DLV vs DISCOPLAN

Nicola Gatta
Matr. 35042

31 ottobre 2001

Elenco delle figure

1 Interfacciamento di DLV con DISCOPLAN

L'obiettivo è quello di verificare la possibilità di usare *DISCOPLAN* come "preprocessor" per *DLV* nell'ambito del mondo dei blocchi. Nella descrizione del dominio fornita con *DLV* sono presenti due clausole "fastidiose" o comunque esprimibili in altri modi (specificamente mediante l'utilizzo di *DISCOPLAN*). Oltre a questo è stata fatta una controprova su un risultato ottenuto mediante *DISCOPLAN*. In sostanza il dominio DLV risulta espresso in modo non minimale, infatti la clausola

:- on (B,B,T) ,

che sta ad indicare il fatto che un blocco non puo' essere appoggiato sotto se stesso, è inutile. I tests eseguiti eliminando la clausola dall'input di *DLV* hanno dimostrato che l'output non subiva alcun cambiamento.
Dunque una delle formule di tipo *Implicative Constraints* fornita da *DISCOPLAN* è di fatto inutile. Si tratta della seguente:

$$\forall x, y \qquad On(x\ y) \Longrightarrow Neq(x\ y)$$

La seconda clausola "fastidiosa" invece è fondamentale, senza di essa *DLV* non riesce ad eseguire alcuna inferenza.

$$\neg on(B, L1, T) \quad :- \quad on(B, L, T),\ location(L1),\ L <> L1$$

Questa clausola è fornita da *DISCOPLAN* nella categoria *Single-Valuedness Constraints* in questa forma

$$\forall x, y_1, y_2 \qquad On(x\, y_1) \wedge On(x\, y_2) \Longrightarrow Equal(y_1\, y_2)$$

Il problema sorge quando la si vuole esprimere nel "linguaggio" che *DLV* accetta in input. In un primo caso ho provato a esprimerla in una forma implicativa:

$$\neg on(A,B,T) \quad :- \quad on(A,C,T),\ location(B),\ location(C),\ B <> C$$

che, essenzialmente, è identica all'originale da sostituire. E tutto funziona a dovere.
In un secondo caso ho scelto di esprimere la formula come vincolo (*Costraint*) nella seguente forma:

$$:-\, on(A,B,T),\ on(A,C,T),\ location(B),\ location(C),\ C <> B$$

e *DLV* non è stato in grado di eseguire alcuna inferenza. Una riprova del fatto che *DLV* non "digerisce" una clausola corretta espressa in modo a lui non congeniale. Si prenda la prima clausola del dominio definito per *DLV*

$$move(B,L,T) \vee \neg move(B,L,T)\ :-\ block(B),\ location(L),\ time(T)$$

essa è una clausola di Horn, ed è pertanto equivalente alla seguente

$$move(B,L,T)\ :-\ block(B),\ location(L),\ time(T),\ move(B,L,T)$$

in cui il termine $\neg move(B,L,T)$ è stato spostato nell'altro membro dell'implicazione. Questo è chiaramente un esempio singolare, infatti dichiarare come "body" di un implicazione $move(B,L,T) \vee \neg move(B,L,T)$ è fuorviante.

2 Performance di *DLV*

2.1 Il problema

Sono state eseguite alcune prove, variando alcuni parametri nella riga di comando di *DLV* per valutare l'efficienza del programma in questione. Il dominio è sempre quello dei blocchi, nell'ambito "Large", in cui ci sono dieci blocchi posizionati in un dato modo (1): *DLV* deve pianificare le azioni di spostamento da eseguire per ottenere una configurazione finale data in input.

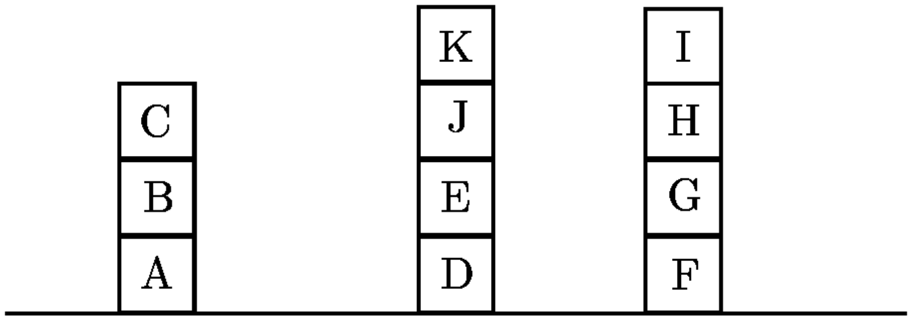

Figura 1: Situazione iniziale

Viene in seguito data in pasto a *DLV* la situazione iniziale e la definizione del dominio originale, ovvero quella fornita insieme al software.

Per come è stato definito il goal esiste un'unica situazione finale accettabile (2) (naturalmente raggiungibile in vari modi) :

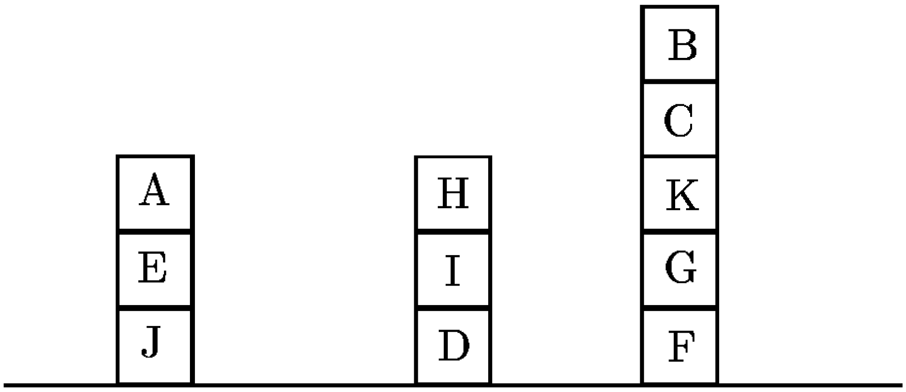

Figura 2: Situazione finale

2.2 Le prove e le prestazioni

Le prove sono state effettuate su un sistema *CPU: AMD Duron 800MHz, RAM: 256MB, sistema operativo: Linux 2.4.3-20.*

Sono state effettuate otto prove di funzionamento di *DLV*, dando ad esso forzandolo a generare piani con un numero di mosse variabile da otto a quindici.

Sono state misurati:

- **Positività** della ricerca del piano : *DLV* è in grado di generare un piano di n mosse?

- **Elapsed Time** tempo impiegato per generare il piano

- **RSS min** minima memoria reale fisica utilizzata da *DLV* (secondo la terminologia Linux)

- **RSS max** massima memoria reale fisica utilizzata da *DLV* (secondo la terminologia Linux)

- **CPU perc.** percentuale di CPU utilizzato da *DLV* in "user mode"

Si noti come i valori di RAM fisica utilizzata dipendono anche da altri fattori, quali situazione dei processi gestiti dal sistema operativo, istante in cui il programma che misura la memoria utilizzata va a campionare il valore di memoria utilizzata, etc...

In definitiva i risultati del test sono stati questi:

N° mosse	Positività	Elapsed time	RSS min (Kb)	RSS max(Kb)	CPU perc.
8	No	14.21sec			>99
9	Sì	23.34 sec	6136	6444	>99
10	Sì	2min 21sec	6704	7024	>99
11	Sì	30.83 sec	6972	7520	>99
12	Sì	1min 46sec	7580	8112	>99
13	Sì	1min 17sec	6704	9364	>99
14	Sì	1min 51sec	7404	9984	>99
15	Sì	3min 46sec	9588	10300	>99

2.3 Valutazione delle performance

Innanzitutto si nota che *DLV* non riesce a trovare un piano con meno di nove passi, ciò vuol dire che non è possibile raggiungeer il goal con un numero inferiore di passi. Ho provato anche a mano e non sono riuscito a trovarne alcuno in otto passi o meno. Si noti come il tempo impiegato per ottenere il piano non sia pefettamenet correlato linearmente con il numero dei passi del piano. Dall'analisi delle inferenze e delle mosse generate da *DLV* si capisce anche perchè l'utilizzo di memoria cresce col crescere del numero di mosse del piano. Però un piano con un numero di mosse superiore a 9 compie azioni "poco significative": nei piani con più di nove passi si aveva sempre una o più azioni di spostamento di un blocco A su di un altro blocco B seguita a breve da uno spostamento di A sul tavolo o su un altro blocco C. Come ulteriore verifica, il piano con nove mosse non muove i blocchi più di una volta ciascuno:

1. move(k,c,0)

2. move(j,table,1)

3. move(e,j,2)

4. move(i,d,3)

5. move(h,i,4)

6. move(k,g,5)

7. move(c,k,6)

8. move(b,c,7)

9. move(a,e,8)