Geração de Planos (Planning)

May 26, 2019

AI Planning

- Estratégias de busca conseguem produzir sequências de ações que resultam num estado final
- Porém as representações de estados são atômicas (simples e primitivas)
- Portanto, requerem heurísticas específicas ao domínio para encontrar a solução de forma eficiente
- É necessário ter uma solução que tire partido da **estrutura** do problema
  - Utilização de linguagens baseadas em lógica de primeira ordem para representar a estrutura do problema

AI Planning

Algoritmo básico para *planning*

function SIMPLE-PLANNING-AGENT(*percept*) **returns** an *action*
  static: *KB*, a knowledge base (includes action descriptions)
    p, a plan, initially *NoPlan*
    t, a counter, initially 0, indicating time
  local variables: *G*, a goal
    current, a current state description

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  current ← STATE-DESCRIPTION(*KB*, *t*)
  if *p* = *NoPlan* **then**
    G ← ASK(*KB*, MAKE-GOAL-QUERY(*t*))
    p ← IDEAL-PLANNER(*current*, *G*, *KB*)
  if *p* = *NoPlan* **or** *p* is empty **then** *action* ← *NoOp*
  else
    action ← FIRST(*p*)
    p ← REST(*p*)
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  t ← *t* + 1
  return *action*

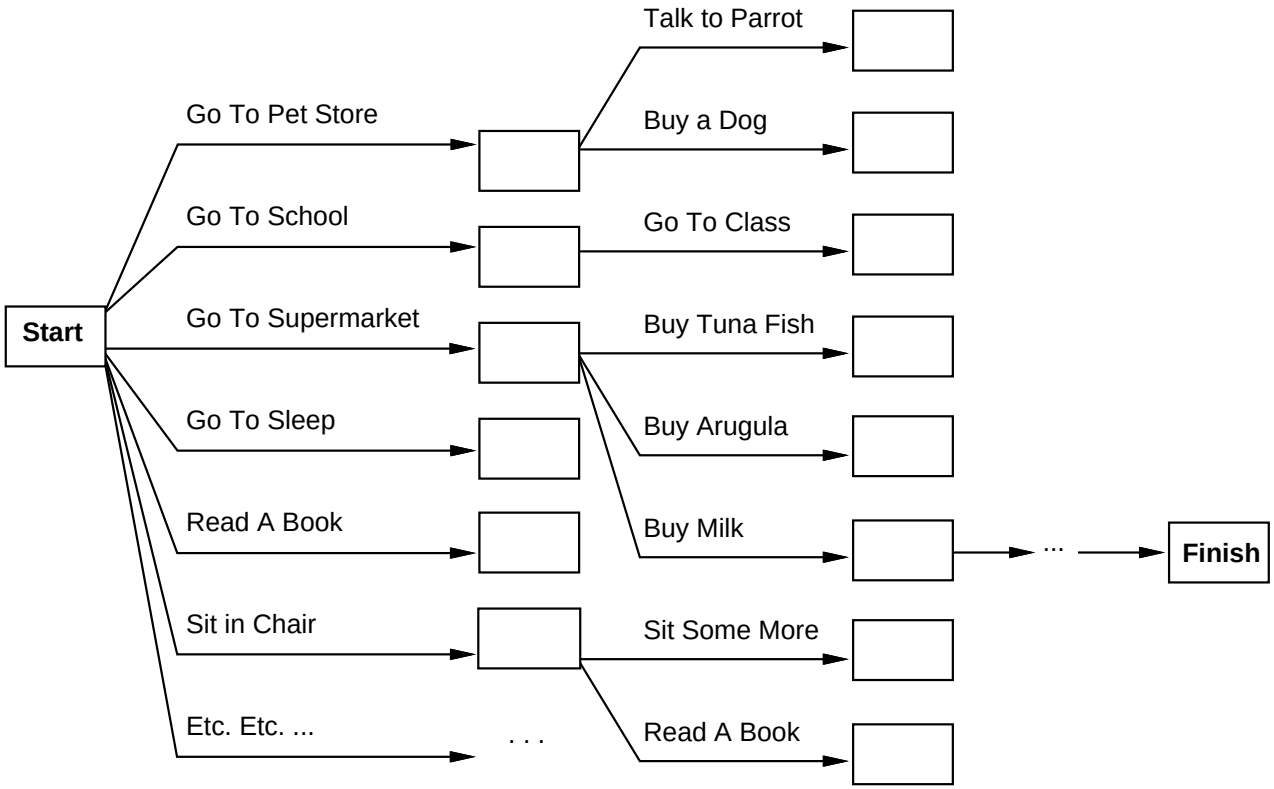
AI Planning

- PDDL: Planning Domain Definition Language

AI Planning

- diferenças: representações de ações, estados, objetivos e planos.
- exemplo simples: comprar 1 l de leite, uma dúzia de bananas e um barbequim.
- problem solving:
  - estado inicial: agente está em casa, mas sem todas as coisas necessárias.
  - operadores: tudo que o agente pode fazer para obter os objetos.
  - opcional: heurística (ex, número de coisas ainda não compradas).

AI Planning


Start
Go To Pet Store
Go To School
Go To Supermarket
Go To Sleep
Read A Book
Sit in Chair
Etc. Etc. ...
. . .
Talk to Parrot
Buy a Dog
Go To Class
Buy Tuna Fish
Buy Arugula
Buy Milk
Sit Some More
Read A Book
...
Finish

AI Planning

- três idéias principais: transparência de ações e estados, liberdade para adicionar ações ao plano de ações a qualquer momento (sem ser incremental), e independência das partes.
- Planning utilizando cálculo de situações:
  - **estado inicial**: $At(Home, S_0) \wedge \neg Have(Milk, S_0) \wedge \neg Have(Bananas, S_0) \wedge \neg Have(Drill)$
  - **estado final**: $\exists s\ At(Home, s) \wedge Have(Milk, s) \wedge Have(Bananas, s) \wedge Have(Drill, s)$
  - **operadores**: ex, axioma do estado sucessor para comprar leite: $\forall a, s\ Have(Milk, Result(a, s)) \Leftrightarrow [(a = Buy(Milk) \wedge At(Supermarket, s)) \vee (Have(Milk, s) \wedge a \neq Drop(Milk))]$

AI Planning

- Em "planning" representamos *sequências de ações*, não somente ações únicas. Em cálculo de situações representamos Result:
  - $\forall s\ Result'([], s) = s$
  - $\forall a, p, s\ Result'([a \mid p], s) = Result'(p, Result(a, s))$
- Uma solução para o problema das compras é um **plano** p que qdo aplicado ao estado inicial S_0 produz a solução:
  - $At(Home, Result'(p, S_0)) \wedge Have(Milk, Result'(p, S_0)) \wedge Have(Bananas, Result'(p, S_0)) \wedge Have(Drill, Result'(p, S_0)$
- ASK daria uma solução do tipo:
  $p = [Go(Supermarket), Buy(Milk), Buy(Bananas), Go(HardwareStore), Buy(Drill), Go(Home)]$

AI Planning

- soluções teóricas nem sempre são *eficientes* se usarmos regras de inferência não guiadas.
- para soluções práticas:
  - Restringir a linguagem de definição do problema.
  - Utilizar algoritmos de "planning" particulares (*planner* para cada problema, invés de algoritmos gerais para provas de teoremas).

AI Planning

- Maioria dos "planners" utilizados descreve estados e operadores em STRIPS (linguagem baseada em cálculo de situações) ou extensões.
- Estados: representados por conjunções de literais "ground" (completamente instanciados), sem funções (predicados aplicados a símbolos constantes). Ex: $At(Home) \wedge \neg Have(Milk) \wedge \neg Have(Bananas) \wedge \neg Have(Drill) \wedge \ldots$
- Descrição do estado não precisa ser completa.
- Maior parte dos "planners" assume que se nada é dito sobre um literal, ele é falso (negação por falha em programação lógica).

AI Planning

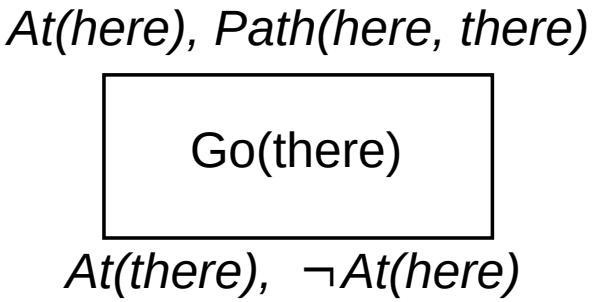
- Objetivos: tb representados por conjunções de literais. Ex: $At(Home) \wedge Have(Milk) \wedge Have(Bananas) \wedge Have(Drill)$.
- Podem conter variáveis. Ex, estar numa loja que venda leite: $At(x) \wedge Sells(x, Milk)$.
- Como em provadores automáticos de teoremas, assume-se que todas as variáveis do objetivo estão quantificadas existencialmente.
- Diferenças entre objetivos num "planner" e num provador de teoremas: objetivo no "planner" pergunta por uma sequência de ações que permitam validar o objetivo se este for executado. Objetivo no provador de teoremas é diretamente provado verdadeiro.

AI Planning

- Representação de ações: operadores STRIPS consistem de três componentes:
  - **Descrição da ação**: para o "planner" serve apenas como um nome para a ação.
  - **Precondição**: conjunção de átomos (literais positivos) que devem ser verdadeiros antes do operador ser aplicado.
  - **Efeito (ou poscondição)**: conjunção de literais (positivos ou negativos) que descrevem como a situação é modificada quando o operador é aplicado.

AI Planning

- Ex: $Op(ACAO : Go(there), PRECON : At(here) \wedge Path(here, there), EFEITO : At(there) \wedge \neg At(here)$



AI Planning

- Um operador pode conter variáveis: *esquema de operador*.
- Corresponde a uma família de ações.
- Somente operadores completamente instanciados podem ser executados.
- Linguagem de descrição de precondições e efeitos bastante restrita. Pode ser menos restrita.

AI Planning

- Operador é dito *aplicável* num estado s, se há alguma forma de instanciar variáveis do operador tal que a precondição seja verdadeira para esta instância. No estado resultante, todos os literais positivos da poscondição também devem ser verdadeiros, exceto os negados.
- Ex: situação inicial, $At(Home), Path(Home, Supermarket) \ldots$, ação Go(Supermarket) é aplicável, e a situação resultante contém $\neg At(Home), At(Supermarket)$, $Path(Home, Supermarket) \ldots$

AI Planning

- Dada uma definição do problema em STRIPS, algoritmos de busca podem ser utilizados para resolver o problema de "planning" do estado inicial para o objetivo.
- Chamados geradores de planos sobre o *espaço de situações*, e *progressivos*.
- Problema com esta abordagem: fator de ramificação!
- Alternativa: gerador de planos *regressivo*.
- Algoritmo original utilizado por STRIPS era de um gerador de planos sobre o espaço de situações regressivo, mas era incompleto, pois não sabia lidar com o fato de gerar uma conjunção.
- Ineficiente produzir algoritmo completo utilizando esta abordagem de geração de planos por situações.

AI Planning

- Alternativa: busca pelo *espaço de planos*.
- Começamos com um plano simples, incompleto: *plano parcial*.
- Aplicamos os operadores (de plano) e geramos outros planos até chegar a um que nos leve à solução.
- Possíveis operadores de plano: adicionar um novo passo ao plano, imposição de ordem aos passos do plano, instanciação de uma variável etc.
- Solução é o plano final. Caminho para chegar a este plano é irrelevante (eficiência...)

AI Planning

- Operações sobre planos:
  - **Operadores de refinamento**: adicionam restrições a um plano parcial. Eliminam alguns planos do conjunto possível de planos.
  - **Operadores de modificação**: qualquer outro operador que não seja de refinamento.
- Ex: Plano para calçar um par de sapatos.
- Objetivo: $ColocadoSapatoPeDireito \wedge ColocadoSapatoPeEsquerdo.$
- Estado inicial: não necessário.

AI Planning

- Operadores:
  - $Op(ACAO : ColocarSapatoPeDireito, PRECOND : ColocadaMeiaPeDireito, EFEITO : ColocadoSapatoPeDireito)$
  - $Op(ACAO : ColocarMeiaPeDireito, EFEITO : ColocadaMeiaPeDireito)$
  - $Op(ACAO : ColocarSapatoPeEsquerdo, PRECOND : ColocadaMeiaPeEsquerdo, EFEITO : ColocadoSapatoPeEsquerdo)$
  - $Op(ACAO : ColocarMeiaPeEsquerdo, EFEITO : ColocadaMeiaPeEsquerdo)$

AI Planning

- Plano parcial deve ter dois passos: ColocarSapatoPeDireito, ColocarSapatoPeEsquerdo. Quem vem primeiro?
- *Menos "comprometedor"*: somente escolher planos com informações relevantes no momento. Deixar outras escolhas para serem feitas mais tarde.
- *Plano de ordem parcial*: alguns passos do plano são ordenados, outros não.
- *Plano de ordem total*: todos os passos são ordenados.
- *Linearização* de planos: geração de um plano de ordem total após adição de restrições ao plano anterior.
- Restrições no valor das variáveis a serem instanciadas.

AI Planning

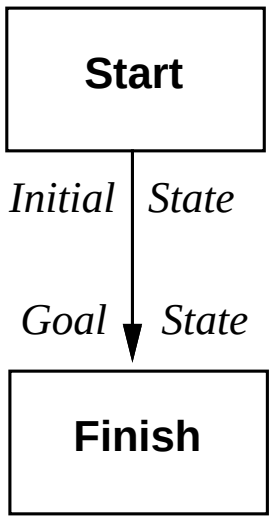
- Um plano é formalmente definido como uma estrutura de dados contendo os seguintes 4 componentes:
  - **conjunto de passos**. Cada passo é um dos operadores do problema.
  - **conjunto de restrições de ordenação** dos passos.
  - **conjunto de instâncias de variáveis** para cada passo.
  - **conjunto de ligações causais**. Ex: $S_i \longrightarrow^c S_j$, com c sendo precondição de S_j.
- Plano inicial descreve o problema ainda não resolvido. Consiste de dois passos: início e fim com relação de precedência Inicio $\prec$ Fim.

AI Planning

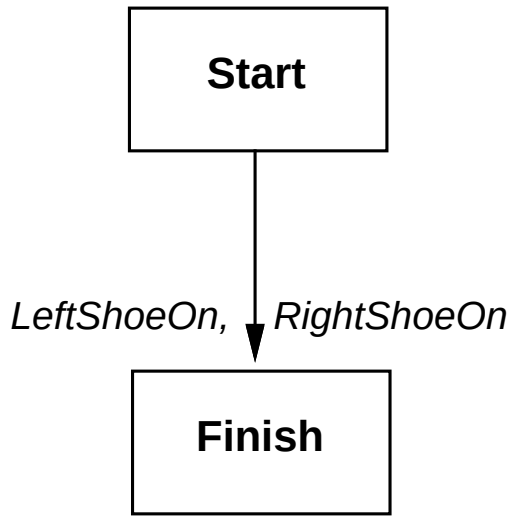
- Passo Início não tem precondições e o efeito é adicionar todas as proposições verdadeiras ao estado inicial.
- Passo Fim tem o objetivo como precondição, e nenhum efeito.
- Ex:

Plan(STEPS:{ S_1: Op(ACAO:Inicio),
S_2: Op(ACAO: Fim,
PRECOND: $ColocadoSapatoPeDireito \wedge$
$ColocadoSapatoPeEsquerdo$)},
ORDERINGS: { $S_1 \prec S_2$ },
BINDINGS: {},
LINKS: {})

AI Planning



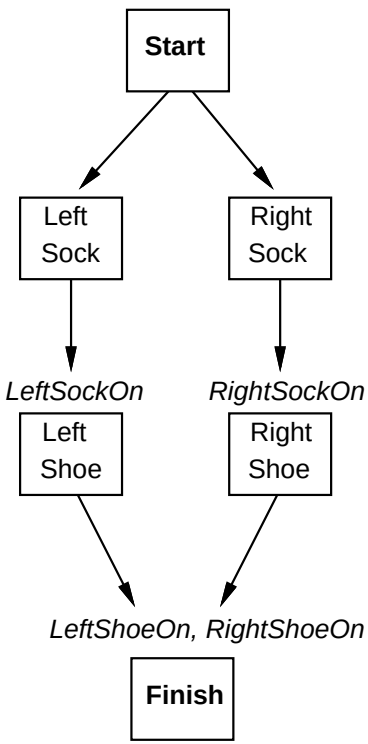
(a)



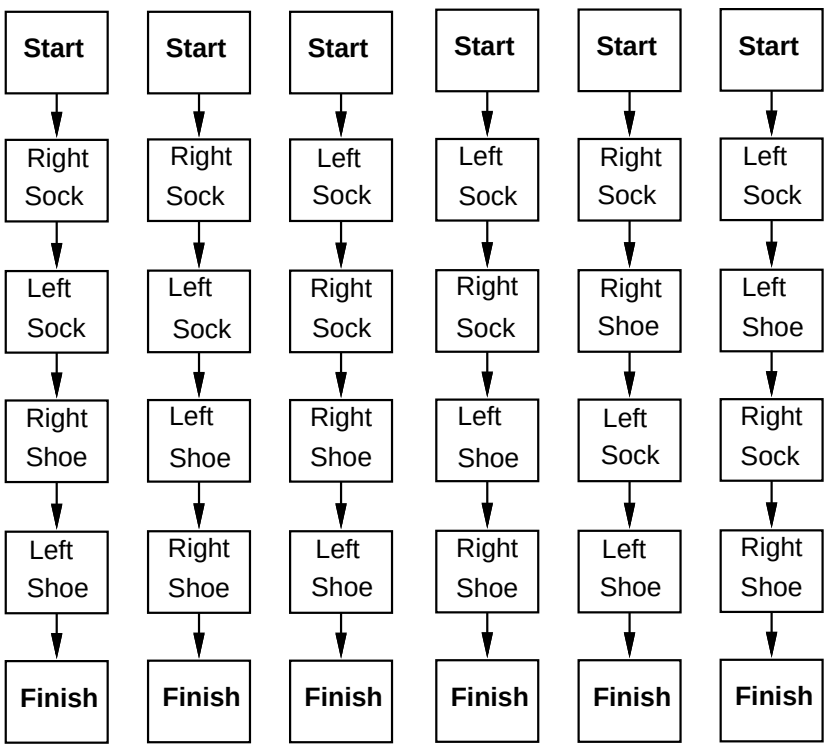
(b)

AI Planning

Partial Order Plan:



Total Order Plans:



AI Planning

Soluções:

- uma solução é um plano que um agente possa executar, e que garanta chegar ao objetivo.
- para garantir que um plano é uma solução, basta garantir que o plano é totalmente ordenado e instanciado.
- na prática, não razoável usar esta abordagem.
  - 1) mais natural gerar um plano de ordem parcial (mais geral).
  - 2) alguns agentes podem executar em paralelo.
  - 3) mais flexível para combinar com outros planos.

AI Planning

- Solução deve ser **consistente** e **completa**.
- Um plano é completo se toda pré-condição de cada passo for satisfeita por algum outro passo.
- mais formalmente: um passo S_i atinge uma pré-condição c do passo S_j se:
  - 1) $S_i \prec S_j$ e $c \in EFEITOS(S_i)$
  - 2) não há nenhum passo S_k tal que $(\neg c) \in EFEITOS(S_k)$, onde $S_i \prec S_k \prec S_j$ em alguma linearização do plano.

AI Planning

- Um plano é consistente se não tiver contradições na ordem ou valores das restrições.
- Contradição ocorre quando $S_i \prec S_j$ aparece ao mesmo tempo que $S_j \prec S_i$ no plano gerado ou quando $v = A$ e $v = B$.
- Como $\prec$ e $=$ são operadores relacionais com propriedade transitiva, teremos contradição se $S_1 \prec S_2$, $S_2 \prec S_3$, e $S_3 \prec S_1$.

AI Planning

Exemplo de Planning com Ordem Parcial

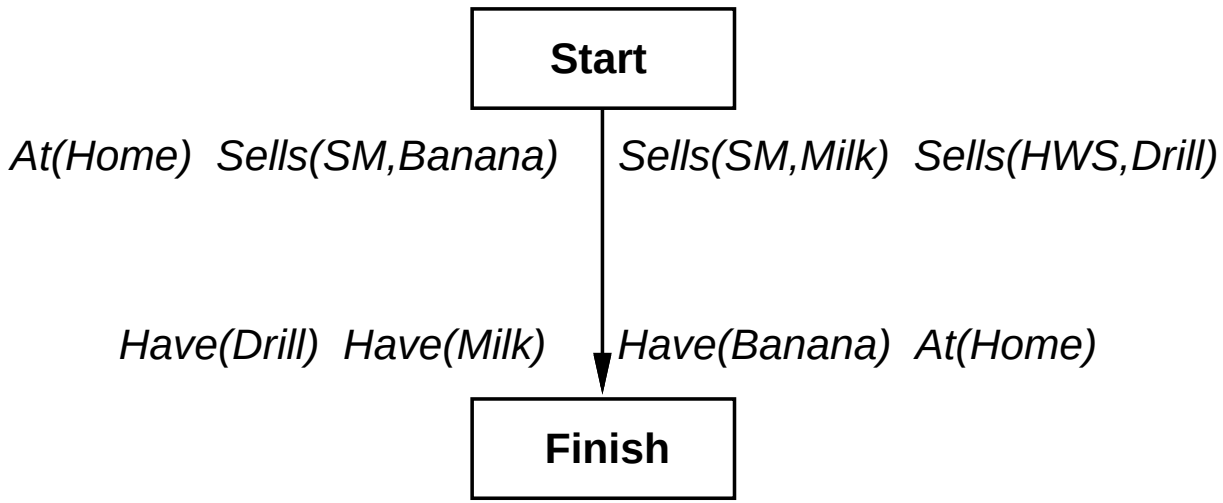
- Planner regressivo usando ordenação parcial, buscando soluções no espaço de planos.
- Começa com um plano inicial representado pelos passos inicial e final.
- Adiciona passos a cada iteração.
- Se o passo for inconsistente, retrocede (*backtracks*) e tenta outro ramo no espaço de busca.
- Para manter a busca guiada, o planner somente considera adicionar passos cuja pré-condição ainda não tenha sido alcançada (links causais são utilizados neste caso).

AI Planning

Exemplo de Planning com Ordem Parcial

Op(ACAO: Inicia,
EFEITO: At(Home) $\wedge$ Sells(HWS,Drill) $\wedge$
Sells(SM,Milk) $\wedge$ Sells(SM,Bananas))
Op(ACAO: Termina,
PRECOND: Have(Drill) $\wedge$ Have(Milk) $\wedge$
Have(Bananas) $\wedge$ At(Home))
Op(ACAO: Go(there),
PRECOND: At(here)
EFEITO: At(there) $\wedge \neg$ At(here))
Op(ACAO: Buy(x),
PRECOND: At(shop) $\wedge$ Sells(shop,x)
EFEITO: Have(x)

AI Planning

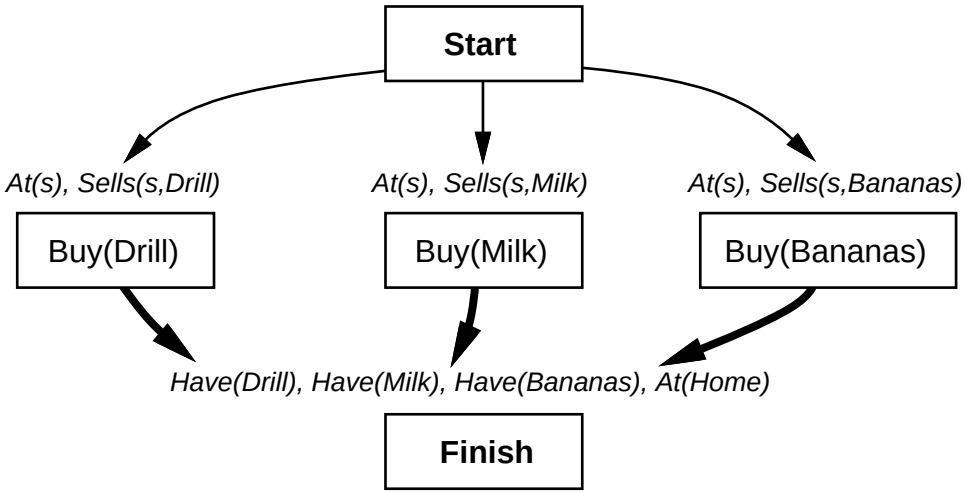

Start
At(Home)
Sells(SM,Banana)
Sells(SM,Milk)
Sells(HWS,Drill)
Have(Drill)
Have(Milk)
Have(Banana)
At(Home)
Finish

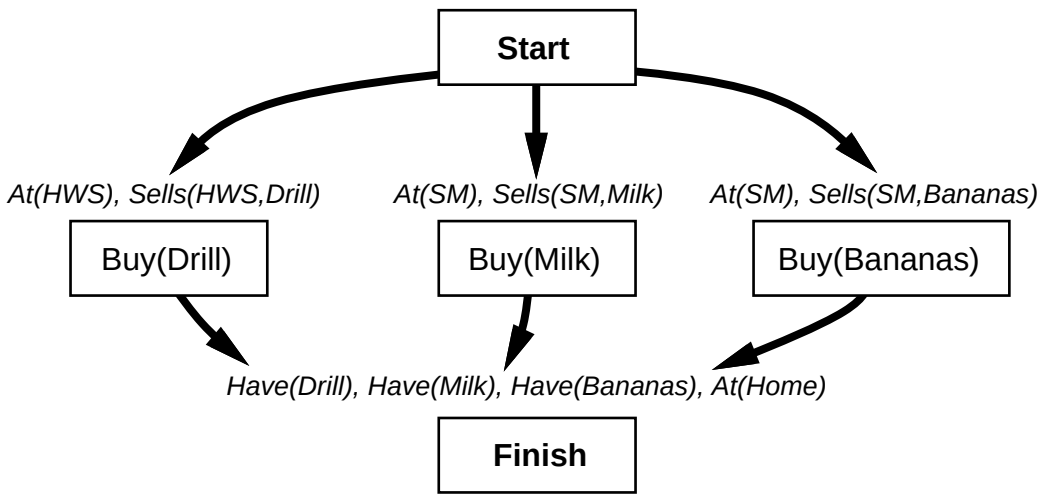
AI Planning

Exemplo de Planning com Ordem Parcial

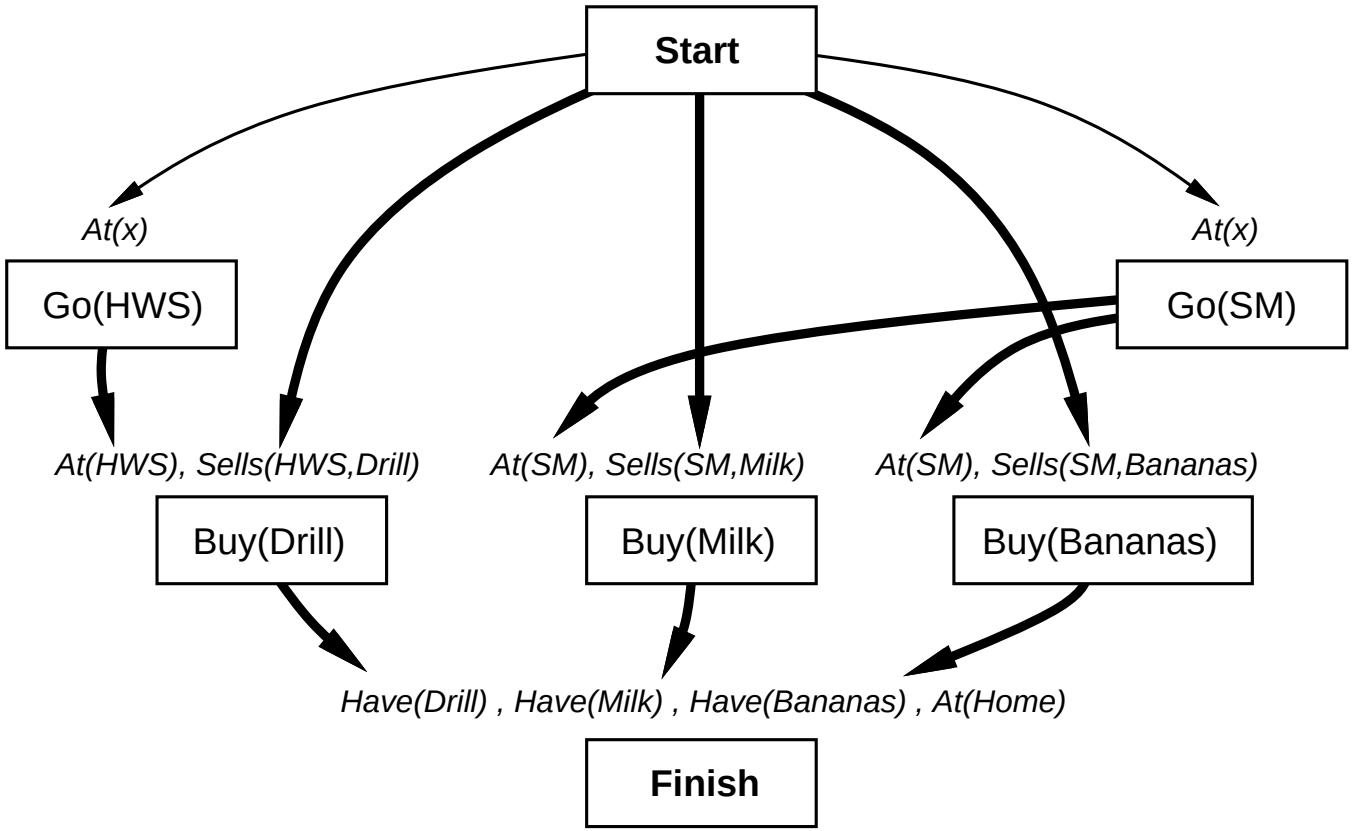
- Várias maneiras de gerar planos para este problema.
- Algumas levam a caminhos que não são soluções.
- Planner escolhe comprar somente aquelas coisas que interessam, dos lugares que interessam, sem gastar muito tempo.
- **Links protegidos**, um link causal é protegido garantindo que 'ameaças', isto é, passos que poderiam 'deletar' uma condição protegida, são ordenados de forma a aparecer antes ou depois do link protegido: **demotion** ou **promotion**.

AI Planning

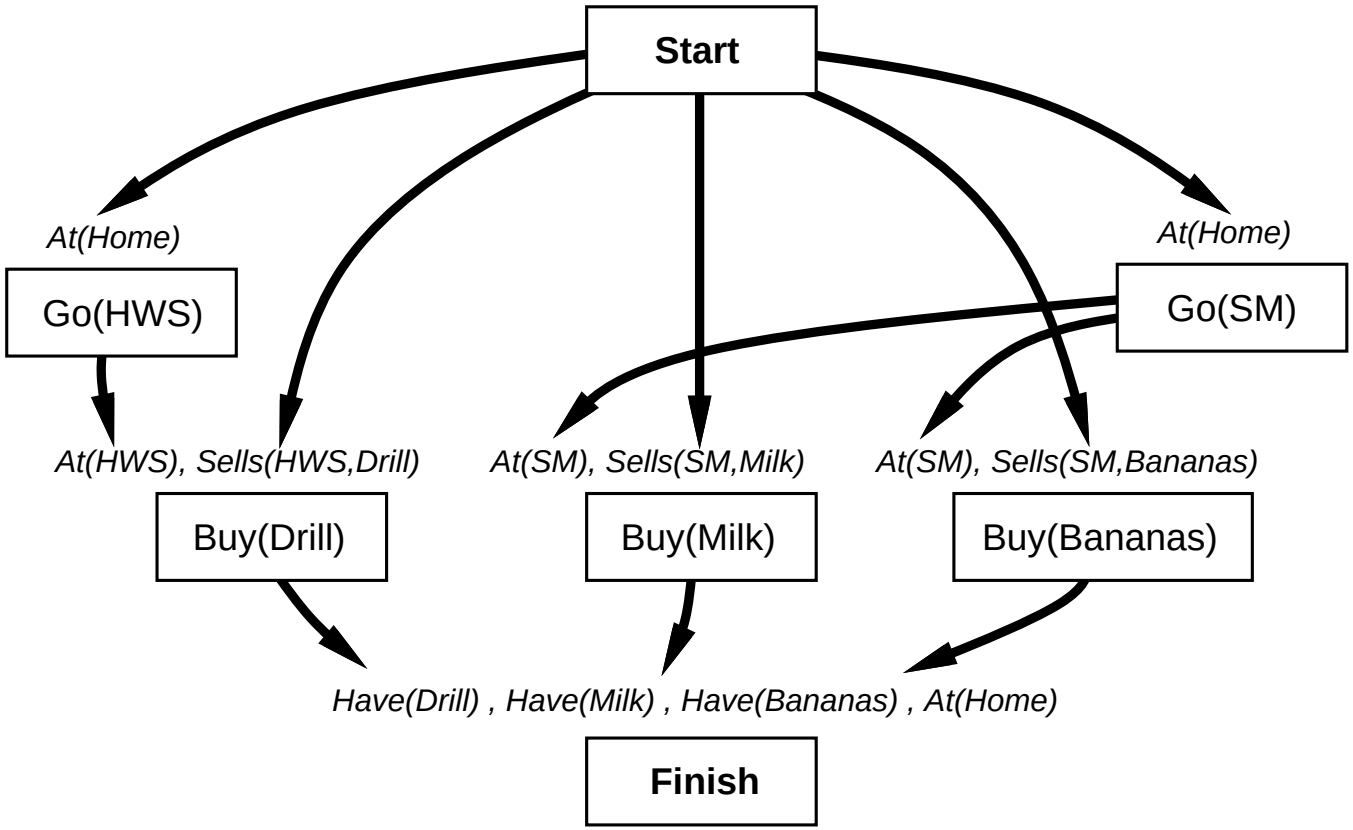

Start
At(s), Sells(s,Drill)
At(s), Sells(s,Milk)
At(s), Sells(s,Bananas)
Buy(Drill)
Buy(Milk)
Buy(Bananas)
Have(Drill), Have(Milk), Have(Bananas), At(Home)
Finish


Start
At(HWS), Sells(HWS,Drill)
At(SM), Sells(SM,Milk)
At(SM), Sells(SM,Bananas)
Buy(Drill)
Buy(Milk)
Buy(Bananas)
Have(Drill), Have(Milk), Have(Bananas), At(Home)
Finish

AI Planning

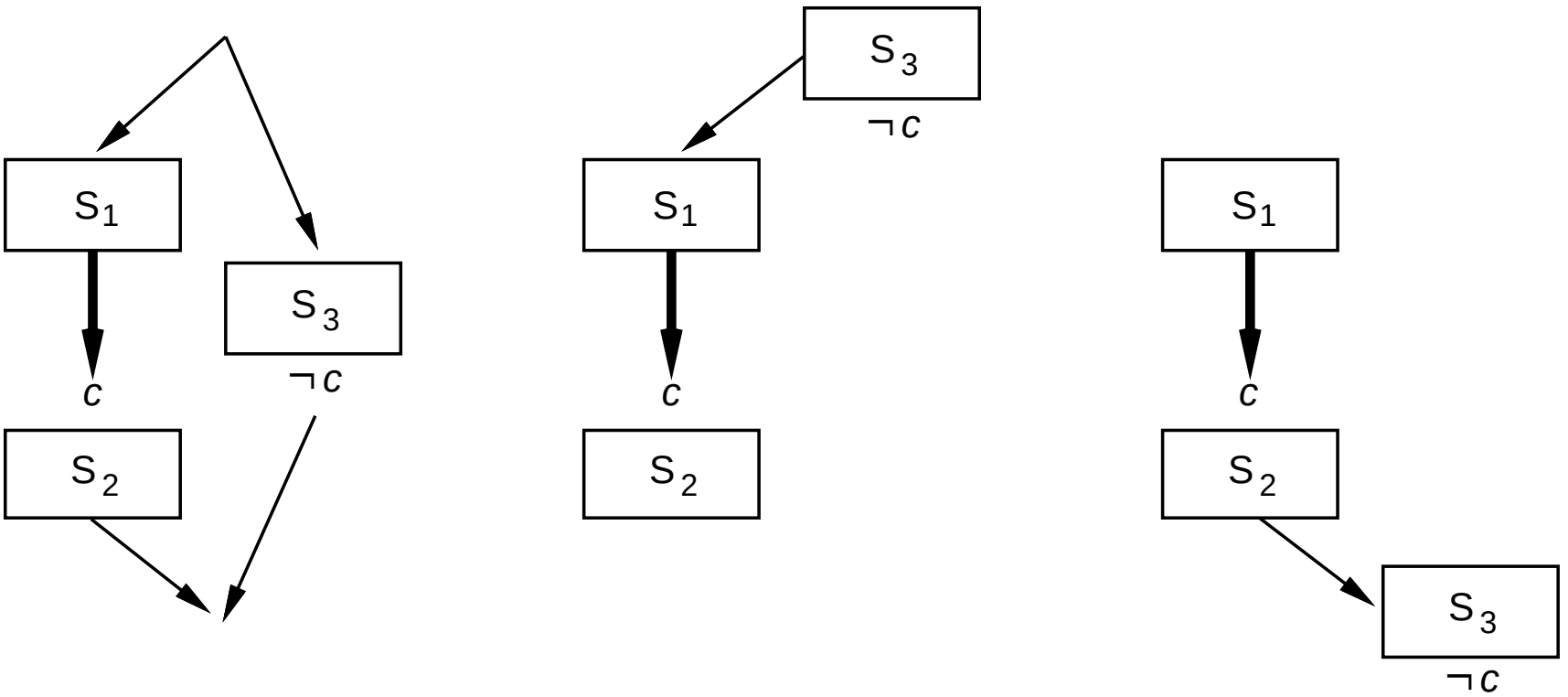

Start
At(x)
Go(HWS)
At(x)
Go(SM)
At(HWS), Sells(HWS,Drill)
Buy(Drill)
At(SM), Sells(SM,Milk)
Buy(Milk)
At(SM), Sells(SM,Bananas)
Buy(Bananas)
Have(Drill) , Have(Milk) , Have(Bananas) , At(Home)
Finish

AI Planning


Start
At(Home)
Go(HWS)
At(Home)
Go(SM)
At(HWS), Sells(HWS,Drill)
Buy(Drill)
At(SM), Sells(SM,Milk)
Buy(Milk)
At(SM), Sells(SM,Bananas)
Buy(Bananas)
Have(Drill) , Have(Milk) , Have(Bananas) , At(Home)
Finish

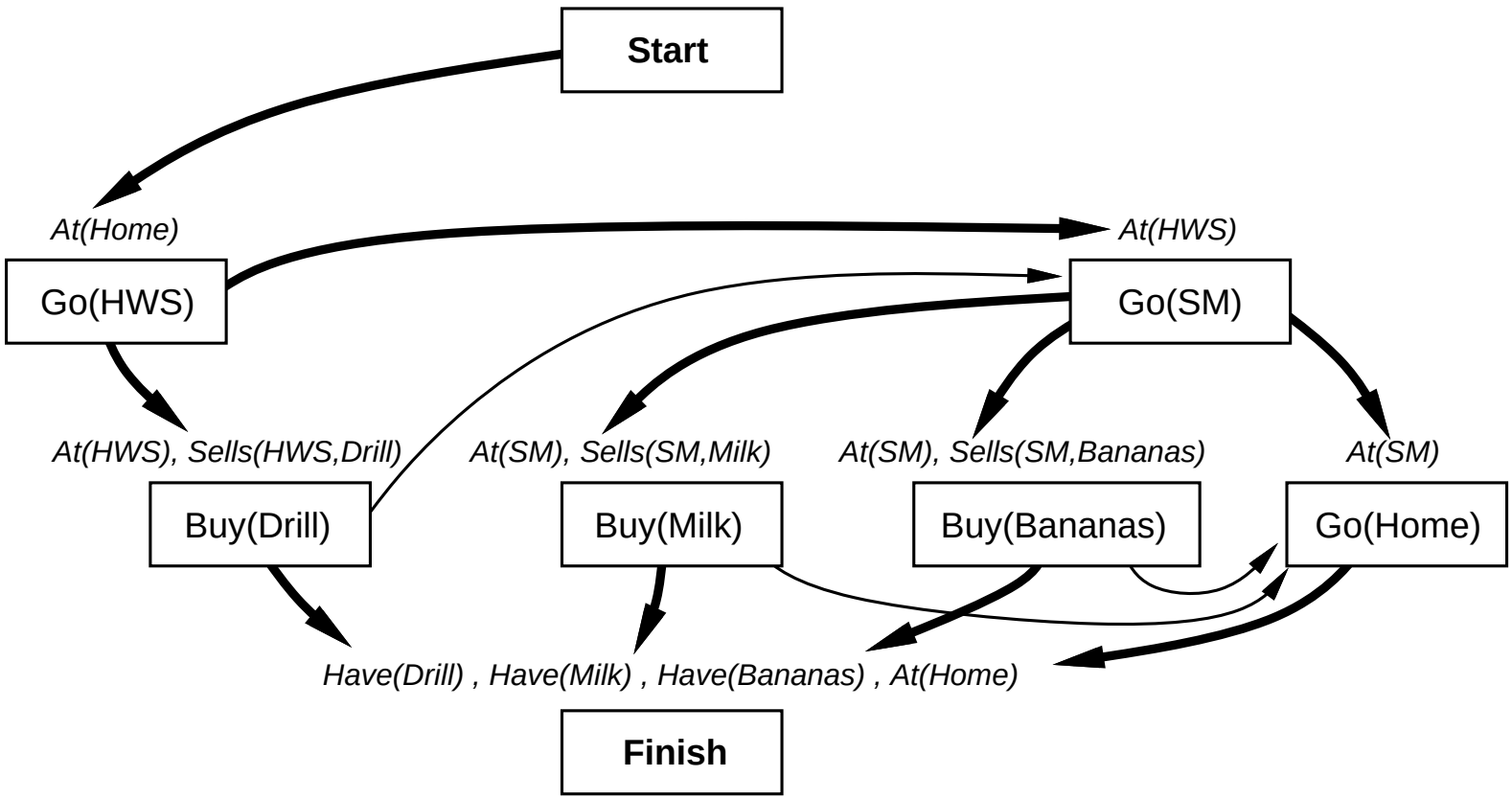
AI Planning

Exemplo de Planning com Ordem Parcial

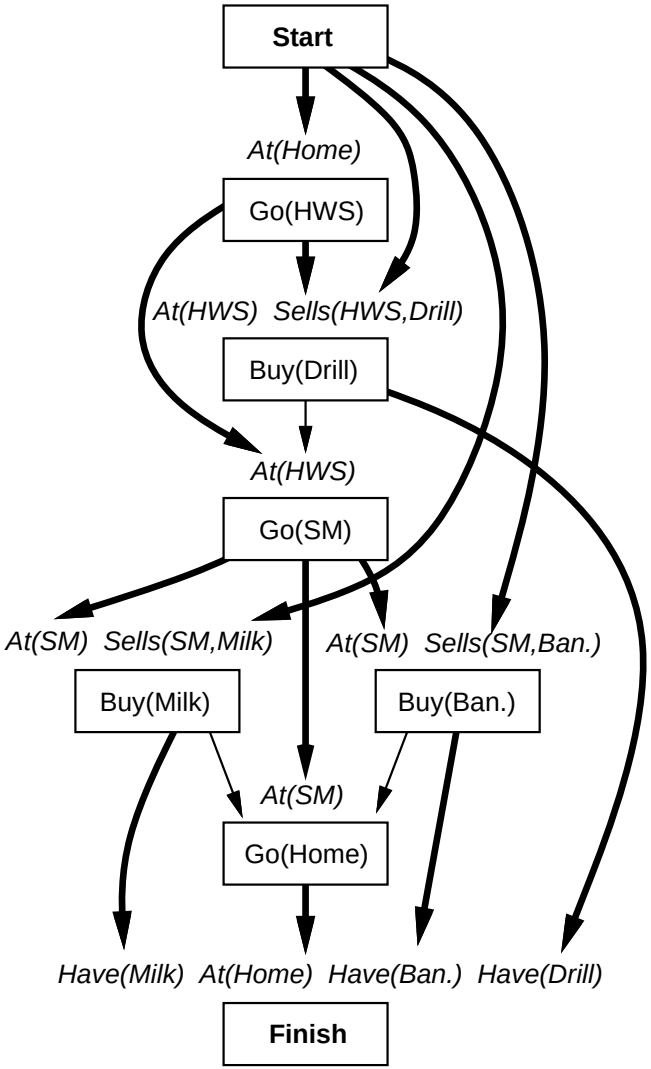


AI Planning

Exemplo de Planning com Ordem Parcial



AI Planning


Start
At(Home)
Go(HWS)
At(HWS) Sells(HWS,Drill)
Buy(Drill)
At(HWS)
Go(SM)
At(SM) Sells(SM,Milk)
At(SM) Sells(SM,Ban.)
Buy(Milk)
Buy(Ban.)
At(SM)
Go(Home)
Have(Milk) At(Home) Have(Ban.) Have(Drill)
Finish

Algoritmo Gerador de Planos: POP – Partial-Order Planner

- Algoritmo *não-determinístico*.
- Utilização de **choose** e **fail**.
- Começa com um plano parcial mínimo.
- A cada passo expande o plano ao tentar alcançar uma pré-condição c de um passo S_{need}.
- Estende o plano: através dos passos do plano parcial ou através de um conjunto de operadores que alcançam as pré-condições.
- Guarda o link causal para a pré-condição alcançada, para poder resolver as 'ameaças'.
- Importante: SELECT_SUBGOAL não tem várias alternativas.
- Razões: todas as precond precisam ser consideradas, comutativo.

POP – Partial-Order Planner

function POP(*initial*, *goal*, *operators*) **returns** *plan*

plan ← MAKE-MINIMAL-PLAN(*initial*, *goal*)
loop do
  if SOLUTION?(*plan*) **then return** *plan*
  S_{need}, c ← SELECT-SUBGOAL(*plan*)
  CHOOSE-OPERATOR(*plan*, *operators*, S_{need}, c)
  RESOLVE-THREATS(*plan*)
end

---

function SELECT-SUBGOAL(*plan*) **returns** S_{need}, c

pick a plan step S_{need} from STEPS(*plan*)
  with a precondition c that has not been achieved
return S_{need}, c

---

procedure CHOOSE-OPERATOR(*plan*, *operators*, S_{need}, c)

choose a step S_{add} from *operators* or STEPS(*plan*) that has c as an effect
if there is no such step **then fail**
add the causal link $S_{add} \xrightarrow{c} S_{need}$ to LINKS(*plan*)
add the ordering constraint $S_{add} \prec S_{need}$ to ORDERINGS(*plan*)
if S_{add} is a newly added step from *operators* **then**
  add S_{add} to STEPS(*plan*)
  add $Start \prec S_{add} \prec Finish$ to ORDERINGS(*plan*)

---

procedure RESOLVE-THREATS(*plan*)

for each S_{threat} that threatens a link $S_i \xrightarrow{c} S_j$ in LINKS(*plan*) **do**
  choose either
    Promotion: Add $S_{threat} \prec S_i$ to ORDERINGS(*plan*)
    Demotion: Add $S_j \prec S_{threat}$ to ORDERINGS(*plan*)
  if not CONSISTENT(*plan*) **then fail**
end

Gerador de Planos com Operadores Parcialmente Instanciados

- POP não trata de valores possíveis para uma variável.
- Ex: efeito $\neg At(x)$ de algum operador deve ser considerado uma ameaça para uma condição $At(Home)$?
- O efeito em questão é uma *possível* ameaça. Soluções possíveis:
  - 1) Resolver agora com uma restrição de igualdade: se o gerador de planos escolhe um operador cujo efeito é $\neg At(x)$, adicionar $x = HWS$ de forma a não ameaçar a condição $At(Home)$ já alcançada.
  - 2) Resolver agora com uma restrição de desigualdade: $x \neq Home$ torna a unificação mais complexa.
  - 3) Resolver mais tarde: deixar que $\neg At(x)$ seja uma possível ameaça. Se, mais tarde, $x = Home$, resolver. Desvantagem: difícil de saber se o plano é solução.

Gerador de Planos com Operadores Parcialmente Instanciados

- Usando operadores não totalmente instanciados, temos que garantir que todas as instanciações vão chegar ao objetivo.
- Definição: um passo S_i **alcança** uma pré-condição c do passo S_j se:
  1. $S_i \prec S_j$ e S_i tem um efeito que necessariamente unifica com c, e
  2. não há nenhum passo S_k tal que $S_i \prec S_k \prec S_j$ em alguma linearização do plano, e S_k tem um efeito que possivelmente unifique com $\neg c$.
- Algoritmo POP pode ser visto como um procedimento de prova de que cada pré-condição é alcançada.
- Novo procedimento CHOOSE_OPERATOR devolve o S_i que atende condição (1). Novo RESOLVE_THREAT atende condição (2).
- Usam abordagem de "resolver possíveis ameaças mais tarde".

Operadores Parcialmente Instanciados

procedure CHOOSE-OPERATOR($plan, operators, S_{need}, c$)

choose a step S_{add} from *operators* or STEPS(*plan*) that has c_{add} as an effect
such that u = UNIFY(c, c_{add}, BINDINGS(*plan*))
if there is no such step
then fail
add u to BINDINGS(*plan*)
add $S_{add} \xrightarrow{c} S_{need}$ to LINKS(*plan*)
add $S_{add} \prec S_{need}$ to ORDERINGS(*plan*)
if S_{add} is a newly added step from *operators* **then**
add S_{add} to STEPS(*plan*)
add $Start \prec S_{add} \prec Finish$ to ORDERINGS(*plan*)

---

procedure RESOLVE-THREATS(*plan*)

for each $S_i \xrightarrow{c} S_j$ **in** LINKS(*plan*) **do**
for each S_{threat} **in** STEPS(*plan*) **do**
for each c' **in** EFFECT(S_{threat}) **do**
if SUBST(BINDINGS(*plan*), c) = SUBST(BINDINGS(*plan*), $\neg c'$) **then**
choose either
Promotion: Add $S_{threat} \prec S_i$ to ORDERINGS(*plan*)
Demotion: Add $S_j \prec S_{threat}$ to ORDERINGS(*plan*)
if not CONSISTENT(*plan*)
then fail
end
end
end

Gerador de Planos com Operadores Parcialmente Instanciados

- POP é "sound" e completo qdo trata de operadores parcialmente instanciados?
- Se todas as ameaças forem resolvidas e POP conseguir gerar um plano completo totalmente instanciado (por exemplo, instanciando variáveis através do conjunto de valores possíveis, "binding list"): sound.
- É completo porque o algoritmo gera todos os possíveis planos que atendem condição (1), e remove todos os planos que não atendem condição (2).

Engenharia do Conhecimento para Geração de Planos

- Metodologia para resolver problemas com a abordagem de geração de planos:
  - Decidir sobre o que falar.
  - Decidir o vocabulário de condições (literais), operadores, e objetos.
  - Codificar operadores.
  - Codificar uma descrição de uma instância do problema (estado inicial, por exemplo).
  - Apresentar problemas ao gerador de planos e obter planos (problema = objetivo).

Ex 1: O Mundo dos Blocos

- Sobre o que falar? blocos, mesa, pilhas de blocos, regras para movimentação de blocos.
- Vocabulário?
  - objetos (blocos e mesa) representados por constantes.
  - $Sobre(b, x)$ usado para representar que bloco b está sobre x, onde x pode ser um outro bloco ou a mesa.
  - Operador para mover blocos: $Move(b, x, y)$, move bloco b da posição no topo de x para a posição no topo de y.
  - Não podemos representar $\neg\exists x\ Sobre(x, b)$ como pré-condição, então usamos $TopoLimpo(b)$.

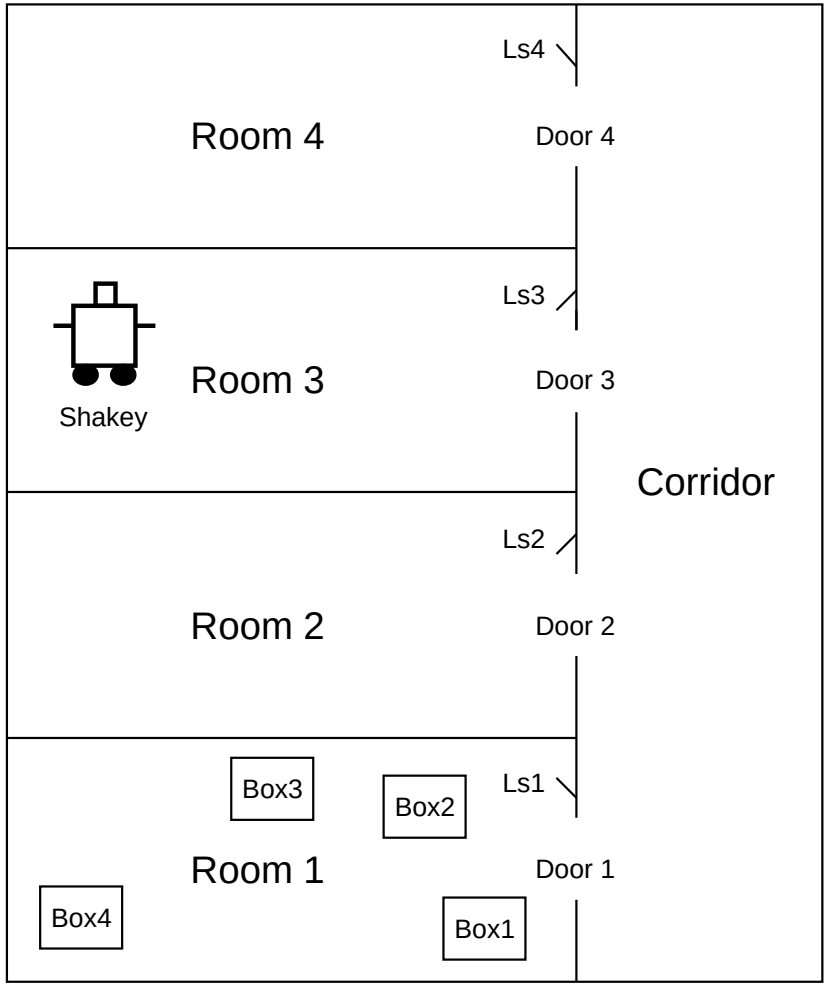
Ex 1: O Mundo dos Blocos

- Operadores?

  Op(ACAO: Move(b,x,y),
  PRECOND: Sobre(b,x) $\wedge$ TopoLimpo(b) $\wedge$ TopoLimpo(y),
  EFEITO: Sobre(b,y) $\wedge$ TopoLimpo(x) $\wedge$
  $\neg$ Sobre(b,x) $\wedge$ $\neg$ TopoLimpo(y))

- Problema: não opera bem quando x ou y é uma mesa.
- Solução: introduzir nova ação p/ mover um bloco b de x para a mesa (MovePara Mesa(b,x)) e nova interpretação para TopoLimpo(x) (há espaço vazio em x para colocar um bloco).
- Nada para prevenir planner de sempre usar Move(b,x,Mesa) invés de MoveParaMesa(b,x). Espaço de busca maior, mas encontra solução.
- Sobre(B,C,C) pode ocorrer no plano.

Ex 2: O Mundo de Shakey



Ex 2: O Mundo de Shakey

- O que falar? caixas, interruptores, portas, posições, regras para movimentação.
- Vocabulário e Operadores?
  - $Va(y)$: vai da posição corrente para a posição y, pré-condição $Em(Shakey, x)$ representa a posição corrente. Para este problema, $EstaEm(x, r) \wedge EstaEm(y, r)$ para poder representar q uma porta está em duas salas ao mesmo tempo e traçar plano para Shakey passar de uma sala p/ outra.
  - $Empurrar(b, x, y)$: empurrar objeto b de posição x para y.
  - $Subir(b)$: subir numa caixa.
  - $Descer(b)$: descer de uma caixa.
  - $AcenderLuz(int)$.
  - $ApagarLuz(int)$.