Escola de Ciências e Tecnologia

Curso: Engenharia Informática

Disciplina: Inteligência Artificial

Professora: Irene Rodrigues

Cálculo de Situações

Quarto Trabalho Prático

Trabalho Elaborado Por:

Marlene Oliveira Nº 25999

Pedro Mateus Nº 26048

João Aiveca Nº 26175

Ano Lectivo 2011 / 2012

1. Vocabulário:

i. Condições:

valor(X,Y) -> é verdade para um par X (que é uma lista de registos) Y (que é uma lista de valores).

X = [a,b,c,d,e]

Y = [va,vb,vc,vd,ve] no estado 0

Y = [vb,va,vb,vd,ve] no estado 1

Y = [vb,va,va+vb,vc,va] no estado 2

ii. Acções:

afectar_r(R1,R2)

- com R1 = {a,b,c,d,e}

- com R2 = {a,b,c,d,e}

somar_r(R1,R2,R3)

- com R1 = {a,b,c,d,e}

- com R2 = {a,b,c,d,e}

- com R3 = {a,b,c,d,e}

2.

a.

i. Consequências positivas:

- o registo X passa a ter o valor do registo Y se, após uma acção afectar_r, Vx = Vy.

- o registo Z passa a ter o valor da soma dos valores dos registos X e Y se, após uma acção soma_r, Vz = Vx + Vy.

ii. Leis da Inércia:

- se o valor do registo X já é igual ao do registo Y, entao nada é alterado quando é efectuada uma acção afectar_r(X,Y)

- se o valor do registo Z já é igual à soma dos valores dos registos X e Y, então nada é alterado quando é efectuada uma acção somar_r(X,Y).

- afectar_r(X,X).

b. e c.

Dado não existirem percepções relevantes, dizemos à knowledge base o estado de uma jogada.

tell(kb,A,S):- asserta(h(valor([a,b,c,d,e],A),S)).

Perguntamos à KB qual a acção a executar para um determinado estado dado como argumento.

ask(kb,accao(afectar_r(P),S)).

ask(kb,accao(somar_r(P),S)).

Definimos estado inicial e um objectivo a atingir. As escolhas da KB vão-se basear no objectivo (segundo predicado s2).

h(valor([a,b,c,d,e],[va,vb,vc,vd,ve]),0).

h(valor([a,b,c,d,e],[vb,va,va+vb,vc,va]),s2).

Modelamos aqui as consequências das acções:

h(valor(X,Y),r(somar_r(valor(U,W)),S)).

h(valor(X,Y),r(afectar_r(valor(U,W)),S)).

Verificamos a vitória – o atingir do estado final – do jogador.

h(ganhou,somar_r(Y)).

h(ganhou,afectar_r(Y)).

As funções auxiliares

val(Var1,Var2,R,afect,S).

val(Var1,Var2,R,somar,S).

escolhejogada([a,b,c,d,e], [A,B,C,D,E], Arg1,Arg2,Jogada).

valaux([a,b,c,d,e], [A,B,C,D,E], a,a, [A,B,C,D,E],afect).

valaux([a,b,c,d,e], [A,B,C,D,E], a,a, [A+A,B,C,D,E],somar).

Servem para calcular a jogada que mais se aproxima do estado final. De notar que é possível ocorrerem jogadas que não levem à solução, caso determinadas variáveis importantes desapareçam devido a um afectar_r.

3. e 4.

Resultados da implementação:

Estado 2 – [vb,va,va+vb,vc,va]

Queries:

o que? [va,vb,vc,vd,ve].

Faz afectar_r([va,va,vc,vd,ve])

Exemplificamos o problema das variáveis se perderem. Aproximamo-nos da solução neste passo – temos va no segundo argumento – mas perdemos vb, o que torna impossível atingir o resultado final.

o que? [vb,va,va,vc,va].

Faz somar_r([vb,va,va+vb,vc,va])

Execution aborted.

Neste caso, temos a execução correcta do algoritmo.

o que? [va,ve,va,vb,vc].

Faz afectar_r([va,va,va,vb,vc])

Neste caso, temos uma outra execução correcta do algoritmo.

Estado 1 – [vb,va,vb,vd,ve]

Queries:

o que? [va,vb,vc,vd,ve].

Faz afectar_r([va,va,vc,vd,ve])

Mesma situação que no rimeiro exemplo do estado 2; perde-se vb.

o que? [va,va,vb,vd,ve].

Faz afectar_r([vb,va,vb,vd,ve])

% Execution Aborted

Neste caso atingimos o estado final.

o que? [ve,va,vb,vc,vd].

Faz afectar_r([ve,va,vb,vc,ve])

Neste caso demonstramos a ordem de procura de um resultado: afectar em primeiro lugar, e ela ordem a>b>c>d>e.