ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Avenida Professor Luciano Gualberto, travessa 3 nº 158 CEP 05508-900 São Paulo SP
Telefone: (11) 3091-5583 Fax (11) 3091-5294

Departamento de Engenharia de Computação e Sistemas Digitais

Inteligência Artificial

Planejamento Clássico I

Autores: *Paulo da Silveira Moraes Neto* *Rodrigo Rosa* *Rodrigo Suzuki Okada*	Data de emissão: 15/10/2008 Grupo 2

PLC0000.RS.001.00

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Avenida Professor Luciano Gualberto, travessa 3 nº 158 CEP 05508-900 São Paulo SP
Telefone: (11) 3091-5583 Fax (11) 3091-5294

Departamento de Engenharia de Computação e Sistemas Digitais

Índice

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Avenida Professor Luciano Gualberto, travessa 3 nº 158 CEP 05508-900 São Paulo SP
Telefone: (11) 3091-5583 Fax (11) 3091-5294

Departamento de Engenharia de Computação e Sistemas Digitais

Inteligência Artificial

Planejamento Clássico

1 Introdução

Este documento tem como objetivo dar uma introdução básica nos conceitos de planejamento clássico, dando exemplos de como este modelo de inteligência artificial funciona.

Para seguir este documento, um conhecimento de modelos matemáticos é de fundamental importância.

2 Conceitos Básicos

2.1 Planejamento

Entende-se planejamento como sendo a tarefa de se procurar uma seqüência de ações que leve a um objetivo. Em outras palavras, é um método para encontrar a solução de um problema.

O planejamento de qualquer sistema começa a partir do entendimento do problema. Como a inteligência artificial é fortemente baseada na abstração da realidade em modelos, o problema deve ser decomposto em um modelo que possa ser, metodicamente, solucionado. Este modelo é escrito em uma linguagem padronizada, para que a solução possa ser encontrada de forma metódica, não aleatória.

Então, a seqüência de ações encontrada pelo planejamento nada mais é do que a solução procurada.

Duas das soluções mais comuns para este tipo de problema são por busca (ex: A*) e problemas solucionáveis por lógica.

2.2 Planejamento Clássico

Planejamento clássico é um tipo particular de planejamento, aplicado em ambientes observáveis, determinísticos, finito, estático e discreto. Este ambiente é dito como sendo ambiente clássico.

Estas características significam que o sistema pode é completamente visível em qualquer instante de tempo, com regras bem definidas (um estado, mais uma ação sempre leva a um mesmo estado), além de ter tamanho finito. São condições restritas, mas que resolver muitos dos problemas mais comuns.

2.3 Definição de Problema

O problema a ser resolvido, em geral, já existe no mundo real. O empecilho, em geral, é abstrair o problema em um modelo que possa ser facilmente resolvido, pois, de forma geral, um agente simples é incapaz de racionar, e somente executa ações previamente programadas. Já uma pessoa consegue abstrair o problema por si própria e encontrar a solução.

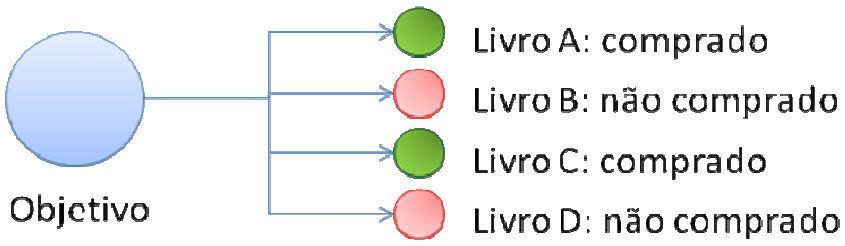
Assim, o agente deve receber estas regras, para que então, saiba como atingir um objetivo, criando modelos que ele possa seguir de forma automática.

Por exemplo, uma pessoa compra um livro procurando pelo seu nome, e caso queira comprar um livro específico, já o procura diretamente pelo nome. Já um agente simples não é capaz de fazer isso por conta própria. Se ele não tiver nenhuma informação prévia, irá procurar cegamente o livro até encontrá-lo, o que é ineficiente.

Assim, não basta apenas abstrair o problema: deve-se criar formas para que o agente saiba o quão próximo está da solução, isto é, criar heurísticas. Um agente sem heurísticas irá procurar cegamente pelo objetivo, testando se seu estado atual é o objetivo, sem saber se está próximo da solução ou não.

No caso de compra de múltiplo livros, uma heurística seria calcular o custo baseado no número de livro que ainda faltam ser comprados. Aliás, este é um exemplo de um objetivo passível de decomposição: o objetivo principal (comprar todos os livros) pode ser visto como o conjunto de pequenos objetivos, que seria a compra individual de cada livro.

Nestes casos, a heurística seria calcular o custo a partir do número de objetivos a serem atingidos. Assim, se tivermos 4 livros para comprar, cada livro faltante aumentaria o custo estimado em 1.



Objetivo Have(A) ^ Have(B) ^ Have(C) ^ Have(D)

Estado Have(A) ^ Have(C)

Custo 2

2.4 Linguagem

A linguagem do problema é forma com que o mesmo será representado no modelo. Apesar de não citar as tecnologias envolvidas para o sistema real, é quem define as regras a serem implementadas.

No planejamento clássico, uma linguagem é definida por 3 conjuntos: de estados, de ações e de objetivos, sendo que cada um será descrito a seguir.

2.4.1 Estado

Estado é a representação da situação atual do sistema. É composto por um conjunto de literais proposicionais que definem as variáveis do sistema. Por exemplo:

- Pessoa Pobre ^ Desconhecido
- Carga At (Plane1, Brasília)

Porém, não é permitido concatenar funções uma dentro da outra:

- Carga At (Plane1, capital (Brasil))

Qualquer variável não listada no estado não poderá ser assumida como falsa ou verdadeira. Esta regra varia para cada tipo de linguagem.

2.4.2 Objetivo

Um objetivo é um conjunto de literais mínimo que identifica a solução do problema. Não é necessariamente um estado, mas sim, as literais que um estado deve possuir para que seja considerado um objetivo.

Por exemplo, o objetivo Rico ^ Famoso de uma pessoa pode ser encontrado no estado Rico ^ Famoso ^ Bonito. Assim, este estado é um objetivo do sistema.

2.4.3 Ação

Uma ação define como será as transições entre os estados. Como cada estado é definido pelas suas literais, a ação é quem altera o valor das literais

É definida em 3 partes:

- **Identificador:** nome pelo qual será representado no sistema
- **Condição:** literais necessárias que o estado deve possuir para que a ação possa ser executada.
- **Efeito:** literais a serem incluídas ou retiradas do estado atual, passando para um novo estado.

Como exemplo, pode-se tomar um sistema aeroportuário:

O estado atual do sistema pode ser descrito como:

At (P1, SP) ^ At (P2, RJ) ^ Plane(P1) ^ Plane(P2) ^ Airport(SP) ^ Airport(RJ)

Já a ação de voar de um aeroporto ao outro, pode ser vista como:

Action(Fly(p, from, to),

PRECOND: At (p, from) ^ Plane(p) ^ Airport(from) ^ Airport(to)

EFFECT: ¬At (p, from) ^ At (p, to))

Esta ação, identificada por "*Fly (p, from, to)*", pode ser chamada quando o estado atual possuir um conjunto de literais que satisfaça a condição acima, alterando o estado atual com as literais escritas nos efeitos.

Pelo estado atual, temos que a ação Fly (P1, SP, RJ) é válida, pois com tais literais, a condição é satisfeita pelo estado atual:

At (P1, SP) ^ Plane(P1) ^ Airport(SP) ^ Airport(RJ)

p = P1

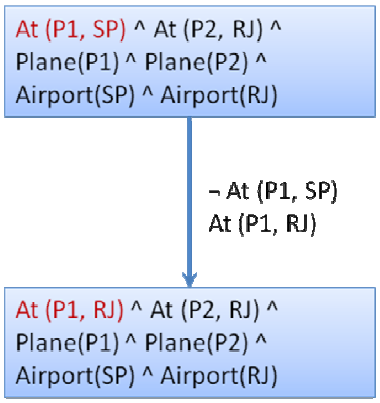
from = SP

to = RJ

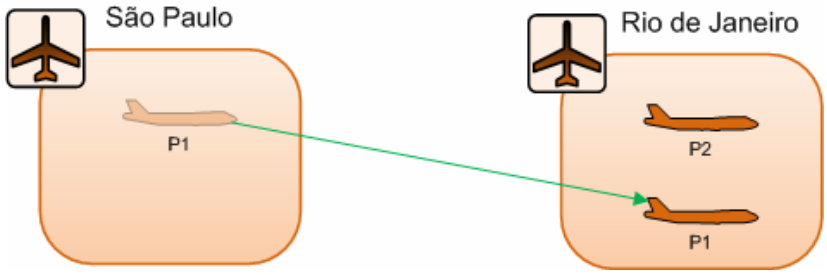
Ao executar esta ação, deve-se executar os efeitos listados:

¬At (P1, SP) ^ At (P1, RJ)

O que estes efeitos dizem é que o novo estado será um que, além das literais do estado anterior, tenha a literal *At (P1, SP)* excluída, e a literal *At (P1, RJ)* adicionada.



No modelo, isto seria equivalente à:



2.4.4 STRIPS

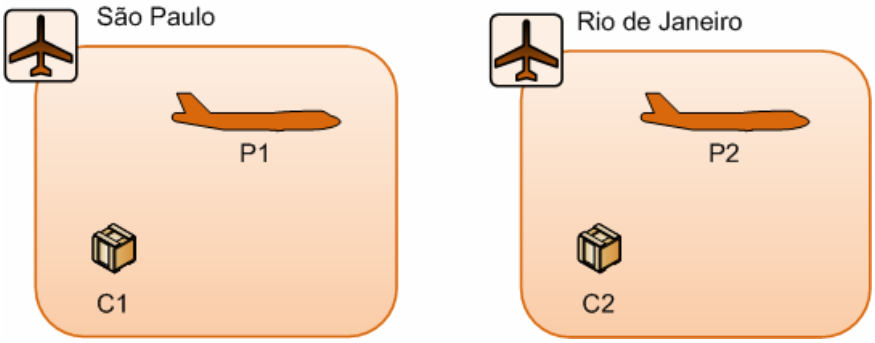
STRIPS é uma das linguagens mais utilizadas em planejamento clássico, obedecendo os 3 conjuntos anteriores. Como restrição, impõe que as literais de um estado sejam sempre positivas, com objetivos livre de funções.

Os estados também variam, pois literais não citadas sempre serão falsas, o que é uma simplificação muito útil em casos simples.

Existem outras linguagens, como ADL, mas que não serão abordadas neste documento.

2.5 Exemplo

Como exemplo, será utlizado o problema de transporte de cargas entre aeroportos.



Para este problemas, será definido que o objetivo será ter a carga C1 no Rio de Janeiro e a carga C2, em São Paulo.

As ações serão de carga, descarga e voar.

Assim, o sistema pode ser representado pelo seguinte conjunto de estados, objetivos e ações:

Init (

At(C1, SP) ^ At(C2, RJ) ^ At(P1, SP) ^ At(P2, RJ)

^ Cargo(C1) ^ Cargo(C2) ^ Plane(P1) ^ Plane(P2)

^ Airport(SP) ^ Airport(RJ)

)

Goal (

At(C1, RJ) ^ At(C2, SP)

)

Action (

Load(c, p, a),

PRECOND: At(c, a) ^ At(p, a) ^ Cargo(c) ^ Plane(p) ^ Airport(a)

EFFECT: ¬ At(c, a) ^ In(c, p)

)

Action (

Unload(c, p, a),

PRECOND: In(c, p) ^ At (p, a) ^ Cargo(c) ^ Plane(p) ^ Airport(a)

EFFECT: At(c, a) ^ ¬ In(c, p)

)

Action (

Fly(p, from, to),

PRECOND: At(p, from) ^ Plane(p) ^ Airport(from) ^ Airport(to)

EFFECT: ¬ At(p, from) ^ At(p, to)

)

O conjunto acima indica o estado inicial, com a posição dos aviões e das cargas, o objetivo das cargas, além das 3 ações, indicando o que deve fazer.

Realizando uma busca qualquer, pode-se chegar a uma solução, como por exemplo:

- *Load (C1, P1, SP)*
- *Fly (P1, SP, RJ)*

- *Unload (C1, P1, RJ)*
- *Load (C2, P2, RJ)*
- *Fly (P2, RJ, SP)*
- *Unload (C2, P2, SP)*

No fim destas ações, teríamos a carga C2 em São Paulo e C1, no Rio de Janeiro, o que completa o problema.

3 Busca no espaço de estados

Como são definidos tanto pré-condições quanto efeitos, o problema define um espaço de estados. O processo de percorrer esse espaço a patir de um estado inicial em direção ao estado objetivo é chamada de busca progressiva. Quando percorrido a patir do objetivo em direção ao estado inicial, o processo é chamado busca regressiva.

3.1 Busca progressiva no espaço de estados

Partindo do estado inicial, objetiva encontrar uma seqüencia de ações que leve ao estado objetivo.

Principais elementos:

Estado inicial: *Define as condições iniciais para o início da busca*

Ações: *são aplicáveis a um estado quando as pré-condições são satisfeitas.*

Teste de objetivo: *verifica se o estado satisfaz o objetivo do problema de planejamento*

Custo do passo: *tipicamente um.*

Problemas

- Ineficiência devido a não considerar o problema das ações irrelevantes. Todas as opções de ações são testadas em cada estado.

Isso faz com que a complexidade do problema cresça muito rápido

Ex: Considere um problema de transporte aéreo de carga no qual existem 10 aeroportos, cada um deles com 5 aviões e 20 unidades de carga.

Objetivo: *levar todas as cargas de A para B*

Cada avião pode voar para qualquer um dos 9 outros aeroportos e cada um dos pacotes pode ser carregado ou descarregado em qualquer aeroporto. Isso implica em uma árvore de busca com aproximadamente 100041 nós!!!

3.2 Backward State-Space Search

Parte do estado objetivo, buscando encontrar a seqüência de ações que podem levar ao estado inicial.

Permite que sejam consideradas apenas as ações relevantes. Um ação é relevante se conduz a uma das conjecturas da solução:

Ex: Dado o objetivo

Em(C1,B)^Em(C2,B)^Em(C2,B)...Em(C20,B)

Para Em(C1,B) o conjunto de ações que resultam nela é contido em Descarregar(C1,p,B), onde p é qualquer avião

Ações também devem ser consistentes. Uma ação é dita consistente se não "desfizer" nenhuma dos literais desejados já alcançados.

O processo de busca resultantes é:

- *Dada uma ação A que seja relevante e consistente, qualquer efeito positivo resultante de A deve ser apagado do objetivo*
- *Toda pré-condição é adicionada a não ser que ela já esteja enunciada*

3.3 Heurísticas

O problema de encontrar o número exato de ações que levam a um objetivo é um problema NP, entretanto é possível encontrar estimativas razoáveis sem cálculo excessivo. Temos de encontrar heurísticas que sejam admissíveis, ou seja, que não superestimem o custo.

Há duas abordagens possíveis

Problema relaxado

A heurística por problema relaxado considerar apenas parte do problema, mudando sua representação para simplificá-lo

Exemplos:

- Remover pré-condições das ações

- Remover efeitos negativos

Independência de sub-objetivo

A heurística por independência de sub-objetivo consiste em assumir que a custo de alcançar o conjunto de sub-objetivos é a mesmo que a soma do custo de alcançar cada um desses sub-objetivos de forma independente.

Será otimista quando a ações puderem causar efeitos que desfaçam algum sub-objetivo já alcançado

Será pessimista quando um duas ou mais ações puderem ser substituída por um única ação.

4 Planejamento de Ordem Parcial

4.1 Definição

O planejamento de ordem parcial busca resolver os problemas pela divisão do mesmo em subproblemas. Sendo necessário buscar resolver aos vários sub-objetivos, utilizando diversos sub-planos e ao final combiná-los.

Essa abordagem apresenta a vantagem de flexibilidade na ordem em que elabora o plano, sendo possível primeiro trabalhar as decisões "óbvias" ou "importantes" ao invés de atuar sempre em ordem linear e cronológica. Essa estratégia é chamada de estratégia de compromisso mínimo.

Um exemplo disso pode é o problema de calçar um par de sapatos sendo necessário primeiro calçar as meias.

Objetivo(SapatoDireitoCalçado, SapatoEsquerdoCalçado)

Iniciar()

Ação(SapatoDireito, PRECOND: MeiaDireitaCalçada, EFFECT: SapatoDireitoCalçado)

Ação(MeiaDireita, EFFECT: MeiaDireitaCalçada)

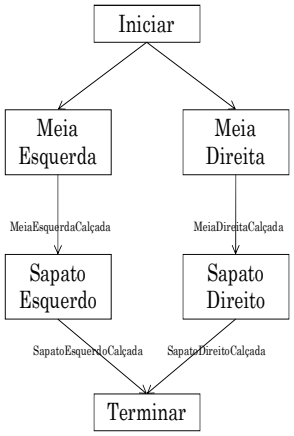
Ação(SapatoEsquerda, PRECOND: MeiaEsquerdaCalçada, EFFECT: SapatoEsquerdaCalçado)

Ação(MeiaEsquerda, EFFECT: MeiaEsquerdaCalçada)

Um planejador deve ser capaz de apresentar a seqüência de duas ações MeiaDireita seguido por SapatoDireito para alcançar o primeiro objetivo e a seqüência MeiaEsquerda seguido de SapatoEsquerdo para o segundo objetivo. Em seguida as duas seqüências podem ser combinadas para gerar o plano final. Qualquer algoritmo de planejamento que possa inserir duas ações em um plano

sem especificar qual delas deve acontecer primeiro é chamado planejador de ordem parcial.

Abaixo segue um grafo que representam a solução do problema de calçar o sapato. As ações fictícias denominadas Iniciar e Terminar que representam o início e o final do plano. O plano de ordem parcial pode ser linearizado em seis linearizações correspondentes em planos de ordem total.



Planejamento de Ordem Parcial

Planejamento de Ordem Total

No planejamento de ordem parcial os estados de busca são planos. Cada plano possui quatro componentes:

1. Um **conjunto de ações**, que contém as ações do problema de planejamento. O plano vazio possui apenas as ações *Iniciar* e *Terminar*. *Iniciar* não possui precondições e seus efeitos são todos os literais no estado inicial, já *Terminar* não possui efeitos e suas precondições são os literais de objetivo.

2. Um **conjunto de restrições de ordenação** na forma A < B (lida A antes de B). Isso significa que A deve ser executada antes de B, mas não necessariamente antes. Qualquer ciclo da forma A < B e B < A é considerada uma contradição.

3. Um **conjunto de vínculos causais** escrito como $A \xrightarrow{p} B$ e lido "A alcança p para B". Isso significa que *p* é um efeito de A e uma precondição de B. Isso também significa que *p* deve permanecer verdadeira desde o instante da ação A até o momento da ação B. Por esse motivo também é chamada de intervalo de proteção.

4. Um **conjunto de precondições abertas**. Uma precondição é aberta se ela não é alcançada por alguma ação do plano. Os planejadores devem reduzir o conjunto de precondições abertas até o conjunto vazio sem inserir contradições

Um plano é considerado consistente quando não existe nenhum ciclo no conjunto das restrições de ordenação e nenhum conflito com os vínculos causais. Se além de ser consistente não existir precondições abertas o plano é uma solução.

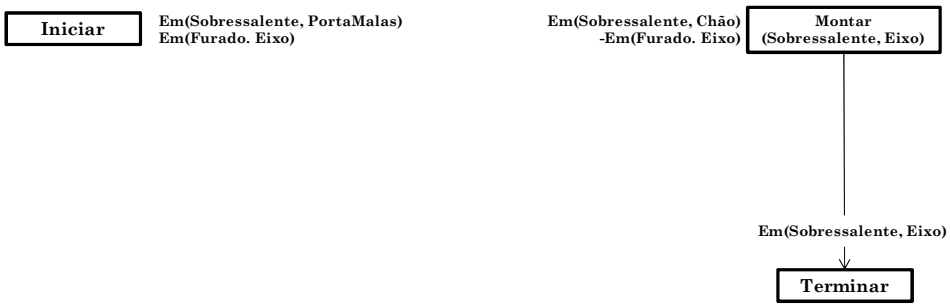
4.2 Exemplo

Como exemplo iremos resolver o problema da troca do pneu furado. Para trocar o pneu é necessário retirar o pneu sobressalente do porta-malas, o pneu furado do eixo e então colar o pneu sobressalente no eixo. Também é dado a opção de abandonar o carro durante a noite, e se isso for feito todos os pneus são roubados inclusive o sobressalente.

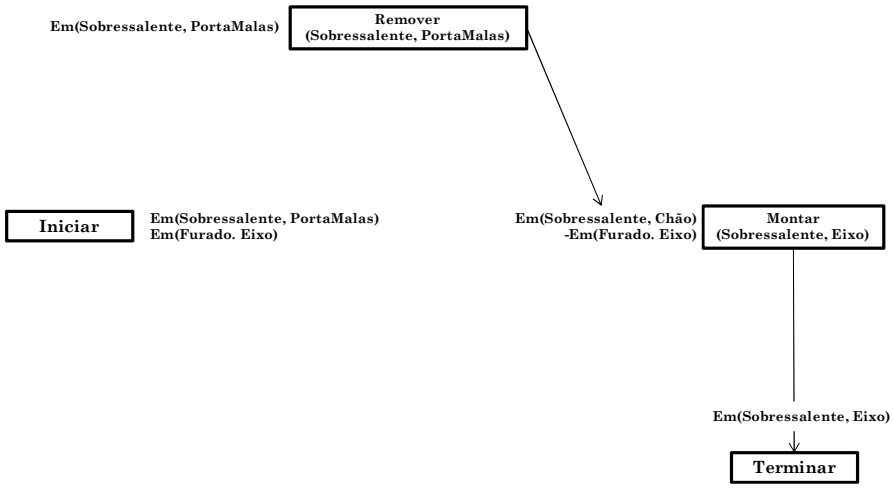
Iniciar(Em(Fundo,Eixo) ^ Em(Sobressalente, PortaMalas))
Objetivo(Em(Sobressalente,Eixo))
Ação(Remover(Sobressalente, PortaMalas),
PRECOND: Em(Sobressalente, PortaMalas)
EFFECT: -Em(Sobressalente, PortaMalas) ^ Em(Sobressalente, Chão))
Ação(Remover(Furado, Eixo),
PRECOND: Em(Furado, Eixo)
EFFECT: -Em(Furado, Eixo) ^ Em(Furado, Chão))
Ação(Montar(Sobressalente, Eixo),
PRECOND: Em(Sobressalente, Chão) ^ -Em(Furado, Eixo)
EFFECT: -Em(Sobressalente, Chão) ^ Em(Sobressalente, Eixo))
Ação(DeixarDuranteNoite,
PRECOND:
EFFECT: -Em(Sobressalente, Chão) ^ Em(Sobressalente, Eixo) ^ Em(Sobressalente, PortaMalas) ^ -Em(Furado, Chão) ^ - Em(Furado, Eixo))

A solução para este problema começa com o plano inicial que contém apenas as ações Iniciar e Terminar com suas precondições e efeitos. A busca da solução deve ser feita utilizando a busca para trás como visto anteriormente.

Escolhe-se a única precondição aberta existente Em(Sobressalente, Eixo) de Terminar, logo a única ação aplicável é Montar(Sobressalente, Pneu).

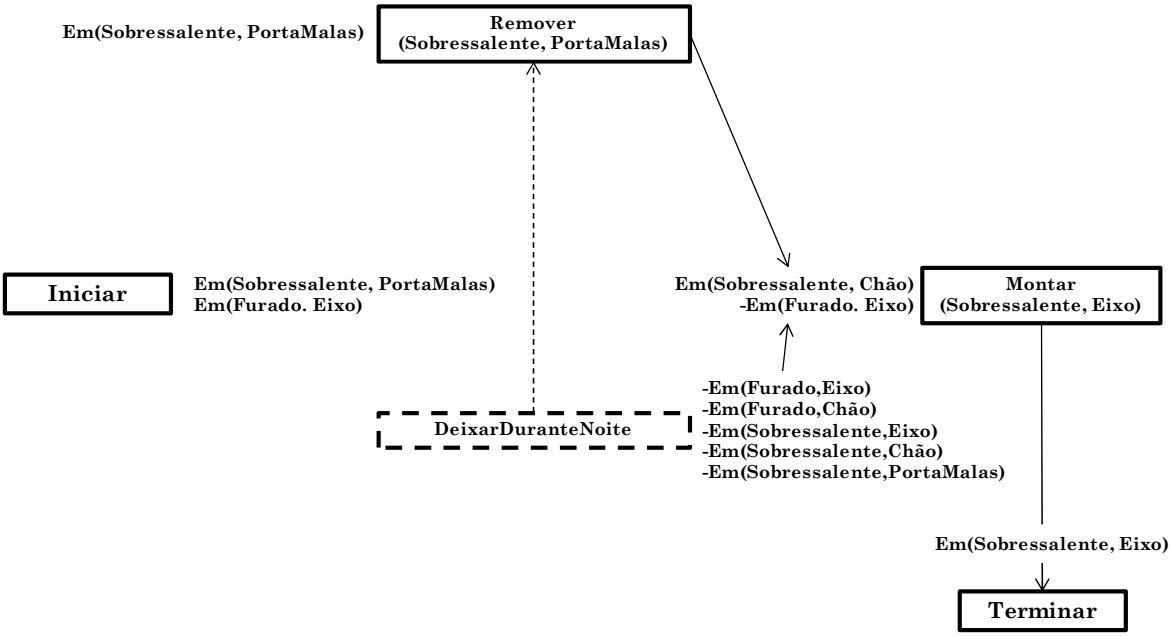


Escolhe-se uma das duas precondições abertas existentes, por exemplo, escolhendo Em(Sobressalente, Chão) a única ação aplicável existente é Remover(Sobressalente, PortaMalas)



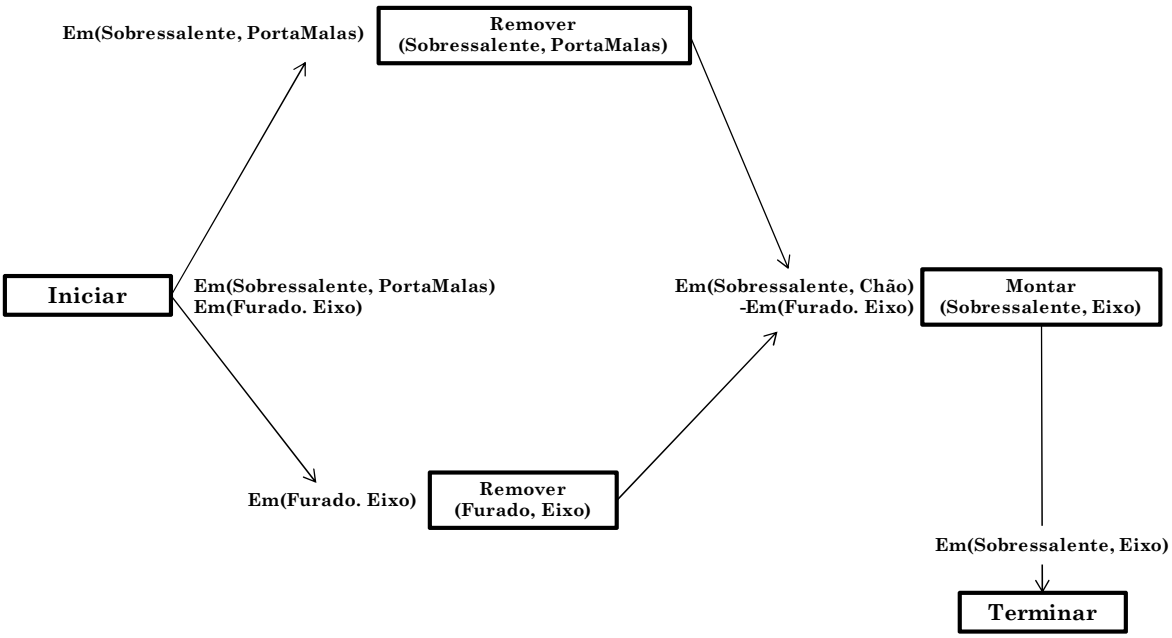
Seleciona-se a outra precondição aberta de Montar(Sobressalente, Eixo). Neste caso existem duas ações que podem ser tomadas DeixarDuranteNoite e Remover(Furado, Eixo). A efeito didático vamos escolher a primeira.

Nota-se que DeixarDuranteNoite também possui o efeito – Em(Sobressalente, Chão), o que significa que ela está em conflito com o vínculo causal. Para solucionar este conflito a única solução é que a ação DeixarDuranteNoite deve acontecer antes de Remover(Sobressalente, PortaMalas). Entretanto a precondição de Remover(Sobressalente, PortaMalas) está em conflito com o efeito de DeixarDuranteNoite mostrando que esta ação não pode ser realizada, sendo então necessário voltar para o estado anterior.



Dessa vez seleciona-se a ação Remover(Furado, Eixo) para a precondição aberta – Em(Furado, Eixo)

Em seguida seleciona-se uma das precondições abertas existentes. Para ambas a ação Iniciar pode alcançá-las.



4.3 Variáveis não acopladas

Quando representações de ações de primeira ordem possuem variáveis. Estas podem estar não acopladas permitindo que algumas decisões sejam adiadas. Por exemplo para a ação:

Ação(Mover(A,x,B),

PRECOND: Sobre(A,x) ^ Livre(A) ^ Livre(B)

EFFECT: Sobre(A,B) ^ Livre(x) ^ -Sobre(A,x) ^ -Livre(B))

A variável x está desacoplada. Ou seja, move-se A de algum lugar, sem ainda dizer o motivo para B. Esse é outro exemplo de compromisso mínimo aonde a decisão pode ser retardada até o momento que uma outra ação especifique o valor de x.

Isso permite estender a representação dos planos com o conjunto restrições de desigualdade. Por exemplo, se existir uma ação M2 com efeito –Sobre(A, z) então M2 estará em conflito se z for igual a B. Logo, z deve ser diferente de B.

4.4 Heurísticas

Para plano de ordem parcial a heurística mais óbvia é contar o número de precondições abertas distintas. Uma melhoria seria subtrair o número de precondições abertas que correspondem a literais no estado Início.

Entretanto, essa heurística superestima o custo quando existem ações que alcançam vários subobjetivos e subestimam o custo quando existem iterações negativas entre passos do plano. Invalidando assim essa heurística.

Uma outra heurística mais eficiente é fornecida pelo algoritmo da variável mais restrita. Essa heurística é mais vantajosa em duas situações. A primeira quando uma condição aberta não puder ser atingida por nenhuma ação, poupando muito trabalho. A segunda quando uma condição aberta só pode ser atingida de uma única maneira, logo ela deve ser selecionada adiantando a decisão inevitável. Desta forma outras restrições são inseridas para as ações ainda não tomadas.

5 Conclusão

O conjunto STRIPS e POP torna-se uma ferramenta poderosa para solucionar problemas chamados clássicos. Ainda que o problema em si seja de uma natureza relativamente mais simples que os demais, ainda é um desafio para muitos dos problemas enfrentados pela inteligência artificial.

Não é a única, nem a melhor ferramenta, já que outras linguagens, como a ADL, também tem sua própria utilidade dentro destes problemas. Porém, um bom conjunto para se conhecer quando um problema de natureza clássica aparecer.

6 Bibliografia

1. **Russell, Stuart e Norvig, Peter.** *Inteligência Artificial.* s.l. : Editora Campus, 2004.