FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

INTELIGÊNCIA ARTIFICIAL

2013-2014

Relatório:

TP2– Prolog

15 de Dezembro de 2013

Docente: Jorge Cruz

Turno Prático: P4

Realizado por:

Ricardo Cruz, nº 34951

Ricardo Gaspar, nº 42038

Luís Silva, nº 34535

Introdução

O trabalho enunciado requer a implementação de um planeador em espaços de estados na linguagem Prolog. Este está faseado em três partes distintas.

A primeira incide na formulação de um problema de planeamento usando operadores *STRIPS*. O problema a formular é o de um robot que se encontra num mundo de três salas e que em cada uma existe um objecto. O robot tem de apanhar todos os objectos e trazê-los para uma das salas.

Na segunda parte pretende-se que, dado um plano e um estado inicial, se construa um predicado que mostre a execução do plano verificando a sua consistência.

Para a última fase é necessário que sejam gerados planos para resolver um problema. Pretende-se que seja utilizado um algoritmo de planeamento progressivo e que os planos contenham o menor número possível de acções de forma a satisfazer os objectivos.

Modelação do problema

Para a primeira parte do trabalho foram utilizados operadores *STRIPS* de modo a melhor formular o problema proposto. O problema enuncia que o robot se encontra inicialmente na sala *s1* e tem de trazer todos os objectos (*b1, b2, b3*) existentes nas três salas (*s1, s2, s3*) para a sala *s3*. Para se deslocar entre as salas o robot tem de atravessar portas, caso estas existam. Neste problema apenas existem portas entre os pares de salas *(s1 ,s2*) *e* (*s2, s3*).

Neste seguimento, as acções que modelam o problema encontram-se apresentadas em baixo.

acção (nome: agarrar (O),
condições: [onde (S), livre, existe (O,S)],
efeitos: [-livre, -existe (O,S), ocupado (O)],
restrições: [S\==O] **).**

acção (nome: largar (O),
condições: [onde (S), ocupado (O)],
efeitos: [-ocupado (O), livre, existe (O,S)],
restrições: [S\==O] **).**

acção (nome: deslocar (A, B),
condições : [onde (A), porta (A,B)],
efeitos: [-onde (A), onde (B)],
restrições: [A\==B] **).**

O estado inicial e a lista de objectivos caracterizam-se pelo seguintes factos:

inicial ([onde(s1), livre, existe(b1,s1), existe(b2,s2), existe(b3,s3), porta(s1,s2), porta(s2,s3), porta(s3,s2), porta(s2,s1)]**).**

objectivos ([existe(b1,s3), existe(b2,s3), existe(b3,s3)]**).**

Definição dos predicados

membro/2 – Verifica se uma dada lista está contida dentro de outra.
naomember/2 – Verifica se um dado elemento não pertence a uma lista.
remover/3 – Dado um elemento verifica se este se encontra numa dada lista e remove-o desta construindo uma nova sem o mesmo.
aplicarrestricoes/1. – Aplica um conjunto de restrições. Para tal é utilizado um predicado do sistema *ground* que verifica se as variáveis estão instanciadas.
aplicarrestricoes/2 – Faz o mesmo que o anterior, mas com a particularidade de devolver uma lista das restrições que não se puderam aplicar. Esta serve para que se possam aplicar mais tarde.
estadoseguinte/3 – Constrói o estado seguinte retirando os efeitos negativos e adicionando os positivos ao estado actual.
satisfiedGoal/1 – Verifica se o objectivo de um problema foi atingido comparando a lista do estado actual com a lista dos objectivos.
satisfiedGoal/2 – Faz o mesmo que anterior, no entanto aplica as restrições pendentes.
writeExec/1 – Escreve o conteúdo de uma lista no ecrã, distinguindo por passos as acções e os estados intermédios.
testPlan – Testa e executa um determinado plano. Existem essencialmente dois predicados, sendo um para a segunda fase e outro para a terceira fase do trabalho. Este último tem a especificidade de ter um limite número de acções por forma a garantir que se construam planos com o menor número de acções.
plano/1 – Constrói um plano de modo a este atingir o objectivo de um problema no menor número de acções.
.

Testes e Resultados

Problema do robot

Estado inicial: ([onde(s1), livre, existe(b1,s1), existe(b2,s2), existe(b3,s3), porta(s1,s2), porta(s2,s3), porta(s3,s2), porta(s2,s1)])
Objectivos: ([existe(b1,s3), existe(b2,s3), existe(b3,s3)])

Execução:

```
plano(L).
L = [agarrar(b1), deslocar(s1, s2), deslocar(s2, s3), largar(b1), deslocar(s3, s2), agarrar(b2), deslocar(s2, s3), largar(b2)] ;
false.
```

Problema dos Sapatos

Estado Inicial: []
Objectivos: ([sapato(esq),sapato(dir)])

Execução:

```
plano(L).
L = [calcarMeia(dir), calcarSap(dir), calcarMeia(esq), calcarSap(esq)] ;
L = [calcarMeia(esq), calcarSap(esq), calcarMeia(dir), calcarSap(dir)] ;
L = [calcarMeia(esq), calcarMeia(dir), calcarSap(dir), calcarSap(esq)] ;
L = [calcarMeia(dir), calcarMeia(esq), calcarSap(esq), calcarSap(dir)] ;
L = [calcarMeia(dir), calcarMeia(esq), calcarSap(dir), calcarSap(esq)] ;
L = [calcarMeia(esq), calcarMeia(dir), calcarSap(esq), calcarSap(dir)] ;
false.
```

Problema das Compras

Estado Inicial: ([at(home),sell(super,banana),sell(hws,drill),sell(super,milk)])
Objectivos: ([have(milk),have(drill),have(banana),at(home)])

Execução:

```
plano(L).
L = [go(home, super), buy(banana), buy(milk), go(super, hws), buy(drill), go(hws, home)] ;
L = [go(home, hws), buy(drill), go(hws, super), buy(banana), buy(milk), go(super, home)] ;
L = [go(home, hws), buy(drill), go(hws, super), buy(milk), buy(banana), go(super, home)] ;
L = [go(home, super), buy(milk), buy(banana), go(super, hws), buy(drill), go(hws, home)] ;
false.
```

Conclusão

O presente trabalho permitiu aos alunos adquirir conhecimentos como a formulação de problemas utilizando os operadores STRIPS, a programação utilizando a linguagem Prolog e o seu funcionamento.

Algumas das principais dificuldades sentidas foram a adaptação à linguagem e ao modo de raciocínio dos problemas visto ser uma linguagem declarativa e, consequentemente, bastante diferente das linguagens procedimentais. Estas dificuldades sentiram-se, sobretudo, nas segunda e terceira fases do trabalho.

Em relação à implementação, é necessário mencionar que na geração do plano foi utilizado o planeamento progressivo. Este foi escolhido por ser de mais simples compreensão e realização.
Na sua elaboração utilizou-se um predicado responsável pela procura de um plano que satisfaça os objectivos com o menor número de acções. Para tal, começa-se por testar com zero acções e incrementa-se sucessivamente este número até serem atingidos os objectivos.

Apesar de tudo esta abordagem não é a mais eficaz. Segundo os conhecimentos adquiridos durante as aulas, é possível afirmar que a melhor forma para resolver problemas deste género é o planeamento regressivo. Este consiste em partir dos objectivos e ir deduzindo as acções necessárias para se atingir o estado inicial.