Rappresentazione della Conoscenza

Esercitazione 7

Sommario

- ♢ **Rappresentazione di domini dinamici**
- ♢ **Implementazione in PROLOG**

Robot Geologo

Si vogliono rappresentare le azioni di un robot geologo, che deve analizzare delle rocce indicate.

Il robot è in grado di muoversi su ruote per arrivare alla sua destinazione. E' in grado di prelevare una roccia, analizzarla e depositarla in un contenitore, posto all'interno del robot. Ogni roccia avrà un identificativo.

Vocabolario

Azioni

- $vaiVerso(r)$ - Il robot si muove verso la roccia r;
- $preleva(r)$ - Il robot preleva la roccia r;
- $analizza(r)$ - Il robot analizza la roccia r;
- $depositaInContenitore(r)$ - Il robot deposita la roccia r nel contenitore;

Vocabolario

Fluenti

- $InMano(r, s)$ - Il robot ha in mano la roccia r nella situazione s;
- $Vicino(r, s)$ - Il robot è vicino alla roccia r nella situazione s;
- $Analizzata(r, s)$ - La roccia r è stata analizzata nella situazione s;
- $InContenitore(r, s)$ - La roccia r è nel contenitore nella situazione s;

Assiomi di precondizione

$Poss(vaiVerso(r), s) \leftrightarrow \top$

$Poss(preleva(r), s) \leftrightarrow Vicino(r, s) \wedge \forall r_1 \ \neg InMano(r_1, s)$

$Poss(analizza(r), s) \leftrightarrow InMano(r, s) \wedge \neg Analizzata(r, s)$

$Poss(depositaInContenitore(r), s) \leftrightarrow InMano(r, s) \wedge Analizzata(r, s)$

Assiomi di stato successore

$$InMano(r, do(a, s)) \leftrightarrow a = preleva(r) \vee$$
$$\quad InMano(r, s) \wedge a \neq depositaInContenitore(r)$$
$$Vicino(r, do(a, s)) \leftrightarrow a = vaiVerso(r) \vee$$
$$\quad Vicino(r, s) \wedge \forall r_1 a \neq vaiVerso(r_1)$$
$$Analizzata(r, do(a, s)) \leftrightarrow a = analizza(r) \vee Analizzata(r, s)$$
$$InContenitore(r, do(a, s)) \leftrightarrow a = depositaInContenitore(r) \vee$$
$$\quad InContenitore(r, s)$$

Situazione iniziale e pianificazione

Situazione Iniziale

$$S_0 = \{roccia(r_1), roccia(r_2), base(b_1)\}$$

Obbiettivo

$$Goal(s) \leftrightarrow InContenitore(r_1, s) \wedge Vicino(b_1, s)$$

Pianificazione

$$\mathcal{D} \models executable(do([a_1, \ldots, a_n], S_0) \wedge Goal(do([a_1, \ldots, a_n], S_0)))$$

Il mondo dei blocchi

Il mondo dei blocchi è un tipico dominio di IA. Il mondo è rappresentato da un insieme di blocchi sovrapposti. L'agente può sollevare un blocco libero (in cima ad una pila) e depositarlo sul tavolo o su di un'altra pila.

Fluenti

Il dominio è rappresentato dai fluenti seguenti:

- $clear(x, s)$: il blocco x non ha nessun blocco al di sopra di lui (libero) nella situazione s;
- $on(x, y, s)$: il blocco x è immediatamente sopra il blocco y nella situazione s;
- $onTable(x, s)$: il blocco x è sul tavolo nella situazione s.

Azioni

il dominio è rappresentato dalle azioni seguenti:

- $move(x, y)$: sposta il blocco x sul blocco y, se entrambi liberi;
- $moveToTable(x)$: muove il blocco x sul tavolo se è libero e non c'è già.

Precondizioni

- è possibile spostare un blocco su di un altro se entrambi sono liberi:

$$Poss(move(x, y), s) \equiv clear(x, s) \wedge clear(y, s) \wedge x \neq y$$

- è possibile spostare un blocco sul tavolo se è libero e non si trova già sul tavolo:

$$Poss(moveToTable(x), s) \equiv clear(x, s) \wedge \neg onTable(x, s)$$

Assiomi di stato successore

Per il mondo dei blocchi è più semplice definire direttamente gli assiomi di stato successore:

- $clear(x, do(a, s)) \equiv \exists y.((\exists z.a = move(y, z) \vee a = moveToTable(y)) \wedge$
  $on(y, x, s)) \vee clear(x, s) \wedge \neg\exists y.a = move(y, x)$

- $on(x, y, do(a, s)) \equiv a = move(x, y) \vee$
  $on(x, y, s) \wedge a \neq moveToTable(x) \wedge \neg\exists z.a = move(x, z)$

- $onTable(x, do(a, s)) \equiv a = moveToTable(x) \vee$
  $onTable(x, s) \wedge \neg\exists z.a = move(x, z)$

Stato Iniziale

Lo stato iniziale corrisponde alla situazione S_0 seguente:

$ontable(D, S_0), ontable(F, S_0), ontable(G, S_0);$
$on(U, A, S_0), on(A, B, S_0), on(B, C, S_0), on(C, D, S_0).;$
$on(V, E, S_0), on(E, F, S_0);$
$on(W, G, S_0);$
$clear(U, S_0), clear(V, S_0), clear(W, S_0).$

Implementazione in PROLOG

I domini dinamici si possono implementare in PROLOG. Per poter garantire la correttezza, è necessario che:

- la teoria possieda assiomi di nome unico
- una definizione per ogni predicato non-fluente ($\equiv$)
- fluenti relazionali: $F(\vec{x}, S_0) \equiv \Psi_F(\vec{x}, S_0)$

N.B. La seconda condizione non permette di utilizzare conoscenza incompleta

Traduzione in PROLOG

Per ottenere l'implementazione in PROLOG il dominio precedentemente presentato, occorre applicare:

- le trasformazioni di Lloyd-Topor (opz.)
- la semantica del completamento di Clark

Trasformazioni di Lloyd-Topor

- Serve per tradurre in forma normale le parti destre delle definizioni di Poss e fluenti (se già OK, non si applica)
- NB: per eliminare gli esistenziali potrebbe essere necessario introdurre un nuovo predicato

Teoria del dominio in Prolog

- non-fluenti: $\Theta_P(\vec{x}) \supset P(\vec{x})$
  al posto di $\Theta_P(\vec{x}) \equiv P(\vec{x})$
- fluenti in S_0: $\Psi_F(\vec{x}, S_0) \supset F(\vec{x}, S_0)$
  al posto di $\Psi_F(\vec{x}, S_0) \equiv F(\vec{x}, S_0)$
- assiomi precondizione: $\Pi_A(\vec{x}, s) \supset Poss(A(\vec{x}), s)$
  al posto di $Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s)$
- assiomi stato successore: $\Phi_F(\vec{x}, a, s) \supset F(\vec{x}, do(a, s))$
  al posto di $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$

Specificata dal programmatore GOLOG.

Trasformazioni di Lloyd-Topor - Precondizioni

$$(\mathbf{clear(x,s)} \wedge \mathbf{clear(y,s)} \wedge \mathbf{x \neq y}) \rightarrow \mathbf{Poss(move(x,y),s)}$$

$$(\mathbf{clear(x,s)} \wedge \neg\mathbf{onTable(x,s)}) \rightarrow \mathbf{Poss(moveToTable(x),s)}$$

Trasformazioni di Lloyd-Topor - StatoSucc

$$(\exists \mathbf{y}.((\exists \mathbf{z}.\mathbf{a} = \mathbf{move}(\mathbf{y},\mathbf{z}) \vee \mathbf{a} = \mathbf{moveToTable}(\mathbf{y})) \wedge \mathbf{on}(\mathbf{y},\mathbf{x},\mathbf{s})) \vee$$
$$\mathbf{clear}(\mathbf{x},\mathbf{s}) \wedge \neg\exists \mathbf{y}.\mathbf{a} = \mathbf{move}(\mathbf{y},\mathbf{x})) \rightarrow \mathbf{clear}(\mathbf{x},\mathbf{do}(\mathbf{a},\mathbf{s}))$$
$$(a = move(y,z) \vee a = moveToTable(y)) \wedge on(y,x,s)) \vee$$
$$clear(x,s) \wedge a \neq move(y,x)) \rightarrow clear(x,do(a,s))$$

$$(\mathbf{a} = \mathbf{move}(\mathbf{x},\mathbf{y}) \vee \mathbf{on}(\mathbf{x},\mathbf{y},\mathbf{s}) \wedge \mathbf{a} \neq \mathbf{moveToTable}(\mathbf{x}) \wedge$$
$$\neg\exists \mathbf{z}.\mathbf{a} = \mathbf{move}(\mathbf{x},\mathbf{z})) \rightarrow \mathbf{on}(\mathbf{x},\mathbf{y},\mathbf{do}(\mathbf{a},\mathbf{s}))$$
$$(a = move(x,y) \vee on(x,y,s) \wedge a \neq moveToTable(x) \wedge$$
$$a \neq move(x,z)) \rightarrow on(x,y,do(a,s))$$

Trasformazioni di Lloyd-Topor - StatoSucc

$$\mathbf{(a = moveToTable(x) \vee onTable(x,s) \wedge \neg \exists z.a = move(x,z) \rightarrow onTable(x, do(a,s))}$$

$$(a = moveToTable(x) \vee onTable(x,s) \wedge a \neq move(x,z) \rightarrow onTable(x, do(a,s))$$

Trasformazioni di Lloyd-Topor - StatoIniz

Lo stato iniziale precedente rispetta le condizioni di mondo chiuso.

Implementazione in PROLOG

```
poss(move(X,Y),S) :- clear(X,S), clear(Y,S), not(X = Y).
poss(moveToTable(X),S) :- clear(X,S), not(ontable(X,S)).

clear(X,do(A,S)) :- (A = move(Y,_Z);
                     A = moveToTable(Y)), on(Y,X,S);
                     clear(X,S), not(A = move(Y,X)).
on(X,Y,do(A,S)) :- A = move(X,Y) ;
                   on(X,Y,S), not(A = moveToTable(X)),
                              not(A = move(X,_Z)).
ontable(X,do(A,S)) :- A = moveToTable(X) ;
                      ontable(X,S), not(A = move(X,_Y)).

primitive_action(move(_X,_Y)).
primitive_action(moveToTable(_X)).
```

Implementazione in PROLOG: query

```
/* [moveToTable(u), moveToTable(a), move(b, a), move(c, b)] e' una
sequenza eseguibile? */

poss(moveToTable(u),s0), poss(moveToTable(a),do(moveToTable(u),s0))
poss(move(b, a),do(moveToTable(a),do(moveToTable(u),s0))),
poss(move(c, b),do(move(b,
a),do(moveToTable(a),do(moveToTable(u),s0)))).

/* Quale blocco e' sul tavolo prima e dopo la sequenza? */
ontable(X,s0). ontable(X,do(move(b,
a),do(moveToTable(a),do(moveToTable(u),s0)))).

/* Quale blocco si puo' muovere sul tavolo dopo la sequenza
[moveToTable(a), move(b, a)]? */
poss(moveToTable(X),do(moveToTable(a),do(move(a,b),s0))).
```

RobotGeologo-Implementazione in PROLOG

```
poss(vaiVerso(_X),_S).
poss(preleva(R),S):-vicino(R,S),not(inMano(_K,S)).
poss(analizza(R),S):-inMano(R,S),not(analizzata(R,S)).
poss(depositaInContenitore(R),S):-inMano(R,S), analizzata(R,S).

inMano(R,do(A,S)):-A=preleva(R);
inMano(R,S),not(A=depositaInContenitore(R)).
vicino(R,do(A,S)):-A=vaiVerso(R); vicino(R,S), not(A=vaiVerso(_K)).
analizzata(R,do(A,S)):-A=analizza(R); analizzata(R,S).
inContenitore(R,do(A,S)):- A=depositaInContenitore(R);
                           inContenitore(R,S).

roccia(r1). roccia(r2). base(b1).

goal(S):- inContenitore(r1,S),vicino(b1,S).
```

RobotGeologo-PROLOG query

```
/*La sequenza [vaiVerso(r1), preleva(r1), analizza(r1),
depositaInContenitore(r1), vaiVerso(b1)] e' eseguibile?*/
poss(vaiVerso(r1),s0),poss(preleva(r1),do(vaiVerso(r1),s0)),
poss(analizza(r1),do(preleva(r1),do(vaiVerso(r1),s0))),
poss(depositaInContenitore(r1),do(analizza(r1),do(preleva(r1),
                                        do(vaiVerso(r1),s0)))),
poss(vaiVerso(b1),do(depositaInContenitore(r1),do(analizza(r1),
                          do(preleva(r1),do(vaiVerso(r1),s0))))).

/*Che cosa c'e' nel contenitore dopo la sequenza?*/
inContenitore(X,do(vaiVerso(b1),do(depositaInContenitore(r1),
     do(analizza(r1),do(preleva(r1),do(vaiVerso(r1),s0)))))).

/*Che cosa posso analizzare dopo la sequenza [vaiVerso(r2),
preleva(r2)]? */
poss(analizza(X),do(preleva(r2),do(vaiVerso(r2),s0))).
```