Planejamento 1

Capítulo 11

Como um agente pode tirar vantagens da estrutura de problemas para construir planos complexos (sequência de ações) que ele possa vir a executar

Problemas

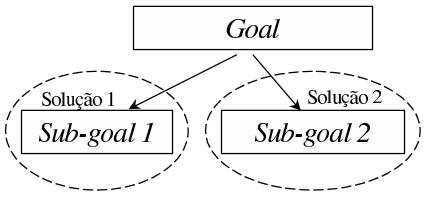
- **Agente reativo**: seleciona ações baseando-se em percepções ou na representação interna do estado do mundo
- **Agente *problem-solving***: decide o que fazer procurando uma sequência de ações que levam a um estado goal (ex.: algoritmos de busca)
- **Agente baseado em conhecimento**: seleciona ações baseando-se na representação lógica explícita do estado corrente e dos efeitos de suas ações

Agente que planeja

- Pode ser visto como um problema de busca
  - **agente *problem-solving*** que faz busca num espaço de estados. A sequências de ações corresponde a um caminho da árvore de busca**.**
  - **agente de planejamento** que faz busca num espaço de planos: cada nó é uma sequência de ações.
- Pode ser visto como provador de teoremas de propósito específico que raciocina eficiente-mente sobre descrições de ações (axiomas)

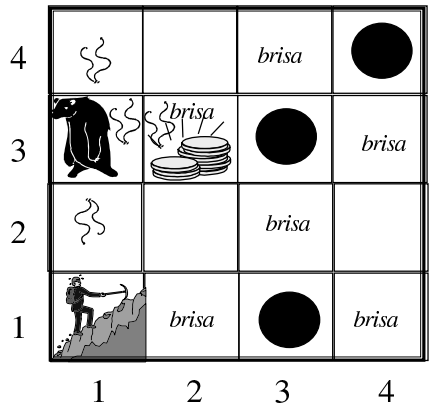
Idéias básicas sobre planejamento

- Dividir problemas complexos em sub-goals cuja solução pode ser combinada para fornecer uma solução (plano) para o problema completo



- Execução do plano: requer monitoração em ambientes que envolvam o problema de contingência

O mundo de Wumpus



Objetivo: sair da caverna depois de pegar o ouro

Ambiente

- quadrados adjacentes ao Wumpus *cheiram* mal
- quadrados adjacentes ao abismo possuem uma *brisa*
- um quadrado *brilha sss* contém o ouro
- o agente pode estar direcionado: *N, S, L, O*
- *atirar* a flexa mata o Wumpus se o agente estiver de frente para ele
- o agente só possue uma flexa
- *pegar* o ouro só é possível se o agente estiver no mesmo quadrado que o contém
- *soltar* deixa o ouro no mesmo quadrado em que o agente estiver

7

O mundo de Wumpus

- **percepção**: brisa, cheiro, brilho, escuta, choque (não percebe sua localização)
- **ação**: escala, vira para direita, vira para esquerda, vai para frente, segura, solta, atira
- **goals**: encontrar o ouro e trazê-lo para o início o mais rápido possível sem entrar num quadrado com abismo ou com Wumpus
  - O agente morre se entra em um quadrado que possue um wumpus vivo ou um abismo

8

Mundo de wumpus

- Determinístico?
- Acessível?
- Estático?
- Discreto?

9

Agente simples de planejamento

- No mundo do Wumpus o acesso é local
- Se o mundo for totalmente acessível um agente pode usar a percepção para construir um modelo completo do estado atual (localização do agente, do wumpus, e dos abismos)
- O mundo muda somente através das ações do agente (ações x eventos)

10

Agente simples de planejamento

Geração do plano: dado o estado do mundo e um goal, o agente pode chamar um algoritmo de planejamento ((*Ideal-Planner*) para gerar um plano de ações

Execução do plano: o agente poderá executar uma ação (passo do plano) por vez

Agente simples de planejamento

Function Simple-Planning-Agent (*percept*) **returns** an *action*
static: *KB* (includes action description)
p, a plan, initially *NoPlan*
t, a counter to indicate time, initially 0
local variables: *G,* a goal, *current-state* is a current state description
TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
current-state ← STATE-DESCRIPTION(*KB*,*t*)
if *p* = *NoPlan* **then**
G ← ASK(*KB*, Make-Goal-Query(*t*))
p ← IDEAL-PLANNER(*current-state, G, KB*)
if *p*=*NoPlan* **or** *p* is empty **then** *action* ← *NoOp ; p* ← *NoPlan*
else *action* ← FIRST(*p*)
p ← REST(*p*)
TELL(*KB*, MAKE-PERCEPT-SENTENCE(*action*, *t*))
t ← *t* + *1*
return *action*

Planejamento x Busca

- Diferem na maneira como representam goals, estados, ações e planos
- Diferem na maneira de raciocinar

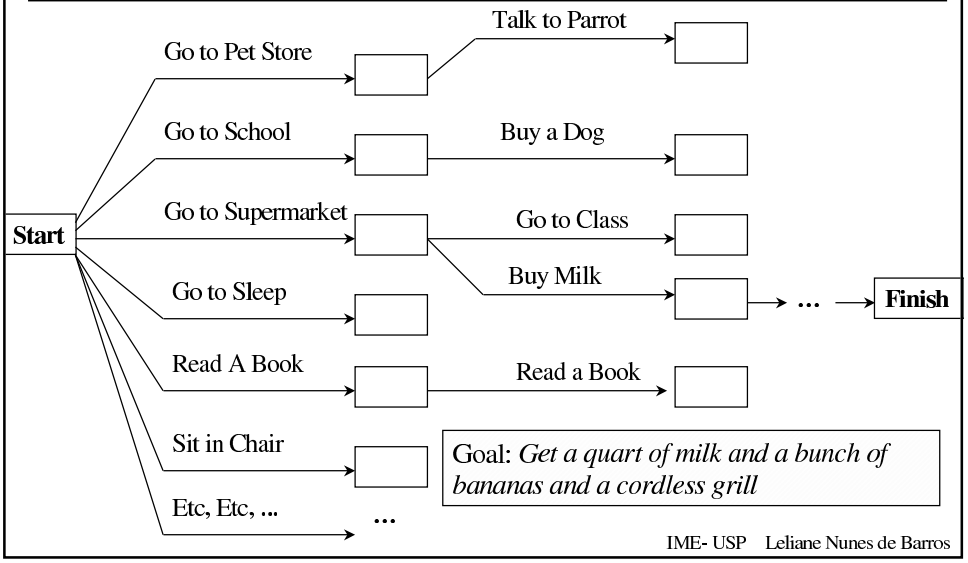
13

Problem-solver baseado em busca

- Representação de ações:
  - programas (procedimentos) que geram sucessores
- Representação de estados:
  - descrição completa do estado inicial e os programas que representam as ações, geram descrições completas dos estados sucessores
- Representação de goal:
  - na forma de teste de goal e função heurística. O agente não pode olhar para dentro e escolher ações que possam ser úteis para a realização do goal
- Representação de planos:
  - sequência de ações construida sequencialmente a partir do estado inicial (caminho da árvore de busca)

14

Exemplo: busca para frente através de um espaço de situações



15

Busca para planejamento - dificuldades

- Existem **muitas ações e estados** para serem considerados (fator de ramificação e profundidade da solução)
- **Informações sobre o goal**: a função de avaliação heurística pode somente escolher qual estado está mais próximo do goal - não pode descartar ações
  - goal não pode ser decomposto e não está diretamente relacionado com a descrição de ações
- Considerar a sequência completa a partir do estado inicial não permite que o agente raciocine sobre qual parte do goal é mais fácil resolver no estado corrente
- Comportamento do agente: considera todas as ações aplicáveis e usa a função heurística para avaliar o estado - não raciocina sobre o plano

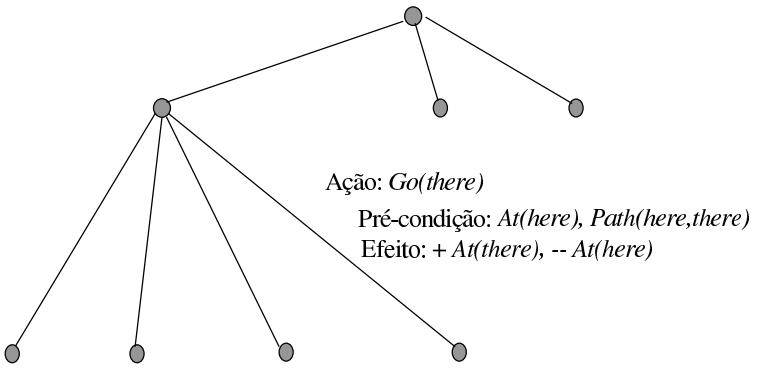
16

Como resolver as dificuldades dos agentes que fazem busca?

- O agente precisa de uma maneira mais flexível para estruturar suas deliberações
- Solução de planejamento: **"abrir" a representação** de estados, goals e ações

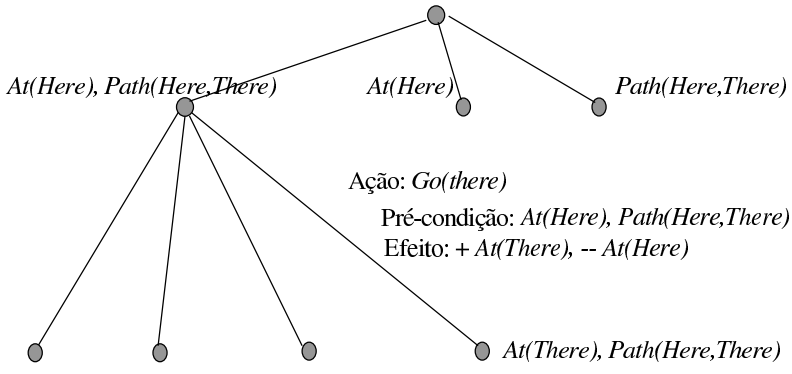
17

Busca com ações estruturadas

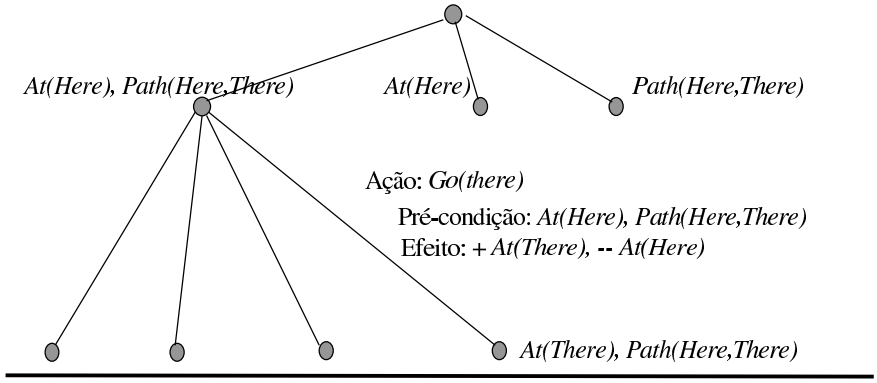


18

Busca com estados estruturados



Busca com planos estruturados



Plano (Passos do plano: {P_1: *Op*(Ação: *Start* Pré-condição: {}),
P_2: *Op*(Ação: *Finish* Pré-condição: *At(There) and Path(Here,There)* },
Ordenações: {$P_1 < P_2$},
Unificações: { },
Ligações Causais: { })

Representação de Conhecimento estruturada

Algoritmos de planejamento devem usar alguma **linguagem formal** para descrever estados, goals ou ações - LPO ou um sub-conjunto de LPO:

- estados e goals: conjunto de sentenças da lógica
- ações: implicações lógicas (precondições e efeitos)

Lógica Proposicional

Objetos da linguagem:

- átomos/proposições (P, Q, homem-rico, João-é-pai-de-Maria, … etc)
- Conectivos lógicos:
  - ¬ negação
  - ^ conjunção (AND)
  - v disjunção (OR)
  - => implicação (condicional)
  - <=> equivalência (bicondicional)

LP - Sintaxe

Sentença => SentençaAtômica | SentençaComplexa

SentençaAtômica => *Verdade* | *Falso* | P | Q | R | …

SentençaComplexa => (Sentença) |
Sentença Conectivo Sentença |
¬ Sentença

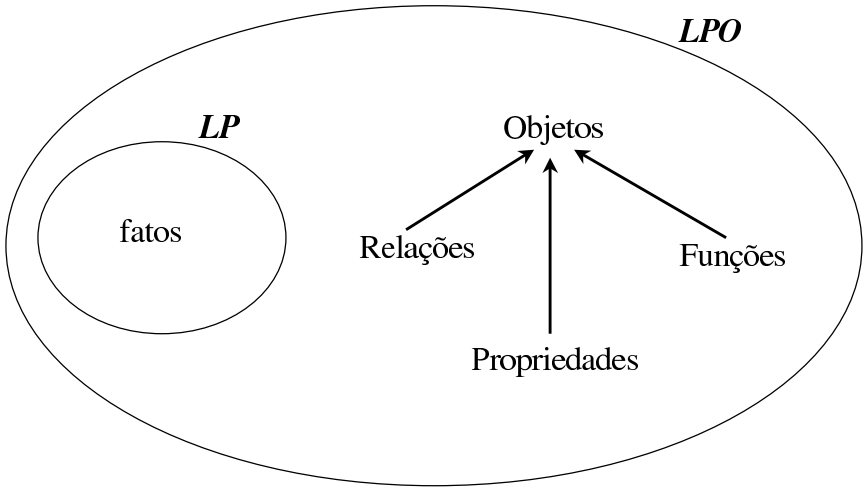
Conectivo => ^ | v | <=> | =>

Gramática BNF (Backus-Naur-Form)

Lógica Proposicional - sentenças

- Uma **sentença,** também chamada de fórmula ou fórmula bem formada (*wff - well formed formula*), é definida como:
  - Um átomo é uma sentença
  - Negação de uma sentença é uma sentença
  - Conjunção, disjunção e implicação de 2 sentenças é uma sentença
- **Literal**: um átomo ou um átomo negado

LP versus LPO



Lógica de Predicados de Primeira Ordem (LPO)

- objetos: indivíduos do mundo (pessoas, casas, números, cores)
- funções: mapeamento de indivíduos para indivíduos. Ex.: pai-de(Maria)=Joao, cor-de(Carro)=Vermelha
- relações: mapeamento de indivíduos para valores Verdades. Ex.: maior-que(5,3), verde(grama); cor(Carro, Vermelho)

LPO

- Constantes: Joao, 2, USP, ...
- Predicados: irmão, >, ...
- Funções: RaizQuadrada, DistânciaXY, ...
- Variáveis: x, y, a, b, ...
- Conectivos: ^, v, ~, =>, <=>
- Igualdade: =
- Quantificadores: ∀ (Universal), ∃ (Existencial)

Lógica de Predicado de Primeira Ordem

LPO

- Fatos em LPO
- LPO:
  - **Sentenças atômicas**: predicado(termo$_1$, ..., termo$_n$)
    irmão(Maria, João).
  - **Termo**: constante, variável, função(termo$_1$, ..., termo$_n$)
    casado(Pai-de(Joao), Mae-de(Joao)).
  - **Sentenças complexas**: predicado1(termo$_1$, ..., termo$_n$) => predicado2(termo$_1$', ..., termo$_n$')
    irmão(Maria, João) => pai-de(Maria, X) ^ pai-de(João, X).

função

predicado

Planejamento (I)

estado

Goal: *Have*(*Milk*) ^ ...

Buy(*x*) → *Have*(*x*)

ação

estado

O agente conclue que vale a pena incluir *Buy(Milk)* no plano (note que esse raciocínio não é uma dedução lógica mas uma abdução lógica!)

☛ Planejamento permite ao agente fazer uma correlação direta entre estados, ações e goals

Planejamento (II)

Planejamento permite:

- ✚ que o agente adicione ações em qualquer lugar do plano, relaxando o requisito de se construir um plano sequencialmente, como na busca).
- ✚ não existe um compromisso entre a ordem de planejamento (geração do plano) e a ordem de execução

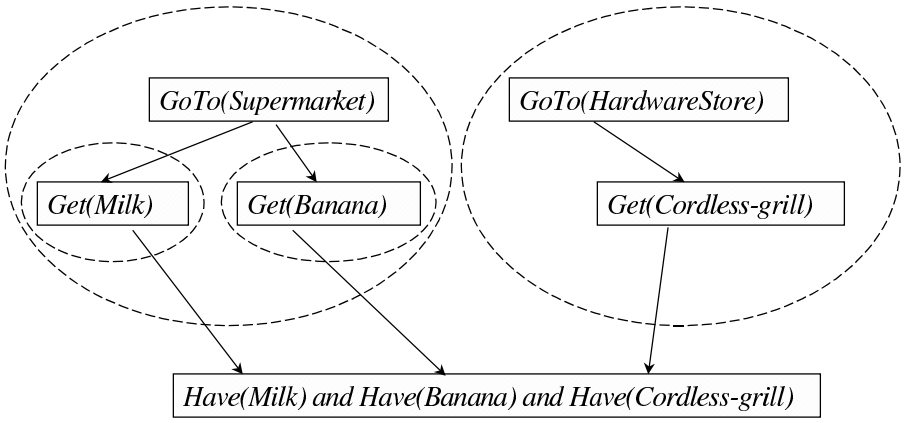
Planejamento (III)

- ☛ Muitas partes do mundo são independentes de outras partes.

Dado o goal: (*get a quart of milk)* *and* *(get a bunch of bananas)* *and* *(get a cordless grill)*

- ☛ Planejamento permite que se selecione um sub-goal de um goals descrito através de uma sentença conjuntiva
  - ➡ estratégia de resolução de problemas do tipo *divide-and-conquer*

Planejamento (III)



Descrição de ações do tipo Strips

- STRIPS (***ST****anford* ***R****esearch* ***I****nstituto* ***P****roblem* ***S****olver*)
- Desenvolvido em 1970 por Nills Nilsson (originou do GPS)
- *Strips-like language:* linguagem usada pela maioria dos planejadores clássicos
- PDDL: extensão da linguagem Strips, usada em competições de planejamento

Descrição de estado

- Conjunções de literais instanciados sem funções, ou seja, predicados aplicados a símbolos constantes, possivelmente negados

  Estar-em(Casa) ^ ¬ *Ter(Leite)* ^ ¬ *Ter(Bananas)* ^ ¬ *Ter(Grelha)* ^ ...
- Uma descrição de estado não precisa ser completa: um estado pode corresponder a um conjunto estados (*possible completions), ou então ...*
- *Closed World Assumption*: tudo que for incluido na descrição de estado é verdade; o que não for declarado é falso!

  Estar-em(Casa)
- Goals: também descritos como conjunções de literais, mas podem conter variáveis

  Estar-em(X) ^ *Ter(Leite)* ^ *Ter(Bananas)* ^ *Ter(Grelha)*

Descrição de goal em Planejamento

- Dado um goal, o agente planejador *pergunta* por uma sequência de ações (plano) que quando executada torna o estado goal verdadeiro, a partir do estado inicial
- Provador de teoremas: *pergunta* se a sentença do goal é verdadeira em uma dada Base de Conhecimento (KB)

Operador Strips

Ação: *Buy(x)*

Pré-condição: *At(p), Sells(p,x)* //*conjunção de átomos(literais positivos)*

Efeito: *Have(x)* //*conjunção de literais (positivos ou negativos)*
(a versao original de Strips divide os efeitos em **add list** *e* **delete list)**

Linguagem restrita ==> algoritmo eficiente

Representação gráfica:

At(p) *Sells(p,x)*

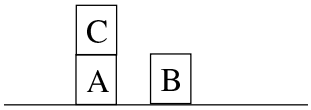
Buy(x)

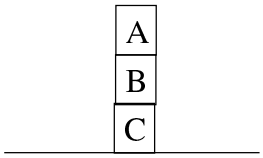
Have(x)

Mundo dos Blocos

- *Move(x,y,z)* move bloco x de cima de y para cima de z e z pode ser a mesa ou outro bloco
- *Move* é aplicável somente se x e z estiverem livres (*clear*) e x estiver sobre y.
- Um bloco está livre (*clear*) se não possue nenhum outro bloco em cima dele
- A mesa possue sempre espaço livre

Mundo dos Blocos





Estado Inicial:
On(B,table)
On(A,table)
On(C,A)
Clear(C)
Clear(B)

Ação: *Move(x,y,z)*

...

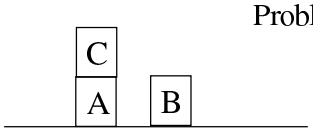
Goal:
On(A,B) ^ On(B,C)

Mundo dos Blocos (Subarao)

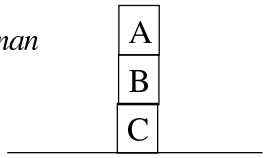
- Outros operadores para o mundo dos blocos
  - Pickup(x)
  - Putdown(x)
  - Stack(x,y)
  - Unstack(x,y)
  - Puton(x,y,z)
  - Puton_table(x,y)
  - ou somente G-Puton(x,y) com efeito condicional

Mundo dos Blocos

Problema: *Anomalia de Sussman*



Estado Inicial



Estado Gol

Clear(x), On(x,z), Clear(y)

PutOn(x,y)

¬ *On(x,z), ¬ Clear(y),*
Clear(z), On(x,y),

Clear(x), On(x,z)

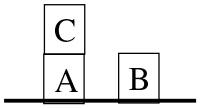
PutOnTable(x)

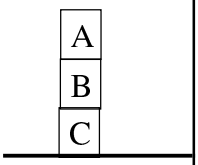
¬ *On(x,z),Clear(z),*
On(x,Table),

Exemplo: Anomalia de Sussman

Start

On(C,A) On(A,Table) Clear(B) On(B,Table) Clear(C)

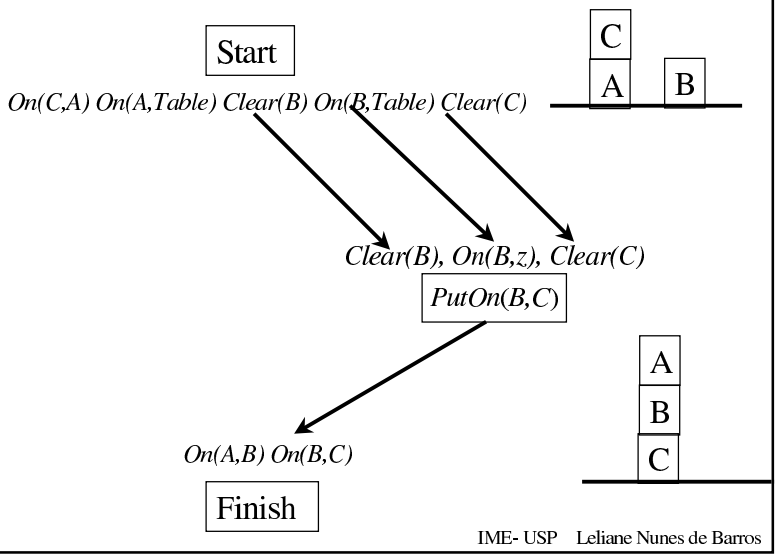




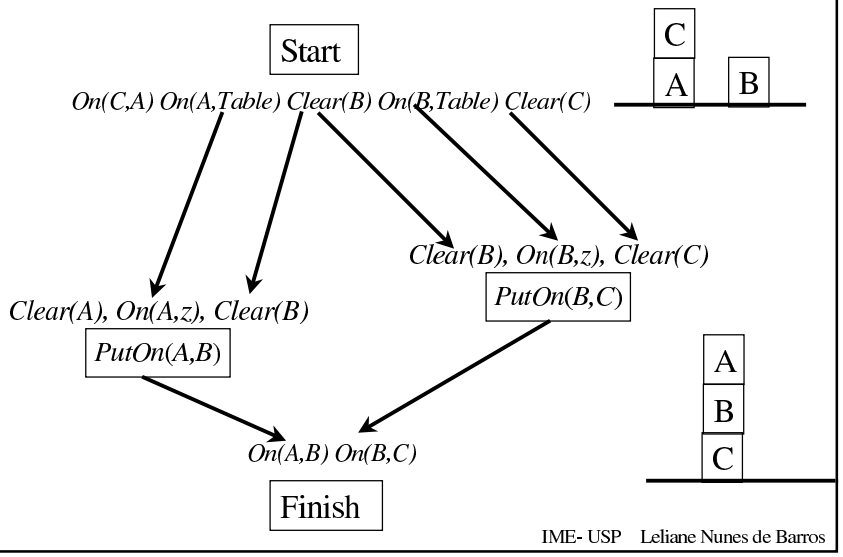
On(A,B) On(B,C)

Finish

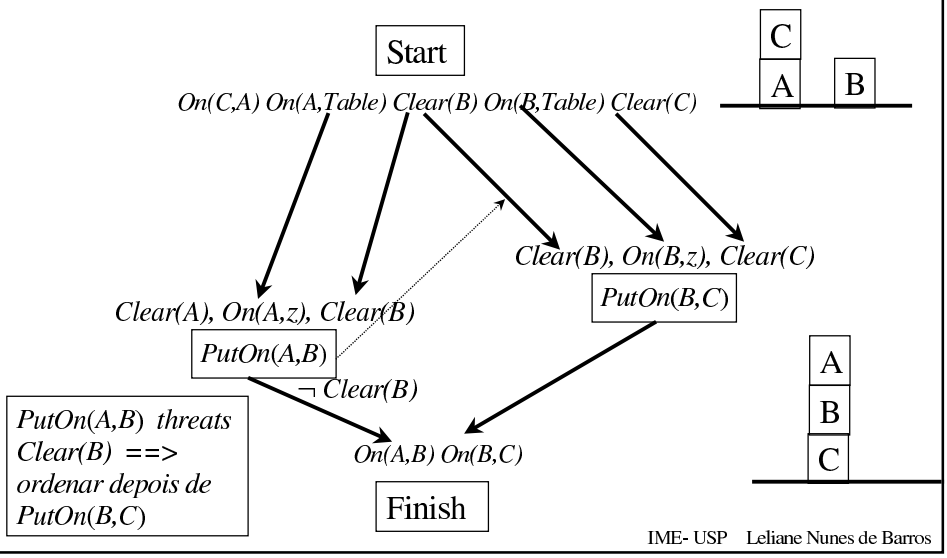
Anomalia de Sussman



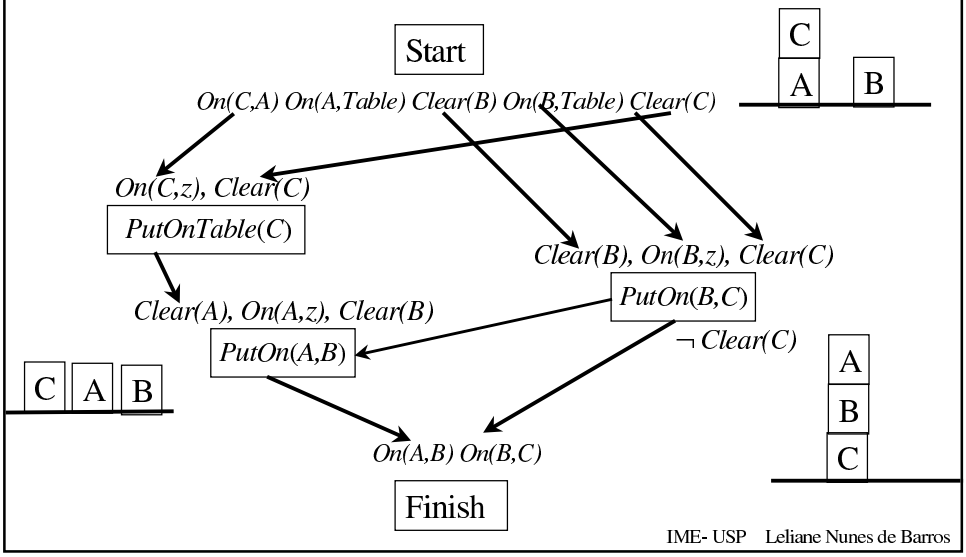
Anomalia de Sussman



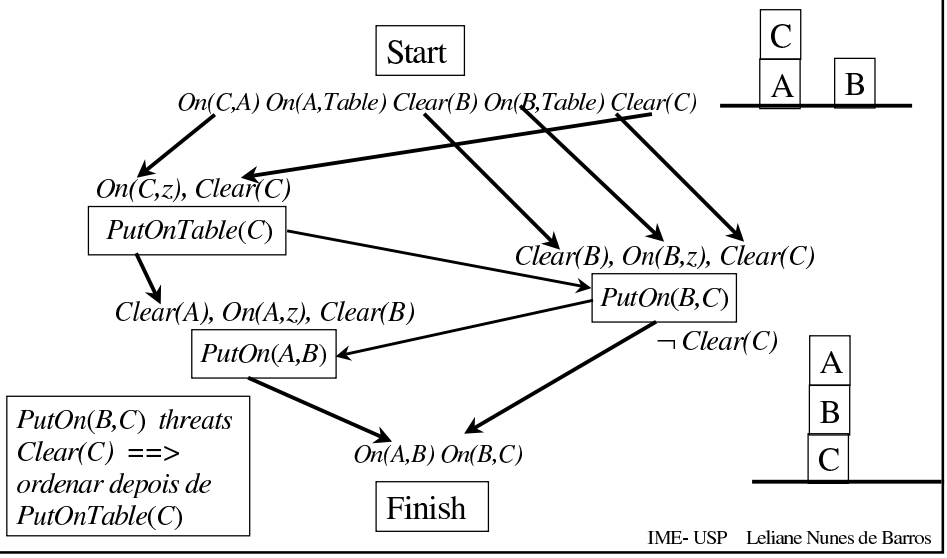
Anomalia de Sussman



Anomalia de Sussman



Anomalia de Sussman



Agente de planejamento simples (capítulo de agentes)

Function Simple-Problem-Solving-Agent (*p*) **returns** an *action*
input: *p,* a percept
static: *s*, an action sequence, initially empty
state, some description of the current world state
g, a goal, initially null
problem, a problem formulation
state ← UPDATE-STATE (*state, p*)
if *p* is empty **then**
g ← FORMULATE-GOAL*(state)*
problem ← FORMULATE-PROBLEM (*state, g*)
s ← SEARCH (*problem*)
action ← RECOMMENDATION (*s, state*)
s ← REMAINDER (*s, state*)
return *action*