HTN - Hierarchical Task Network

Introducción a los algoritmos de planeamiento

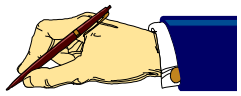
2018

Agenda

- Repaso de algunos conceptos
  - Suposiciones Básicas
  - Búsqueda en el espacio de estados vs. espacio de planes
- Action-based Planning
  - UCPOP
  - Tópicos Avanzados
  - Limitaciones de UCPOP
- Knowledge-based Planning
  - Incorporando el conocimiento disponible
  - HTN Planning (SHOP)
  - Mixed-initiative planning
- Planning con información incompleta
- Algunas aplicaciones reales

Problema de planning

- Formalmente un problema de planning involucra 3 entradas:
  - Una descripción del mundo (estado inicial)
  - Una descripción de los objetivos a alcanzar (estado final)
  - Una descripción de las posibles acciones (Operadores)
- Salida Esperada
  - Una secuencia de acciones que, si es ejecutada empezando desde el estado inicial, permitirá alcanzar los objetivos.
- Dos clases de algoritmos implementan esta idea.
  - Action-based planning (UCPOP, GraphPlan)
  - Knowledge-based planning (SHOP, O-Plan)

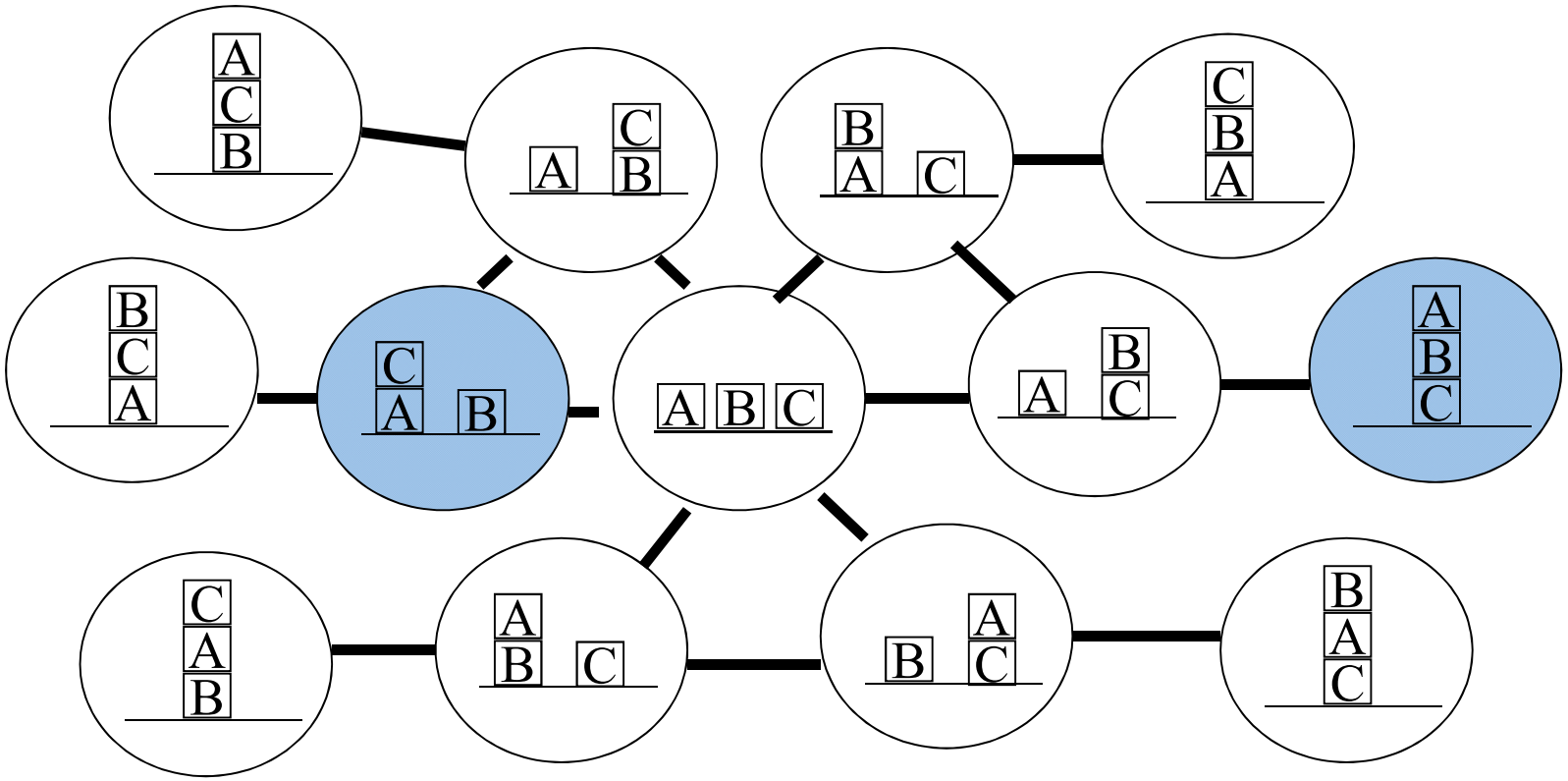


Suposiciones Básicas

< *EstadoInicial, ConjuntoDeObjetivos, ConjuntoDeAcciones* >

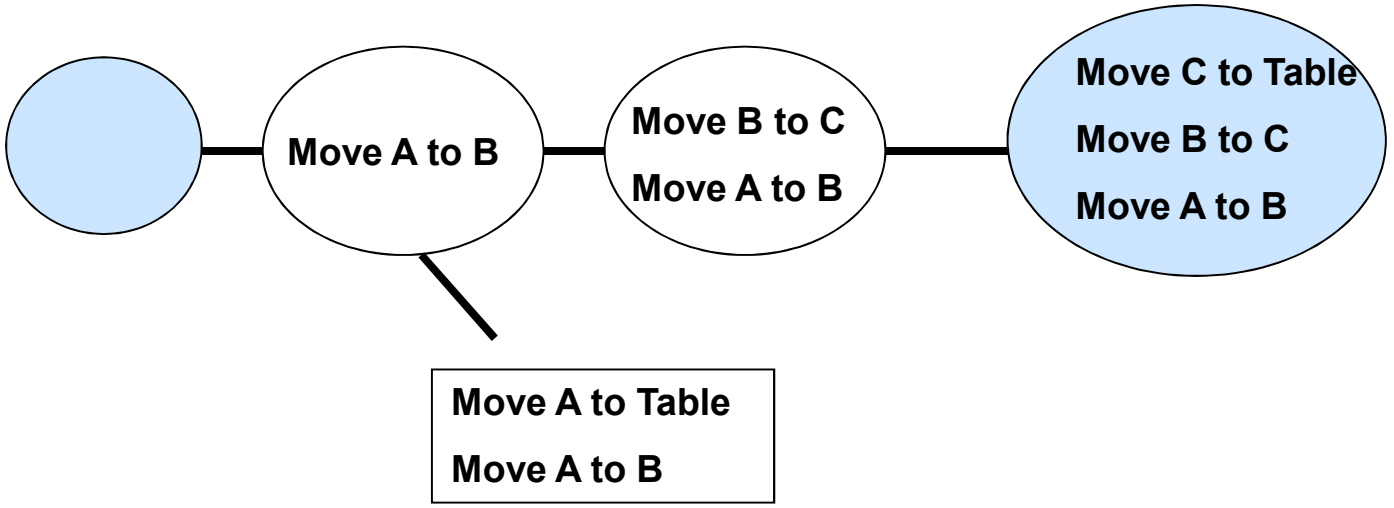
- Tiempo atómico (Ejecución de las acciones)
- Efectos determinísticos (Acción x Estado → Estado)
- Total conocimiento del mundo
- Las acciones son la única causa de cambio

Búsqueda en el espacio de estados



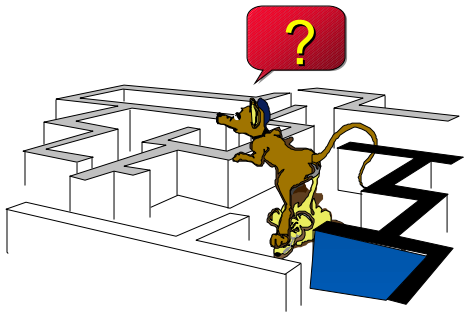
Búsqueda en el espacio de planes

- Los nodos representan planes parcialmente especificados
- Los arcos representan operaciones de refinamiento
  - Una nueva acción es agregada al plan actual
- Principio del menor compromiso



Perspectivas en el Algoritmo UCPOP

- Extensión de POP (más expresividad)
  - Variables
  - Precondiciones disyuntivas
  - Cuantificación Universal
  - Efectos Condicionales



- Tópicos Avanzados
  - Cuantificación sobre universos dinámicos
  - Eficiencia Computacional
  - Cálculos Matemáticos
  - Información Incompleta

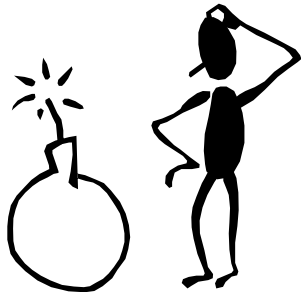
UCPOP: Cuantificación Universal (Estática)

move(B,L,M) :
Preconditions: *notEqual(M,L), briefcase(B), at(B,L)*
Effects: *at(B,M), not(at(B,L),*
forAll(object(X), when((in(X,B)), (at(X,M), not(at(X,L))))

- Se asume cuantificación “estática”
  - El mundo es estático y finito
  - Los objetos son asociados a tipos
  - Las acciones no pueden agregar nueva información de tipos
- Estrategia: los predicados cuantificados son transformados a cláusulas ground (base universal)

UCPOP: Cuantificación en Universos dinámicos

- Qué sucede con los dominios en los cuales los efectos de las acciones pueden crear o eliminar objetos?
  - Como se debería representar (sintácticamente) la creación/destrucción en el lenguaje de la acción
  - Como el planner debería manejar objetivos cuantificados universalmente cuando es posible que sucedan estos efectos



UCPOP: Modelando la destrucción de Objetos

- Una acción con un efecto que niega un predicado de tipo de objeto

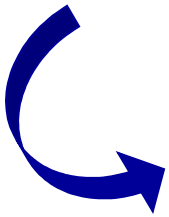
anyaction(...) : ...

Effects: ... not (book(constitution)) // Destrucción de un libro

- books = {bible, constitution} forAll(book(X), in(X,library))

and (in(bible, library), in(constitution, library))

Base Universal



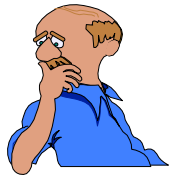
and (
or (in(bible, library), not book(bible))
or (in(constitution, library), not book(constitution))
)

UCPOP: Modelando la creación de Objetos

- Similarmente, un efecto puede agregar un nuevo objeto para un tipo dado

anyaction(...) : ...

Effects: ... book(b234) // Creación de un libro



- Cuando se expande la base universal para una precondición de una acción A
  - Se deben considerar todos los objetos que posiblemente sean creados por todas las acciones que posiblemente estén ordenadas antes que A
  - Esto es suficiente?

Aritmética

- Razonando con números
  - Tiempo, recursos compartidos, recursos con una especifica capacidad, etc.
  - En algunos casos se pueden invocar rutinas embebidas

Ejemplo: Conseguir suficientes soldados o equipamiento para una operación militar dada

Knowledge-based Control Rules

- Agregar reglas de control para guiar la búsqueda no determinística
  - Uso de una heurística de búsqueda (mejor performance)
  - Puede se que no escalen bien para problemas complejos
- Como adquirir este conocimiento?
  - Análisis del dominio
  - Técnicas de Machine learning
  - Case-based planning

Planning con información incompleta

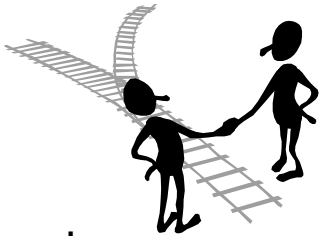
- Se relaja la asunción de mundo cerrado
  - Es casi imposible tener información completa en dominios del mundo real
  - Información incompleta, pero **correcta**
- Dos tipos de acciones
  - Acciones que cambian el estado del mundo (causal actions)
  - Acciones que recolectan información (sensing actions)

anyaction(...) :
Preconditions: *not (computerAlive(?computer)),*
getAnswer('Reset?', ?yesno)
...

- Interacción con personas

Intercalando planning con ejecución

- Cual es el correcto balance entre planning y ejecución?
  - Planear para todas las posibles contingencias puede involucrar mucho trabajo extra
  - Intercalar planeamiento con ejecución fuerza al planner a ajustarse a información que cambia dinámicamente
  - Si una acción es ejecutada, el planner debería ser capaz de retroceder sobre una decisión a ejecutar
- Criterio para ejecutar una acción dada
  - Sus precondiciones deben ser satisfechas
  - Debe estar ordenada antes de todas las otras acciones no ejecutadas del plan
  - La acción no debe estar involucrada en alguna amenaza
  - La acción física que se ejecuta podría ser ambigua



HTN

Hierarchical Task Network

Hierarchical Task Network Planning

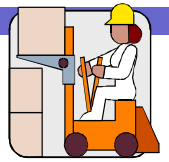
- Ideas Básicas
  - Los planes complejos a menudo tienen una estructura identificable
  - Esta estructura puede ser capturada en forma de jerarquías de sub. planes abstractos
  - Los sub. planes son frecuentemente (casi) independientes unos de otros
  - Búsqueda de reducción del problema, más que una búsqueda en el espacio de estados

Si tiene suficiente dinero para el precio: tomar un taxi para ?y
parar un taxi
entrar al taxi
decir "quiero ir a ?y"
esperar hasta ?y
pagar la tarifa
entonces salir del taxi

Para ir a la conferencia en ?x
ir al aeropuerto
tomar un avión a ?x
entonces ir al hotel de la conf.

Para ir al aeropuerto
Elegir entre conducir o tomar un taxi

HTN Planning: Ejemplo de fabricación (1)



buildProduct(?spec, ?product, ?factory)

Preconditions & constraints

prepareRawProduct(?raw-spec, ?warehouse, ?plant)

setupMachine(?machine, ?plant)

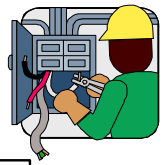
Other tasks …

turnOnCarrier(?carrier)

moveRawProduct(?raw-product, ?from, ?to)

turnOffCarrier(?carrier)

activateHeater(?heater)



initializeFilter(?filter)

Preconditions & constraints

getProduct(?raw, ?product)

getProductWithMethod1(?raw, ?product)

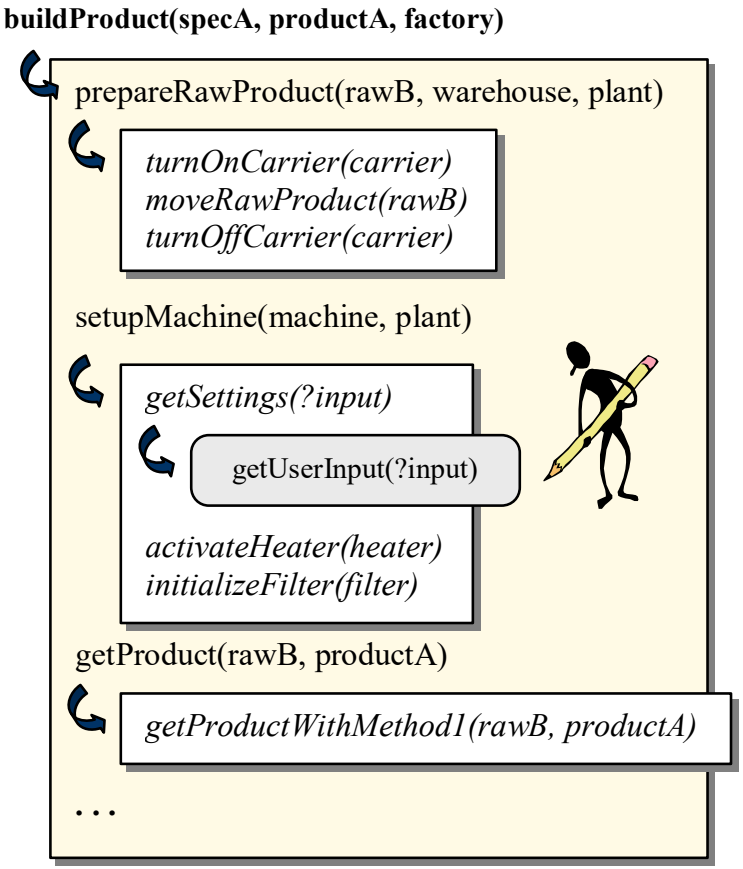
getProductWihtMethod2(?raw, ?product)

…

…

- Ensamblar diferentes sub. productos en base a una especificación de producto dada.

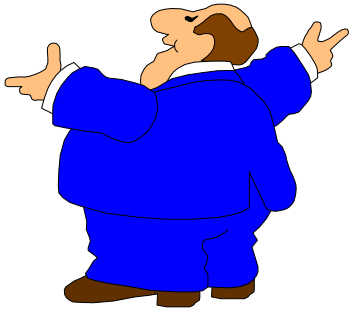
HTN Planning: Ejemplo de fabricación (2)



- La solución modela el mundo en el mismo sentido que lo hacen los humanos
  - Mecanismos similares de abstracción
  - Jerarquía de niveles de abstracción
  - Tácticas, Estrategias
  - Algunos pasos pueden requerir la intervención del usuario

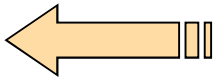
HTN Planning: Planes abstractos

- Operadores abstractos descomponen en sub. planes parciales
- Operadores Primitivos
  - Acciones estándar tipo STRIPS
  - “Ejecutables”
- Operadores Compuestos
  - Precondiciones y efectos
  - Métodos para descomponer operadores en sub. planes más detallados
- Un plan Abstracto contiene operadores Compuestos
- Un plan Completamente instanciado tiene solo operadores primitivos



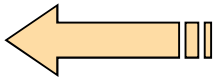
HTN Planning: Métodos

- Los métodos son usados para descomponer operadores
  - Nombre y Variables
  - Precondiciones
  - Efectos



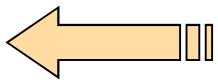
Parte que describe qué hace el método (igual que los operadores primitivos)

  - Expansión
  - Ordenes Temporales
  - Links Causales



Parte que detalla como el operador es descompuesto

  - Ventanas temporales
  - Utilización de recursos



Parte que describe aspectos cuantitativos del sub. plan

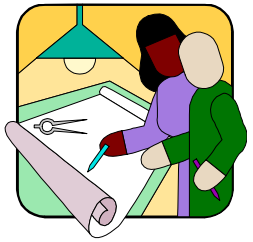
Ejemplo: Construcción de una casa

Method: build(?house, ?land, ?money)

Preconditions: own(?land), have(?money), single-family(?house)

Effects: built(?house,?land)

Expansion:
- S1: buildFoundation(?house)
- S2: buildFrame(?house)
- S3: buildRoof(?house)
- S4: buildWalls(?house)
- S5: buildInterior(?house?)
- S6: decorate(?house)



Ordering: S1<S2, S2<S3, S2<S4, S3<S5, S5<S6

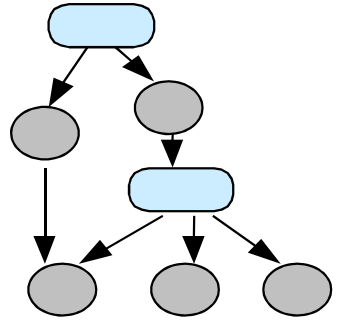
Links:
- S1 causes (foundations laid) for S2
- S2 causes (frame built) for S3 and S4
- S3 causes (roof built) for S5
- S4 causes (walls built) for S5
- S5 causes (interior done) for S6

TimeWindow: start between 11:30 and 14:30 at S3

Resources: bricklayers = between 1 and 2 men at S4

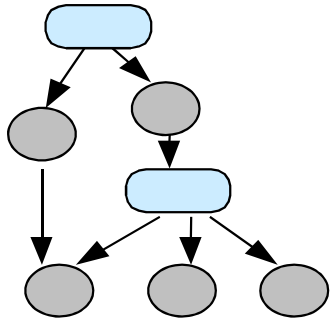
El Algoritmo SHOP

- Simple Hierarchical Order Planner (SHOP)
- Algoritmo HTN básico
  - Empezar con las tareas iniciales de alto nivel (**no son exactamente los objetivos**)
  - Crear redes de tareas mediante la expansión repetitiva de sub. planes hasta que el plan es instanciado completamente
  - Elegir métodos cuyas condiciones de aplicabilidad se cumplen



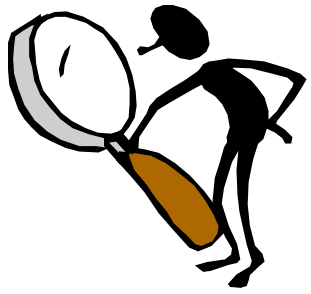
El Algoritmo SHOP

- Algoritmo SHOP
  - Planner lineal de búsqueda hacia adelante
    - Planea en el mismo orden que la ejecución
    - Esencialmente una búsqueda en profundidad (DFS)
  - No hay acciones concurrentes
  - Representación de operadores altamente expresivos
  - Algoritmo de planning eficiente (pero inflexible)



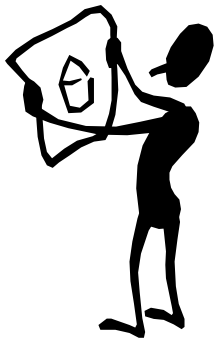
JShop Axiomas y Operadores

```
#axiom: caminarDistancia( ?u, ?v ) ->
    elClimaEs( bueno ) , distancia( ?u, ?v, ?w ) , #call: @menor( ?w, 3)
#end-axiom.

#operator: irRapido( ?vehiculo, ?a, ?b ) ->
    #pre: [ at( ?a ), at( ?vehiculo, ?a ), rapido(?vehiculo) ]
    #add: [at( ?b ), at( ?vehiculo, ?b ) ]
    #delete: [ at( ?a ), at( ?vehiculo, ?a ) ]
#end-operator.

#operator: caminar( ?here, ?there ) ->
    #pre: [ at( ?here ) ]
    #add: [ at( ?there ) ]
    #delete: [ at( ?here ) ]
#end-operator.
```



JShop Métodos

```
#method: viajarA( ?destino ) ->
    #pre: [ at( ?lugar ), caminarDistancia( ?lugar, ?destino ) ]
    #body: [ caminar( ?lugar, ?destino ) ]
#end-method.
```



```
#method: viajarA( ?destino ) ->
    #pre: [ at( ?lugar ), atTaxi( ?taxi, ?lugar ),
    distancia( ?lugar, ?destino, ?d ), tenerDineroTaxi( ?d ) ]
    #body: [ pedir( ?taxi, ?lugar ), irRapido( ?taxi, ?lugar, ?destino ),
    #call: @plus( ?d, 1, ?suma ), pagarConductor( ?suma ) ]
#end-method.
```

JShop Estado Inicial y objetivos

#state: [

distancia(parque, centro, 2), distancia(centro, periferia, 8), atTaxi(taxi1, parque), rutaColectivo(bus2, centro, periferia), at(centro), elClimaEs(bueno), dinero(12), ...].

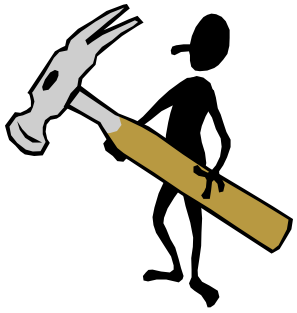
#goals: [viajarA(parque)].

Solución:

1- caminar(centro ,parque)

2- pedir(taxi1, parque), irRapido(taxi1,parque,centro), pagarConductor(3)

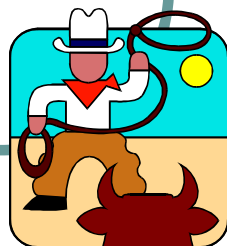
Extensiones al HTN Simple



- Detección de Amenazas
  - Tratar con objetivos que interactúan
  - Buscar formas de re usar operadores
- Los métodos pueden incluir efectos
  - Descubrir peligros en el proceso de planning de manera temprana
- Interacción con el usuario
  - Para proveer ciertos valores como entradas
  - Para decidir entre alternativas
- Los métodos pueden indicar el consumo de recursos
  - Realizar algún tipo de Scheduling

HTN Planning: Detalles

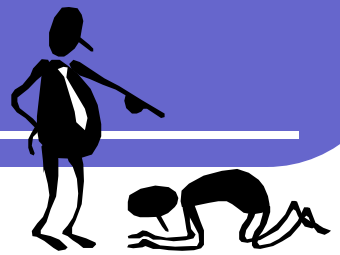
- Los métodos codifican conocimiento del dominio
- Los métodos codifican conocimiento de resolución del problema
- Las abstracciones encapsulan patrones de interacción
- En teoría HTN planning es más expresivo que planning basado en acciones
  - Es análogo a gramáticas libres del contexto



HTN Planning: Detalles

- Advertencias
  - HTN planning, en el peor de los casos es NP Completo
  - Puede no terminar (expansión de métodos recursiva – puede ser difícil de detectar los ciclos infinitos)
  - Es posible tener que esperar hasta que el plan este completamente expandido antes de encontrar que el plan es ilegal.

Advisable Planners



- Dirigen o influencian el proceso de planning
- Advice
  - Restricciones especificas sobre la solución deseada y las decisiones de refinamiento del proceso de planning
  - Orden de los objetivos, selección de los operadores a ser aplicados, instanciación de ciertas variables, etc.
- Diferentes tipos de advice
  - Task-specific advice
  - Evaluational advice
  - Strategic advice (roles y acividades)

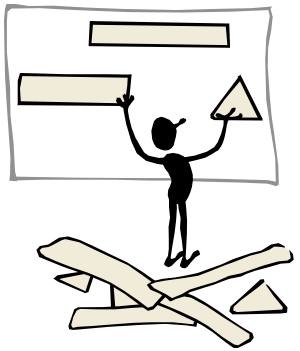
Hacer T1 antes que T2 e Incluir T5 en el plan

Para en hoteles 3 estrella mientras vacacionamos en Tandil

Gastar menos que $800 en todas las comodidades

Descomposición basada en casos en HTN Planning

- Combina CBR con planning
- Teoria del dominio incompleta
  - Guidelines y procedimientos operativos standards
  - Experiencias
  - Ej. Planeamiento de operaciones militares



```
#method: selectTransport( ?base, ?site ) ->
    #pre: [ vehicleAvailable( ?helicopter, ?base) ]
    #question-answer-pairs:
            weather conditions? rainy
            inminent danger to evacuees? no
    #body: [ transport(?helicopter, ?base, ?site) ]
#end-method.
```

Un caso puede ser visto como un método HTN normal más un numero de preferencias (Pares Pregunta – Respuesta)

Algunos Detalles

Las aproximaciones a planning pueden ser mas o menos basadas en el conocimiento, dependiendo del dominio de la aplicación

- Action-based planning es una aproximación utilizada para resolver ciertos aspectos de un problema
  - Tipo de sub. Problemas en una gran pintura
- Si planes necesitan ser utilizados para guiar conductas en entornos dinamicos complejos.
  - El conocimiento debe ser incorporado, codificado en estructuras entendibles
  - Conocimiento del dominio permite escalar los algoritmos a aplicaciones al mundo real.
  - Bayesian networks y CBR pueden ser alternativas interesantes para explorar.

Referencias

- “A Call for Knowledge-based Planning” D. Wilkins and M. desJardins. AI Magazine 2001
- “Shop and MShop: Planning with Ordered Task Decomposition” D. Nau, Y. Cao, A. Lotem and H. Muñoz-Avila. TR 2000