CLINGO

Gli esercizi sviluppati utilizzando il paradigma ASP e lo strumento di sviluppo CLINGO sono stati due, il primo riguardante le cinque case, il secondo invece inerente al problema di pianificazione per il dominio del trasporto aereo di merci descritto nel capitolo 10.1 del Russell e Norvig.

Esercizio 1: CINQUE CASE

L’esercizio è stato svolto definendo inizialmente tutti i fatti del problema, cioè quelli riguardanti nazionalità, professione, animale, bevanda, colore e numero di case. Per i primi 5 tutti i termini possibili per ogni atomo sono stati raggruppati utilizzando il simbolo di pooling “;”, per il numero di case abbiamo invece definito un intervallo da 1 a 5 utilizzando il simbolo “..”.

Subito dopo abbiamo elencato le regole imposte dai 14 punti del problema. Infine sono stati aggiunti tutti i vincoli: il primo definisce il significato di adiacenza tra case (la distanza tra due case adiacenti è sempre pari ad 1), i secondi sono stati creati sfruttando gli aggregati (viene imposto che per ogni abitazione debbano apparire tutti gli atomi precedentemente definiti esclusivamente una ed una sola volta), ed i terzi sono vincoli di integrità (regole senza testa per permettere di trovare una soluzione corretta).

L’esecuzione di CLINGO è estremamente rapida come è possibile notare dall’output di seguito riportato:

Answer: 1

ownCol(3,rosso) ownNaz(4,inglese) ownNaz(2,spagnolo) ownAnim(2,cane) ownNaz(3,giapponese) ownProf(3,pittore) ownNaz(1,italiano) ownBev(5,te) ownCol(1,verde) ownBev(3,caffe) ownCol(2,bianco) ownProf(2,scultore) ownAnim(4,lumaca) ownCol(4,giallo) ownProf(5,diplomatico) ownNaz(5,norvegese) ownCol(5,blu) ownProf(4,violinista) ownBev(4,succo) ownProf(1,dottore) ownAnim(5,volpe) ownAnim(1,cavallo) ownAnim(3,zebra) ownBev(1,latte) ownBev(2,aranciata)

SATISFIABLE

Models : 1+

Calls : 1

Time : 0.061s (Solving: 0.05s 1st Model: 0.00s Unsat: 0.00s)

CPU Time : 0.031s

Esercizio 2: PIANIFICAZIONE SU DOMINIO AIR CARGO

L’esercizio è stato creato seguendo la struttura degli esempi proposti presenti su moodle.

Innanzitutto, per mezzo di 3 regole, sono state definite le azioni di *load*, *unload*, *fly* che possono essere compiute nel dominio; a queste è stata aggiunta una regola definita utilizzando un aggregato, per imporre che ci sia al più una azione per stato, anzi più specificatamente che ci sia esattamente una azione per stato. In seconda battuta sono state definite le regole che rappresentano gli effetti delle azioni. Queste regole impongono vincoli di mutex di modo che azioni diverse per ogni stato siano tra loro mutuamente esclusive. A questo punto abbiamo aggiunto le precondizioni, che definiscono quando è possibile applicare una determinata azione ad uno specifico stato. L’ultimo step per completare il pianificatore è riguardato l’aggiunta dei vincoli di persistenza per mezzo di regole non monotone. Ciò al fine di garantire che un letterale negato (o non negato) rimanga tale anche nello stato successivo se nello stato corrente nessuna azione interviene ad apportare su di esso modifiche. Una volta definito il pianificatore, abbiamo aggiunto tutti i fatti riguardanti il dominio, quindi numero di aerei (intervallo definito tramite “..”), aeroporti e cargo (entrambi raggruppati utilizzando “;”), posizionamento degli aerei e dei cargo nei vari aeroporti, specifica di un particolare goal. Di seguito riportiamo gli output di alcuni degli esperimenti eseguito e le conseguenti valutazioni:

* **Esperimento 1 (10 aerei, 3 cargo, 3 aeroporti, stati 6)**

Answer: 1

occurs(load(c1,1),0) occurs(load(c0,10),1) occurs(fly(1,torino,roma),2) occurs(fly(10,torino,catania),3) occurs(unload(c1,1),4) occurs(unload(c0,10),5)

SATISFIABLE

Models : 1+

Calls : 1

Time : 0.059s (Solving: 0.03s 1st Model: 0.01s Unsat: 0.00s)

CPU Time : 0.031s

* **Esperimento 2 (10 aerei, 3 cargo, 3 aeroporti, stati 31)**

Answer: 1

occurs(load(c1,9),3) occurs(load(c1,7),15) occurs(load(c1,6),23) occurs(load(c0,10),26) occurs(fly(3,catania,torino),0) occurs(fly(9,roma,torino),1) occurs(fly(3,torino,roma),2) occurs(fly(8,catania,torino),4) occurs(fly(7,catania,roma),5) occurs(fly(6,roma,torino),6) occurs(fly(5,roma,catania),7) occurs(fly(5,catania,torino),8) occurs(fly(5,torino,catania),9) occurs(fly(5,catania,torino),11) occurs(fly(7,roma,torino),12) occurs(fly(5,torino,catania),13) occurs(fly(5,catania,torino),14) occurs(fly(2,catania,torino),16) occurs(fly(1,torino,catania),17) occurs(fly(5,torino,roma),18) occurs(fly(5,roma,torino),19) occurs(fly(2,torino,catania),20) occurs(fly(5,torino,roma),21) occurs(fly(1,catania,torino),24) occurs(fly(5,roma,torino),25) occurs(fly(10,torino,catania),27) occurs(fly(6,torino,roma),28) occurs(unload(c1,9),10) occurs(unload(c1,7),22) occurs(unload(c0,10),29) occurs(unload(c1,6),30)

SATISFIABLE

Models : 1+

Calls : 1

Time : 0.421s (Solving: 0.39s 1st Model: 0.28s Unsat: 0.00s)

CPU Time : 0.421s

* **Esperimento 3 (10 aerei, 5 cargo, 7 aeroporti, stati 6)**

Answer: 1

occurs(load(c1,10),0) occurs(load(c0,1),3) occurs(fly(10,torino,roma),1) occurs(

fly(1,torino,catania),4) occurs(unload(c1,10),2) occurs(unload(c0,1),5)

SATISFIABLE

Models : 1+

Calls : 1

Time : 0.069s (Solving: 0.02s 1st Model: 0.00s Unsat: 0.00s)

CPU Time : 0.062s

Ciò che possiamo affermare osservando l’output degli esperimenti 1 e 3 è che a prescindere dal numero di aerei, cargo o aeroporti che aggiungiamo al problema, se questi non sono inclusi nel goal, gli answer set ritornati saranno comunque gli stessi e l’ordine di grandezza dei tempi di calcolo sarà praticamente uguale.

Questo è appunto il caso dell’esperimento 3, in cui sia il numero di cargo sia il numero di aeroporti sono maggiori rispetto a quelli dell’esperimento 1, ma la soluzione resta comunque uguale al netto di possibili variazioni dovute alla scelta di uno o di un altro aereo per effettuare il trasporto.

È possibile trarre una ulteriore conclusione analizzando l’output degli esperimenti 1 e 2. Il numero di answer set restituiti, infatti, così come il tempo di calcolo, sono fortemente dipendenti dal numero di stati che definiscono il problema. Ciò significa che CLINGO restituirà tutti i goal associati a tutti gli stati in cui questi si presentano. Per cui, a prescindere da quanto facile possa essere il goal o da quanto rapidamente questo venga raggiunto, se il numero di stati è ampio, la computazione impiegherà comunque tanto tempo, poiché CLINGO cercherà tutte le soluzioni in tutti gli stati in cui il goal è soddisfatto.