Un sentito ringraziamento ed un ricordo per
Raymond Reiter

Intelligenza Artificiale

Lezione 10

Sommario

- Il calcolo delle situazioni
- Reiter 3 (escluso 3.2.1-3.2.6 e 3.2.8)

Calcolo delle situazioni

Linguaggio del primo ordine con uguaglianza e con più sorti:

- $azione$
- $oggetto$
- $situazione$

Il linguaggio contiene:

– $do(a, s)$, che fornisce come valore una situazione;
– $<$ ($\leq$) che si applica a coppie di situazioni,
– la costante S_0 di sorte $situazione$

Fluenti

Un *fluente* si differenzia da un simbolo normale perché il suo valore può variare al variare della situazione

$P(x_1, \ldots, x_n, s)$;

$f(x_1, \ldots, x_n, s)$.

Esempi: $Su(x, y, s)$, $Su(x, y, s')$,
$colore(x, s) = rosso$, $colore(x, s') = rosso$.

$Poss(a, s)$ è un fluente il cui significato intuitivo è che l'azione a è possibile nella situazione s.

Formalizzazione delle azioni

- **Precondizioni**: quando un'azione può esser eseguita, definite da $Poss(a, s)$
  $Poss(ripara(w, x), s) \rightarrow hacolla(w, s) \wedge rotto(x, s)$
- **Effetti**: come si caratterizza la situazione $do(a, s)$ ottenuta tramite l'esecuzione di a
  $fragile(x, s) \rightarrow rotto(x, do(lascia(r, x), s))$

Problemi (aspetti) della rappresentazione

- qualificazione delle precondizioni (qualification problem)
- specifica degli effetti (frame problem: frame = cornice, problema del cambiamento)
- assiomi statici e ramificazione degli effetti (ramification problem)

Qualificazione (delle precondizioni)

Condizione necessaria per la riparazione

$Poss(ripara(w, x), s) \rightarrow hacolla(w, s) \wedge rotto(x, s)$

ma è anche sufficiente?

$hacolla(w, s) \wedge rotto(x, s) \rightarrow Poss(ripara(w, x), s)$

Basta avere la colla per aggiustare un oggetto rotto?
La colla potrebbe essere secca, inadatta per il materiale dell'oggetto, potrebbe risultare impossibile aprire il barattolo ...

Soluzioni al problema della qualificazione

1. Si assume che le precondizioni siano necessarie e sufficienti (p.e. nel calcolo delle situazioni) In questo modo si esclude la possibililità che esistano impedimenti all'esecuzione dell'azione.
2. Occorre fare un'ipotesi del tipo

   "a meno che non ci siano impedimenti"

   che porta ad una forma di ragionamento non monotono.

Frame problem

Specifica "compatta" di ciò che non cambia.

Effetto di $sposta(b_1, tavolo)$ non cambia il colore di b_1, e nemmeno quello degli altri blocchi, e del tavolo!!!!

Quindi tra gli effetti di $sposta$ occorre aggiungere

$\forall x, b, t, s. colore(b, x, do(sposta(b, t), s)) = colore(b, x, s)$

E poi, c'è il peso, la forma, la posizione ....

Frame problem

In totale ci sono **2 x A x F** assiomi di frame.

Rappresentazione degli assiomi di frame

Deduzione con gli assiomi di frame

Persistenza

Per ogni azione ci sono molte proprietà che non vengono modificate dall'esecuzione, cioè **persistono**.

Soluzioni al Frame Problem (***):

1. assiomi di stato successore 1 per ogni fluente

   la logica rimane monotona, ma ci sono delle restrizioni (p.e. sull'uso degli assiomi "statici")
2. aggiungendo regole del tipo

   "tutto ciò che non viene specificato negli effetti persiste"

   la logica diventa non monotona

Soluzione (monotona) al frame problem

1. si considerano gli effetti positivi e negativi delle azioni per ogni fluente:
   $\gamma_F^+(x, a, s) \rightarrow F(x, do(a, s))$
   $\gamma_F^-(x, a, s) \rightarrow \neg F(x, do(a, s))$
2. Si assume la completezza causale: il fluente F è completamente definito dai due assiomi al punto 1
3. si fa l'ipotesi di nome unico per le azioni
4. si ottengono gli **assiomi di stato successore**:
   $F(x, do(a, s)) \equiv \gamma_F^+(x, a, s) \vee F(x, s) \wedge \neg\gamma_F^-(x, a, s)$

Nel calcolo delle situazioni si scrivono direttamente gli **assiomi di stato successore A + F**.

Ramificazioni degli effetti

È utile spesso specificare delle condizioni che valgono indipendentemente dalle situazioni: **assiomi statici**

Se la porta e la finestra sono aperte si crea corrente (indipendentente dalla particolare situazione).

$portaAperta \wedge finestraAperta \rightarrow corrente$

L'effetto dell'azione $apriPorta$ è $portaaperta$,
ma se $finestraAperta$ un effetto indiretto di $apriPorta$ è $corrente$.

Assiomi statici

Detti anche **vincoli/constraints** devono sempre essere verificato e ciò può creare dei problemi:

♢ con gli assiomi di stato successore: gli effetti indiretti devono essere considerati

♢ con la persistenza: normalmente $\neg corrente$ persiste quando viene eseguita $apriPorta$, ma non se $finestraAperta$.

Si hanno in genere diverse **ramificazioni** (situazioni possibili).

Specifica in Calcolo delle Situazioni

(Descrizione del dominio)

1. Gli assiomi di nome unico per le azioni (una – **unique name axioms**): sono gli assiomi che stabiliscono che tutte le azioni hanno nome unico.
2. La descrizione del dominio nello **stato iniziale** S_0.
3. Gli assiomi di **precondizione** (ap): costituiscono la descrizione delle condizioni per le quali un'azione è eseguibile, un assioma per ciascuna azione.
4. Gli assiomi di **stato successore** (ss): sono la descrizione delle leggi causali del dominio, un assioma per ciascun fluente.

Una descrizione dinamica: l'agente-fidanzato

Agente che compra dei fiori, li porta e li offre alla sua ragazza-agente, che quindi è contenta.

Azioni:

$andare\_da(x)$,
$andare\_via\_da(x)$
$comprare(x)$
$procurarsi\_soldi$,
$offrire(x, y)$

Una descrizione dinamica: l'agente-fidanzato 2

Simboli di predicato (oltre a $Poss$):

- $Dove(a, x)$ è vero se x è il luogo in cui l'azione a viene eseguita;
- $Sono\_vicino\_a(x, s)$ è vero se l'agente è vicino all'oggetto x nella situazione s;
- $Ho\_soldi\_per(x, s)$ è vero se l'agente ha soldi per comprare l'oggetto x nella situazione s;
- $Possiedo(x, s)$ è vero se l'agente ha l'oggetto x nella situazione s;
- $Contenta(x, s)$ è vero se l'agente x è contento nella situazione s.

Precondizioni

1. $Poss(procurarsi\_soldi, s)$
2. $Poss(andare\_da(x), s)$
3. $Poss(andare\_via\_da(x), s)$
4. $Poss(comprare(x), s) \leftrightarrow \exists y Dove(comprare(x), y) \wedge$
   $Sono\_vicino\_a(y, s) \wedge Ho\_soldi\_per(x, s)$
5. $Poss(offrire(x, y), s) \leftrightarrow Possiedo(x) \wedge Sono\_vicino\_a(y, s) \wedge$
   $x = fiori \wedge y = mia\_agente\_ragazza$

Assiomi di stato successore

6. $Ho\_soldi\_per(x, do(a, s))$
   $\leftrightarrow a = procurarsi\_soldi$
   $\vee(Ho\_soldi\_per(x, s) \wedge \neg\exists z\ a = comprare(z))$
7. $Sono\_vicino\_a(x, do(a, s)) \leftrightarrow a = andare\_da(x) \vee$
   $(Sono\_vicino\_a(x, s) \wedge a \neq andare\_via\_da(x))$
8. $Possiedo(x, do(a, s)) \leftrightarrow$
   $a = comprare(x) \vee (Possiedo(x, s) \wedge a \neq offrire(x, y))$
9. $Contenta(x, do(a, s)) \leftrightarrow a = offrire(fiori, x) \wedge$
   $x = mia\_agente\_ragazza$
   $\vee(Contenta(x, s) \wedge a \neq andare\_via\_da(x)))$

Mondo dei blocchi in SitCal

Fluenti

- $Libero(x, s)$ è un predicato a due argomenti; ci dice se un blocco x non ha altri blocchi sopra di sé in s;
- $Su(x, y, s)$ è un predicato a tre argomenti; ci dice se un blocco x si trova sopra (a contatto di) y in s;
- $Sul\_tavolo(x, s)$ è un predicato a due argomenti; ci dice se un blocco x si trova a contatto del tavolo in s;

Mondo dei blocchi in SitCal

Azioni:

- $muovere(x, y)$ ci dice che il blocco x viene mosso su y;
- $muovere\_sul\_tavolo(x)$ ci dice che il blocco x viene mosso sul tavolo.

Assiomi di precondizione

1 $Posso(muovere(x, y), s) \leftrightarrow Libero(x, s) \wedge Libero(y, s) \wedge x \neq y$
2 $Posso(muovere\_sul\_tavolo(x), s) \leftrightarrow Libero(x, s) \wedge \neg Sul\_tavolo(x, s)$

Assiomi di di nome unico

6 $muovere(x, y) \neq muovere\_sul\_tavolo(z)$
7 $muovere(x, y) = muovere(x', y') \rightarrow x = x' \wedge y = y'$
8 $muovere\_sul\_tavolo(x) = muovere\_sul\_tavolo(x') \rightarrow x = x'$

Assiomi di stato successore

3 $Libero(x, do(a, s)) \leftrightarrow$
$\exists y\ ((\exists z\ a = muovere(y, z) \vee a = muovere\_sul\_tavolo(y))$
$\wedge Su(y, x, s)) \vee$
$Libero(x, s) \wedge \neg(\exists y\ a = muovere(y, x))$

4 $Su(x, y, do(a, s)) \leftrightarrow a = muovere(x, y) \vee$
$Su(x, y, s) \wedge a \neq muovere\_sul\_tavolo(x) \wedge \neg\exists z\ a = muovere(x, z)$

5 $Sul\_tavolo(x, do(a, s)) \leftrightarrow a = muovere\_sul\_tavolo(x) \vee$
$Sul\_tavolo(x, s) \wedge \neg\exists y\ a = muovere(x, y)$

Situazione iniziale

$$Blocco(a) \wedge Blocco(b) \wedge Blocco(c) \wedge$$
$$Blocco(d) \wedge Blocco(e) \wedge Blocco(f)$$
$$Su(c, b, S_0) \wedge Su(b, a, S_0) \wedge Su(a, tavolo, S_0) \wedge$$
$$Su(e, d, S_0) \wedge Su(d, tavolo, S_0) \wedge Su(f, tavolo, S_0)$$

Dobbiamo aggiungere che i blocchi sono tutti e soli quelli elencati:

$$Blocco(x) \leftrightarrow x = a \vee x = b \vee x = c \vee x = d \vee x = e \vee x = f$$

Transazioni di una basi di dati

Lo stato (situazione) in cui si trova una base di dati può essere descritto come il risultato di una sequenza di transazioni (operazioni di aggiornamento).

Rappresentazione nel calcolo delle situazioni:

- i dati nella base di dati corrispondono a relazioni
- le operazioni di aggiornamento sono azioni
- tutto ciò che non è modificato da una transazione persiste

Calcolare lo stato della base di dati, consente di rispondere alle interrogazioni, dato la sequenza di transazioni e la base di dati iniziale.

Un esempio di modellazione di un database

Relazioni

1. $enrolled(st, course, s)$: st è iscritto a $course$ in s.
2. $grade(st, course, grade, s)$: il voto di st in $course$ è $grade$ in s.
3. $prerequ(pre, course)$: pre è un corso propedeutico a $course$.

Transazioni

1. $register(st, c)$: iscrive st a c.
2. $change(st, c, g)$: cambia il voto di st in c a g.
3. $drop(st, c)$: cancella st da c.

Database Iniziale

$enrolled(Sue, C100, S_0) \vee enrolled(Sue, C200, S_0)$,

$(\exists c) enrolled(Bill, c, S_0)$,

$(\forall p).prerequ(p, P300) \equiv p = P100 \vee p = M100$,

$(\forall p) \neg prerequ(p, C100)$,

$(\forall c).enrolled(Bill, c, S_0) \equiv c = M100 \vee c = C100 \vee c = P200$,

$enrolled(Mary, C100, S_0)$, $\neg enrolled(John, M200, S_0)$,
$grade(Sue, P300, 75, S_0)$.

Assiomi di precondizione delle transazioni

- Uno studente si può iscrivere ad un corso se ha avuto una votazione superiore a 50 in tutti i corsi prerequisito:

$$Poss(register(st, c), s) \equiv \{(\forall p).prerequ(p, c) \supset (\exists g).grade(st, p, g, s) \wedge g \geq 50\}.$$

- Si può cambiare il voto di uno studente sse ha un voto diverso dal nuovo voto:

$$Poss(change(st, c, g), s) \equiv (\exists g').grade(st, c, g', s) \wedge g' \neq g.$$

- Uno sudente si può cancellare da un corso sse è iscritto a quel corso:

$$Poss(drop(st, c), s) \equiv enrolled(st, c, s).$$

Assiomi di effetto per le Transazioni

$$\neg enrolled(st, c, do(drop(st, c), s)),$$

$$enrolled(st, c, do(register(st, c), s)),$$

$$grade(st, c, g, do(change(st, c, g), s)),$$

$$g' \neq g \supset \neg grade(st, c, g', do(change(st, c, g), s)).$$

Assiomi di stato successore

$$enrolled(st, c, do(a, s)) \equiv a = register(st, c) \ \vee$$
$$enrolled(st, c, s) \wedge a \neq drop(st, c),$$

$$grade(st, c, g, do(a, s)) \equiv a = change(st, c, g) \ \vee$$
$$grade(st, c, g, s) \wedge (\forall g')a \neq change(st, c, g').$$

Pianificazione deduttiva

Un problema di pianificazione deduttiva nel calcolo delle situazioni è definito da:

$$\mathcal{D} \models \exists s(executable(s) \wedge Goal(s))$$

Un piano è una sequenza di azioni la cui esecuzione porta il sistema dalla Sitazione Iniziale ad una situazione in cui è soddisfatta la condizione $Goal(s)$.

La condizione $executable(s)$ garantisce che per ogni passo del piano vengano verificate le precondizioni per l'esecuzione delle azioni.

Esempio: la ragazza contenta

Situazione Iniziale

$$S_0 = \{\neg Sono\_vicino\_a(mar, S_0), \neg Ho\_soldi\_per(x, S_0),$$
$$Possiedo(fiori, S_0), \neg Contenta(mar, S_0)\}$$

Obbiettivo

$$Goal(s) \iff Contenta(mar, s)$$

Piano

$s = do(offrire(fiori, mar), do(andare\_da(mar), S_0))$

$a_1 = andare\_da(mar)$

$a_2 = offrire(fiori, mar)$