicaría Emparejar), entonces añadir los literales componentes en la parte superior de la pila.
        - **Importante**: El literal compuesto ***no*** se suprime, porque es un dispositivo de seguridad que da STRIPS para resolver las posibles interacciones que no era capaz de resolver un grafo Y/O.
        - Pueden descomponerse en cualquier orden.



- **Resolver**:
  - Cuando el objetivo que se encuentra en la parte superior de la pila es un único literal no resuelto, entonces se busca un operador cuya lista de adición contenga un literal que empareje con él.
  - Dado el literal L, se aplica un criterio de buscar una regla que añada ese literal, que lo tenga en su lista de adición, se elimina L y se reemplaza por el operador O_u y se pone su fórmula de precondición FP_u encima, esto se hereda del GPS. Notar que se introducen ahora operadores a la pila.

- **Aplicar**:
  - Cuando el término de la parte superior de la pila es un operador, entonces se suprime de la pila, se aplica sobre el estado actual modificándolo y se anota en el plan. Se ha hecho razonamiento hacia delante.
  - **Modifica** la **pila** de objetivos y el **estado** actual.

- El proceso se repite hasta que la pila queda vacía, el estado debe ser concordante con el objetivo.

Proceso de búsqueda asociado

- Aunque no pareciera que se realiza una búsqueda, se debe de utilizar puesto que los operadores anteriores introducen bifurcaciones en las configuraciones de los nodos, por ejemplo, si un objetivo se empareja con 3 substituciones diferentes, se generan 3 nodos hijos de ese estado, donde en cada nodo se intentará probar con un tipo de sustitución. Esto sucede también con Descomponer (orden de los literales) o Resolver (que existen varios operadores que resuelven un literal).
- Aquí es necesario tener un algoritmo de búsqueda, como un A* o de Anchura. Originalmente se utilizó un Algoritmo en Profundidad Retroactivo con una heurística; el problema de la heurística es que no se tenía una que fuera muy potente o generalista.
  - No se pudieron obtener heurísticas buenas para la ordenación de literales; la heurística más relevante y que era genérica que se logró sería la selección de la mejor acción a aplicar.
  - La heurística para determinar el mejor operador es el número de precondiciones ciertas en el estado actual.

Interacción entre Subobjetivos

- Puede que cuando se realice una descomposición, puede que la resolución de un literal haga que otro literal que anteriormente era válido en el nodo ahora sea falso; resolviendo un objetivo cambia la resolución del otro.
- Por esto es que se mantiene el literal compuesto, si sucede una interacción, el algoritmo vuelve a

descomponerlo en el nuevo estado hasta que se logre tener uno donde se cumpla el literal compuesto. Esto no está garantizado, no es completo.
  - Las interacciones que son suaves son aquellas que con el orden de las acciones se puede evitar.
  - Las interacciones fuertes son aquellas que suceden sin importar el orden en el que se elijan las acciones.
- STRIPS tiene dificultad con las interacciones entre los subobjetivos, se llama la anomalía de **Sussman** y que el planificador no produce planes óptimos. El problema es que STRIPS descompone los problemas secuencialmente, se necesita intercalar las resoluciones de los subobjetivos para dar la optimalidad.

Planificación como búsqueda en un espacio de estados: HSP, FF:

- Existen otra clase de planificadores que lograron resolver el problema haciendo una búsqueda directamente en el espacio de estados.

HSP

- **HSP** (Heuristic Search Planner) fue el primero desarrollado que logró obtener heurísticas independientes del dominio, la idea es utilizar modelos simplificados; se trata de aplicar operadores e ignorar la lista de supresión; utiliza una estimación para el cálculo de la longitud del plan de la solución simplificada.
- Utilizó como algoritmo de búsqueda el algoritmo de escalada por máxima pendiente, donde los empates se resuelven de manera estocástica, una segunda versión utiliza el A* con peso.
  - Se define el peso de f en 0 para todos los literales f del estado inicial y al resto se le asigna valor infinito.
  - Se aplican las acciones y se actualizan los pesos.
  - Para cada acción con precondición $pre(O)$ que añada al literal f se tiene que
    - $peso(f) = \min\left(peso(f), peso(pre(O) + 1)\right)$
  - Se está realizando una especie de estimación de cuantos operadores se están requiriendo para obtener un literal concreto.
  - Cuando se tiene un conjunto de literales, se suman los pesos.
  - El proceso se repite hasta que no cambien los pesos en dos iteraciones sucesivas.
  - La heurística queda como $h_{HSP}(S) = peso(G) = \sum_{g \in G} peso(g)$ donde G es el conjunto de objetivos.
- Como se obtiene por un modelo simplificado, se pensaría que la heurística es admisible y óptima pero no lo es, porque está basada en un modelo simplificado pero la sumatoria del final es la que evita la admisibilidad, porque está suponiendo la independencia del peso en cada uno de los literales, porque no toma en cuenta que ciertos pasos para verificar un literal podrían también ayudar a verificar otro: los toma como independientes, no bastaría sumarlos.
- Aun así es una heurística muy buena.

FF

- **FF** (Fast-Foward) está basado en **HSP** pero añade tres elementos nuevos:
  - Usa una nueva heurística basada en la idea del planificador Graphplan, basado en grafos. Se utiliza una versión simplificada. En las capas se van añadiendo los literales resultantes de realizar ciertas acciones, incluida la acción de no hacer nada, la ejecución finaliza cuando en la última capa se alcanzan todos los objetivos.
    - Una vez que llega al destino se extrae el plan situándose en la última capa y para cada objetivo se hace lo siguiente en la capa i.
      - Si el objetivo está presente en la capa $i-1$, entonces se inserta como objetivo para alcanzar en la capa $i-1$.
      - En otro caso, se selecciona una acción de la capa $i-1$ que añade el objetivo e insertamos cada precondición de la acción como objetivo de la capa $i-1$.
      - Cuando ya se ha trabajado con todos los objetivos de la capa i, se continúa con los objetivos de la capa $i-1$.
      - Se para al llegar a la primera capa.
    - El plan simplificado es una secuencia $\{O_0, O_1, \dots, O_{m-1}\}$ en donde en cada O_i es el conjunto de acciones seleccionadas en cada capa.
    - La longitud del plan es $h_{FF}(S) = \sum_{i=0,\dots,m-1} |O_i|$. Es una mejor estimación de HSP.
  - Nuevo método de búsqueda: Un método de escalada forzado.
    - Este método de escalada evalúa todos los sucesores, se cambia irrevocablemente al mejor de

todos si mejora al actual, si no es el caso, se realiza una búsqueda en anchura con el objetivo de buscar un nodo mejor que el de partida.
        - La búsqueda continúa desde ese nodo.
        - Guarda la traza de los nodos seleccionados para generar el plan.
    - Un proceso para ordenar descendientes.
        - Considera en el proceso de búsqueda un conjunto de acciones útiles
        - Para un estado S, el conjunto $H(S)$ de acciones útiles se definen como
            - $H(S) = \{o | pre(o) \subseteq S, add(o) \cap G_1 \neq \emptyset\}$ donde G_1 representa el conjunto de objetivos marcados en la siguiente capa.
            - Es el conjunto de acciones que se pueden aplicar cuando las precondiciones son ciertas en el estado y que añaden alguno de los literales objetivo.

- Estos planificadores anteriores realizan la planificación como la búsqueda en un espacio de estados, pero no ha sido la única vía.

Planificación como búsqueda en un espacio de planes:

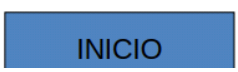
- La filosofía que tienen es que se tiene un espacio donde se tienen todos los planes posibles que podrían trazarse para una situación dada y uno se mueve en este espacio de planes, pasando de un plan a otro, se realiza una búsqueda hasta encontrar un plan que encaja con las condiciones del problema. Es una búsqueda por unas características muy específicas.
- Se obtiene la **Planificación por Orden Parcial (POP)**, que también surge inspirado en STRIPS. Retoma la idea de descomposición de STRIPS.
- Aún así, se tienen diferencias: No se toman decisiones hasta que no se sepa como tomarlas, a esto se le denomina la Hipótesis del Menor Compromiso, el problema de STRIPS y de usar una pila es que estaba forzado a ordenar las acciones y ejecutarlas ahí mismo aún si no supiera como hacerla, esto trae muchos problemas.
- **Hipótesis del Menor Compromiso:** Es necesario ocuparse de las decisiones que actualmente interesen, dejando cualquier otra decisión para más tarde.
- Se utiliza una estructura de grafo, pero no de búsqueda, sino para representar la solución con la misma idea de la pila de STRIPS: cambiar una estructura ordenada por una parcialmente ordenada para representar esa pila.

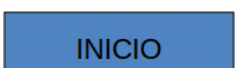
Definición de plan

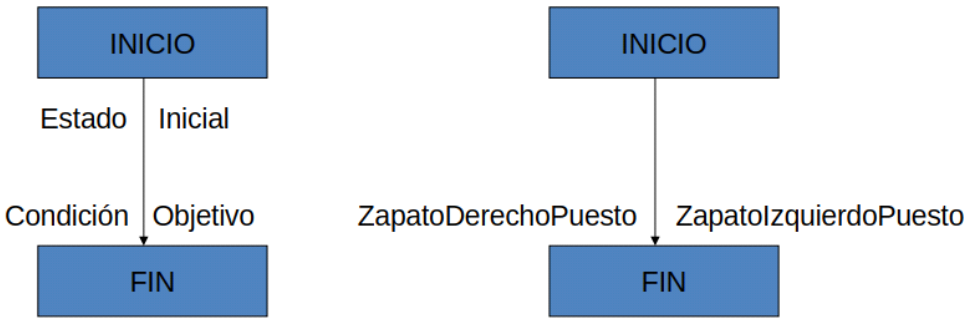
- El concepto de plan es el concepto base de POP.
- Tiene varias componentes
    - Un conjunto de nodos (operadores)
    - Un conjunto de relaciones de orden: $s_i < s_j$ significa que s_i como acción debe producirse en algún momento antes que s_j. Se tenía como la pila en STRIPS.
    - Un conjunto de restricciones de variables del tipo $x = A$.
    - Un conjunto de vínculos causales: $s_i \rightarrow^c s_j$ que significa que s_i aporta el literal c a s_j.
        - Estructura nueva e importante, se originan en los problemas de interacción. Esta estructura de datos que permite proteger la validez de los nodos.

Plan Inicial

- Se incluyen dos operadores (nodos) ficticios INICIO y FIN ya que solamente trata con planes no posee estados ni objetivos como tal por lo que se necesitan operadores para indicar el inicio y el final de un plan.
    - INICIO no tiene precondiciones y tiene de efecto el estado inicial.
    - FIN tiene como precondiciones el objetivo y no tiene efectos.
- Este plan inicial tiene solamente estos dos nodos, no hay ninguna clase de restricciones o vínculos, se tiene que el orden es INICIO $<$ FIN.
- Siempre se parte de este plan.





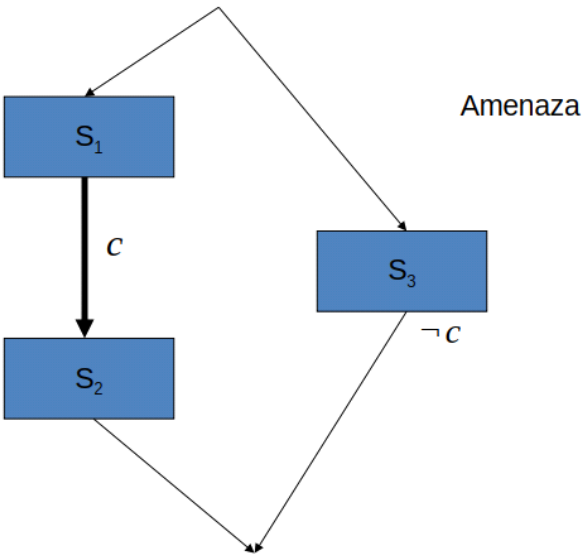


- La idea es que se va moviendo de un plan a otro por refinamiento de las acciones, añadiendo acciones, se realiza buscando alguna acción que añadir al plan hasta que queda resuelto. Se ordenan las acciones lo que tiene explicitadas el orden, el resto se deja en paralelo, STRIPS en cambio habría obtenido diferentes planes secuenciales para poderlo representar.

Plan Completo y Consistente

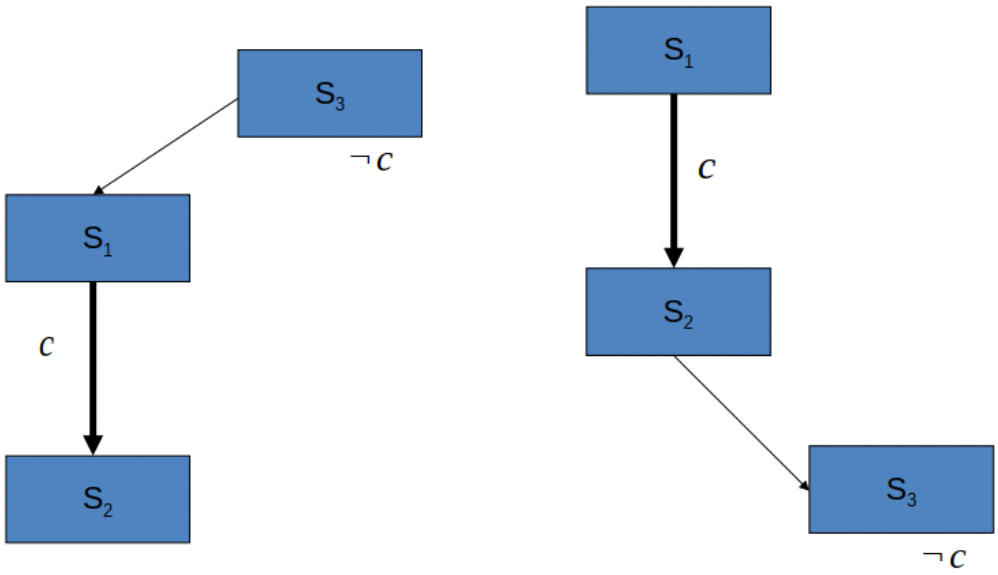
- El algoritmo parte del plan inicial y lo refina hasta que se tiene un plan completo y consistente.
- **Plan Completo**
  - Cada precondición de cada operador se debe de satisfacer mediante otro operador, tiene que ver con los vínculos causales.
  - s_i logra una precondición c de s_j si
    - $s_i < s_j$ y $c \in EFECTOS(s_i)$
    - $\neg\exists s_k$ con $\neg c \in EFECTOS(s_k)$ y $s_i < s_k < s_j$ en alguna linealización del plan.
    - No puede haber una acción s_k que elimine c entre s_i y s_j, porque entonces c ya no sería cierta cuando s_j la requiera, este es el mecanismo de protección.
- **Plan Consistente**
  - No hay contradicciones en las restricciones de orden ni de variables.
    - $x = A$ y $x = B$ sería inconsistente.
    - $s_i < s_j, s_j < s_k, s_k < s_i$ sería inconsistente.

- **Amenazas**: Cuando se viola un vínculo causal se denomina de esta manera



  - ¿Cómo se resuelven? Dos métodos.
    - **Ascenso**: se traslada la acción que produce la amenaza antes del vínculo causal.
    - **Degradación**: se traslada la acción que lo produce después del vínculo causal.

- Se prueba una u otra para ver cual funciona, puede que no se pueda reparar la amenaza y el algoritmo hace una vuelta atrás.



Algoritmo POP(<A,O,L>),agenda,R)

1. **Terminación**: Si **agenda** esta vacía, devuelve <A,O,L>.
2. **Selección del objetivo**: Sea <Q,A_e>un par de la **agenda** (por definición $A_e \in A$ y Q es un elemento de la conjunción de la precondición de A_e).
3. **Selección de la acción**: Sea A_{nuevo} = **elección** de una acción que añada Q (ya sea mediante una nueva acción de R o mediante una acción existente en A que pueda ordenarse consistentemente antes de A_e). Si no existiese tal acción entonces devolver fallo. En otro caso, sea $L' = L \cup \{A_{nuevo} \xrightarrow{Q} A_e\}$, y sea $O' = O \cup \{A_{nuevo} < A_e\}$. Si A_{nuevo} es una acción nueva, entonces $A' = A \cup \{A_{nuevo}\}$ y $\{A_0 < A_{nuevo} < A_\infty\}$ (en otro caso A'=A).
4. **Actualizar el conjunto de objetivos**: Sea **agendanueva**=**agenda**-$\{< Q, A_e >\}$. Si A_{nuevo} no estaba antes, entonces para cada elemento Q_i de la conjunción de su precondición añadir $< Q_i, A_{nuevo} >$ a **agendanueva**.
5. **Protección de enlaces causales**: Para acción A_t que pudiese amenazar a un enlace causal $A_p \xrightarrow{H} A_c \in L'$ elegir una restricción de orden consistente, o bien

   a) **ascenso**: añadir $A_t < A_p$ a O', o

   b) **degradación**: añadir $A_c < A_t$ a O'.

   Si ninguna restricción es consistente, entonces devolver fallo.
6. **Recursión**: Llamar a POP(<A',O',L'>),agendanueva,R).

Heurísticas para la Planificación de Orden Parcial

- Se tienen diferentes heurísticas para la selección de planes
  - Número de acciones en el plan N, sería la g.
  - Precondiciones sin resolver OP, sería la h. Es una estimación de las acciones que faltan.
- Frecuentemente se utiliza el algoritmo A* con la heurística $f(P) = N(P) + OP(P)$.
- No tiene garantía absoluta que sea admisible porque una misma acción puede resolver dos precondiciones, luego la h sobreestima lo que falta por resolver, aún así es una heurística muy buena.

Planificación Jerárquica

- A menudo es interesante generar un plan inicial compuesto por tareas de alto nivel para luego ir desglosándolo en acciones más simples.
- Esto es especialmente útil en problemas complejos en donde planificar desde cero todas las acciones de bajo nivel puede resultar una tarea muy costosa.
- Existen dos vías generales para esta planificación:
  - **ABSTRIPS**, que es una extensión directa de STRIPS. Se jerarquizaba el problema de STRIPS con la asignación de valores numéricos de criticidad sobre los predicados de las precondiciones. Considera el problema a nivel de criticidad máximo, cuando se tiene que insertar una acción aparecen las de nivel más alto solamente, luego se reduce la criticidad y se comienza de nuevo con el plan que dejó el nivel anterior utilizando acciones del nivel mayor hasta la del nivel actual, se va refinando el plan.
  - **Redes de Planificación Jerárquica de Tareas**: Hace uso intensivo del conocimiento y requiere que el mismo sea descrito de forma jerárquica, es decir, requiere un trabajo más arduo al principio pero los planificadores son mucho más rápidos y eficientes.
    - Se tienen dos tipos de acciones:
      - □ Métodos (Alto nivel), describen acciones con primitivas.
      - □ Acciones Primitivas (Bajo nivel)
    - Los métodos se pueden descomponer en subtareas que pueden ser otros métodos o acciones primitivas.
    - Esta descomposición se define mediante una red de tareas, que establece la ordenación total o parcial de las subtareas de los métodos.
    - El funcionamiento de un planificador HTN es muy similar al de un planificador clásico
    - El objetivo del problema consiste en aplicar una serie de tareas mediante descomposición y planificación de una secuencia de métodos que, finalmente, se desglosará en las acciones primitivas.
    - Presenta dos ventajas:
      - □ El dominio del problema se describe en términos de acciones estructuradas jerárquicamente, resultando más intuitivo para el experto que modela el problema.
      - □ La función de planificador consiste en refinar estas estructuras generando las expansiones necesarias y simplifica la resolución del problema original.
    - Esto es un grafo Y/O por debajo que resuelve estas tareas.

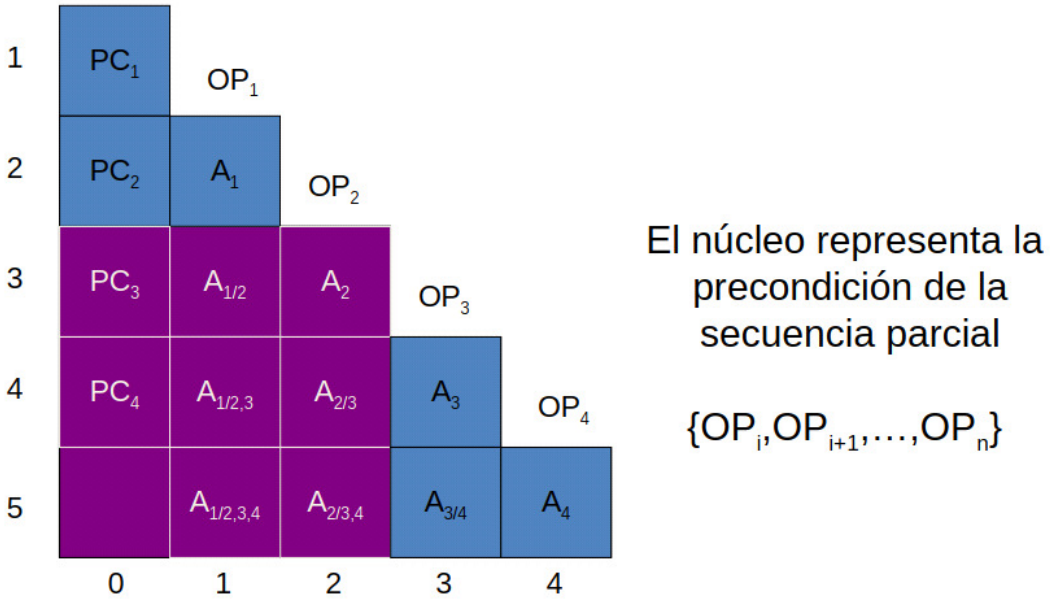
<u>Representación para Planes:</u>

- Se generó al mismo tiempo que surge STRIPS
- El objetivo es conseguir una representación para planes que permita:
  - Usar un plan previamente generado para la resolución de problemas posteriores.
  - Controlar inteligentemente la ejecución de un plan concreto.
- Se basa en que los humanos interiorizan un plan y ya posteriormente tienen internamente una versión general del plan, se entienden como macrooperaciones.
- Se utiliza una vez que se ha generado un plan, permite monitorizarlo y en ocasiones permite reparar un plan si sucede algún inconveniente.

Tablas Triangulares

- Dado un plan que ya se haya obtenido se empieza a rellenar la tabla, la representación se describe con una estructura de datos llamada una tabla triangular inferior de $n + 1$ operaciones del plan con una numeración

especial de filas y columnas.
    - Las columnas van desde 0 hasta n de izquierda a derecha.
    - En las filas es de 1 hasta $n + 1$ de arriba abajo.
- Luego de esto, se etiqueta la tabla con los operadores encima del número que le corresponde al operador, ahora se entiende el porqué de la numeración: el operador OP_1 está asociado a la fila 1 y la columna 1, esta propiedad se verifica para el resto de operadores.
- Ahora se rellena el interior:
    - A_1 refleja la lista de Adición del OP_1, las listas que siguen debajo tienen que ver con esa lista de adición pero ahora teniendo en cuenta el resto de los operadores, o sea, A_i es la lista de adición de OP_i
    - $A_{i/j}$ son los literales de A_i no suprimidos por OP_j.
- El significado de una fila menos en la posición 0, son los literales que van añadiendo una lista de operadores.
- Se tendrá toda la tabla menos la columna 0 rellena.
    - Está relacionada con las precondiciones de los operadores del plan, un operador se puede aplicar en un estado si sus precondiciones son ciertas y se podrá aplicar en un estado futuro porque o bien hay otro operador que añade cosas que lo necesita, están relacionados los operadores; o porque el literal que se necesita es cierto en el estado inicial y no ha cambiado a lo largo del tiempo.
- En la fila quinta debe estar el objetivo que se busca.
- Núcleo i-ésimo: Es una fórmula de la lógica y no está perfectamente definido, es una conjunción de literales de los literales marcados que se encuentran en una región de la tabla: es la intersección de los literales de las filas que quedan por debajo de la i-ésima incluida esta y con las columna que quedan a la izquierda de la i-ésima no incluida.



- Permite comprobar si el plan va a fallar o no, el núcleo permite tener una garantía que el plan va a llegar al objetivo. Si no hay ningún núcleo válido, se tiene que replanificar. Si una operación se hace falsa, se busca en los núcleos alguna que se vuelva cierta y se toma desde allí.
- La estructura relevante es el núcleo, permite aprender plantes y para ejecutarlos inteligentemente se puede ver la columna primera o la última fila.