ESERCIZI SU PIANIFICAZIONE AUTOMATICA

13 Settembre 2004

Si consideri il problema di trasportare un carico tramite un carrello ferroviario che puo' percorrere le tratte tra città connesse. Inizialmente il carico è a Milano, già caricato sul carrello ferroviario. Come goal il carico deve arrivare a Roma.

Caricamento di un oggetto

```
load(Oggetto,Carrello,Location)
PREC:at(Oggetto,Location),
   at(Carrello,Location)
ADD LIST: in(Oggetto,Carrello)
DELETE LIST: at(Oggetto,Location)
```

Trasporto

```
drive(Carrello,Location1,Location2)
PREC:at(Carrello,Location1),
   connected(Location1,Location2)
ADD LIST: at(Carrello,Location2)
DELETE LIST: at(Carrello,Location1)
```

Scaricamento di un oggetto

```
unload(Oggetto,Carrello,Location)
PREC:at(Carrello,Location),
   in(Oggetto,Carrello)
ADD LIST: at(Oggetto,Location)
DELETE LIST: in(Oggetto,Carrello)
```

Stato iniziale:
```
in(carico1,carrello1),
at(carrello1,milano),
connected(milano,bologna),
connected(bologna,roma)
```
Stato goal: `at(carico1,roma)`

Si modellino il problema tramite la formulazione di Kowalski.

SOLUZIONE

Stato iniziale

```
holds(in(carico1,carrello1),s0)
holds(at(carrello1,milano), s0)
holds(connected(milano,bologna),s0)
holds(connected(bologna,roma),s0)
```

Caricamento di un oggetto:

effetti azione

```
holds(in(Oggetto,Carrello),
        do(load(Oggetto,Carrello,Z),S)).
```

Frame axiom

```
holds(V,
do(load(Oggetto,Carrello,Location),S)):-
     holds(V,S),
     V\=at(Oggetto,Location).
```

Precondizioni

```
pact(load(Oggetto,Carrello,Location),S):-
   holds(at(Oggetto,Location),S),
   holds(at(Carrello,Location),S).
```

Scaricamento di un oggetto:

effetti azione

```
holds(at(Ogg,Location),
        do(unload(Ogg,Carrello,Location),S)).
```

Frame axiom

```
holds(V,
do(unload(Oggetto,Carrello,Location),S)):-
     holds(V,S),
     V\= in(Oggetto,Carrello).
```

Precondizioni

```
pact(unload(Oggetto,Carrello,Location),S):-
   holds(in(Oggetto,Carrello),S),
   holds(at(Carrello,Location),S).
```

Trasporto:

effetti azione

```
holds(at(Carrello,Location2),
 do(drive(Carrello,Location1,Location2),S)).
```

Frame axiom

```
holds(V,
do(drive(Carrello,Location1,Location2),S):-
     holds(V,S),
     V\= at(Carrello, Location1).
```

Precondizioni

```
pact(drive(Carrello,Location1,Location2),S)
   :-
   holds(connected(Location1,Location2),S),
   holds(at(Carrello,Location1),S).
```

%Clausola per esprimere la raggiungibilità di uno stato:

```
poss(s0).
poss(do(U,S)):-
     poss(S),
     pact(U,S).
```

%Goal:

```
?-poss(S),holds(at(carico1,roma),S).
```

6 Dicembre 2004
È dato lo stato iniziale descitto dalle seguenti formule atomiche:
[ontable(b), on(a,b), ontable(c), clear(a), clear(c), handempty]
(a,b, c rappresentano dei blocchi e si suppone ci siano infinite posizioni occupabili del tavolo
- le azioni sono modellate opportunamente come segue:

pickup(X)
PRECOND: ontable(X), clear(X), handempty
DELETE: ontable(X), clear(X), handempty
ADD: holding(X)

putdown(X)
PRECOND: holding(X)
DELETE: holding(X)
ADD: ontable(X), clear(X), handempty

stack(X,Y)
PRECOND: holding(X), clear(Y)
DELETE: holding(X), clear(Y)
ADD: handempty, on(X,Y), clear(X)

unstack(X,Y)
PRECOND: handempty, on(X,Y), clear(X)
DELETE: handempty, on(X,Y), clear(X)
ADD: holding(X), clear(Y)

e il goal **ontable(a) and on(b,c)**

si descriva come l'algoritmo lineare backward STRIPS trova un piano per questo goal. Si descriva lo stato e lo stack dei goal passo passo.

SOLUZIONE

Stato:
ontable(b)
ontable(c)
on(a,b)
clear(a)
clear(c)
handempty

Stack:
ontable(a) and on(b,c)

Split del goal:

Stato:
ontable(b)
ontable(c)
on(a,b)
clear(a)
clear(c)
handempty

Stack:
ontable(a)
on(b,c)
ontable(a) and on(b,c)

Goal regression con l'azione **putdown(X):**

Stato:
ontable(b)
on(a,b)
on(c,d)
clear(a)
clear(c)
handempty

Stack:
holding(a)
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Goal regression con l'azione **unstack(a,X):**

Stato:
ontable(b)
ontable(c)
on(a,b)
clear(a)
clear(c)
handempty

Stack:
handempty and on(a,X) and clear(a)
unstack(a,X)
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Split del goal:

Stato:
ontable(b)
ontable(c)
on(a,b)
clear(a)
clear(c)
handempty

Stack:
handempty
on(a,X)
clear(a)
handempty and on(a,X) and clear(a)
unstack(a,X)
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Tutti i primi 4 goal sono verificati nello stato corrente per X=b:

Stato:
ontable(b)
ontable(c)
on(a,b)
clear(a)
clear(c)
handempty

Stack:
unstack(a,b)
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Esecuzione dell'azione unstack(a,b):

Stato:
ontable(b)
ontable(c)
clear(c)
holding(a)
clear(b)

Stack:
putdown(a)

on(b,c)
ontable(a) and on(b,c)

Esecuzione dell'azione putdown(a):

Stato:
ontable(b)
ontable(c)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
on(b,c)
ontable(a) and on(b,c)

Goal regression con l'azione **stack(X,Y):**

Stato:
ontable(b)
ontable(c)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Split del goal:

Stato:
ontable(b)
ontable(c)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
holding(b)
clear(c)
holding(b) and clear(c)
stack(b,c)
on(b,c)

ontable(a) and on(b,c)

Goal regression con l'azione **pickup(X):**

Stato:
ontable(b)
ontable(c)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
ontable(b) and clear(b) and handempty
pickup(b)
clear(c)
holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Split del goal, poi i primi 4 goal sono verificati nello stato corrente.

Stato:
ontable(b)
ontable(c)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
pickup(b)
clear(c)
holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Esecuzione di pickup(b):

Stato:
ontable(c)
clear(c)
ontable(a)
clear(a)
holding(b)

Stack:
clear(c)

holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Primi due goal soddisfatti nello stato corrente:

Stato:
ontable(c)
clear(c)
ontable(a)
clear(a)
holding(b)

Stack:
stack(b,c)
on(b,c)
ontable(a) and on(b,c) and on(c,d)

Esecuzione di stack(b,c)

Stato:
ontable(c)
on(b,c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
on(b,c)
ontable(a) and on(b,c)

Tutti i goal nello stack sono soddisfatti nello stato corrente.

23 Giugno 2004

È dato lo stato iniziale descitto dalle seguenti formule atomiche:
[ontable(b), ontable(d), on(a,b), on(c,d), clear(a), clear(c), handempty]
(a,b, c,d rappresentano dei blocchi e si suppone ci siano infinite posizioni occupabili del tavolo
- le azioni sono modellate opportunamente come segue:

pickup(X)
PRECOND: ontable(X), clear(X), handempty
DELETE: ontable(X), clear(X), handempty
ADD: holding(X)

putdown(X)
PRECOND: holding(X)
DELETE: holding(X)
ADD: ontable(X), clear(X), handempty

stack(X,Y)
PRECOND: holding(X), clear(Y)
DELETE: holding(X), clear(Y)
ADD: handempty, on(X,Y), clear(X)

unstack(X,Y)
PRECOND: handempty, on(X,Y), clear(X)
DELETE: handempty, on(X,Y), clear(X)
ADD: holding(X), clear(Y)

e il goal **ontable(a) and on(b,c)**

si descriva come l'algoritmo lineare backward STRIPS trova un piano per questo goal. Si descriva lo stato e lo stack dei goal passo passo.

SOLUZIONE

Stato:
ontable(b)
ontable(d)
on(a,b)
on(c,d)
clear(a)
clear(c)
handempty

Stack:
ontable(a) and on(b,c)

Split del goal:

Stato:
ontable(b)
ontable(d)
on(a,b)
on(c,d)
clear(a)
clear(c)
handempty

Stack:
ontable(a)
on(b,c)
ontable(a) and on(b,c)

Goal regression con l'azione **putdown(X):**

Stato:
ontable(b)
ontable(d)
on(a,b)
on(c,d)
clear(a)
clear(c)
handempty

Stack:
holding(a)
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Goal regression con l'azione **unstack(X,Y):**

Stato:
ontable(b)
ontable(d)
on(a,b)

on(c,d)
clear(a)
clear(c)
handempty

Stack:
handempty and on(a,b) and clear(a)
unstack(a,b)
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Split del goal:

Stato:
ontable(b)
ontable(d)
on(a,b)
on(c,d)
clear(a)
clear(d)
handempty

Stack:
handempty
on(a,b)
clear(a)
handempty and on(a,b) and clear(a)
unstack(a,b)
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Tutti i primi 4 goal sono verificati nello stato corrente:

Stato:
ontable(b)
ontable(d)
on(a,b)
on(c,d)
clear(a)
clear(c)
handempty

Stack:
unstack(a,b)
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Esecuzione dell'azione unstack(a,b):

Stato:

ontable(b)
ontable(d)
on(c,d)
clear(c)
holding(a)
clear(b)

Stack:
putdown(a)
on(b,c)
ontable(a) and on(b,c)

Esecuzione dell'azione putdown(a):

Stato:
ontable(b)
ontable(d)
on(c,d)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
on(b,c)
ontable(a) and on(b,c)

Goal regression con l'azione **stack(X,Y):**

Stato:
ontable(b)
ontable(d)
on(c,d)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Split del goal:

Stato:
ontable(b)
ontable(d)
on(c,d)
clear(c)

clear(b)
ontable(a)
clear(a)
handempty

Stack:
holding(b)
clear(c)
holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Goal regression con l'azione **pickup(X):**

Stato:
ontable(b)
ontable(d)
on(c,d)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
ontable(b) and clear(b) and handempty
pickup(b)
clear(c)
holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Split del goal, poi i primi 4 goal sono verificati nello stato corrente.

Stato:
ontable(b)
ontable(d)
on(c,d)
clear(c)
clear(b)
ontable(a)
clear(a)
handempty

Stack:
pickup(b)
clear(c)
holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Esecuzione di pickup(b):

Stato:
ontable(d)
on(c,d)
clear(c)
ontable(a)
clear(a)
holding(b)

Stack:
clear(c)
holding(b) and clear(c)
stack(b,c)
on(b,c)
ontable(a) and on(b,c)

Primi due goal soddisfatti nello stato corrente:

Stato:
ontable(d)
on(c,d)
clear(c)
ontable(a)
clear(a)
holding(b)

Stack:
stack(b,c)
on(b,c)
ontable(a) and on(b,c) and on(c,d)

Esecuzione di stack(b,c)

Stato:
ontable(d)
on(c,d)
on(b,c)
ontable(a)
clear(a)
clear(b)
handempty

Stack:
on(b,c)
ontable(a) and on(b,c)

Tutti i goal nello stack sono soddisfatti nello stato corrente.

20 Dicembre 2004

Si consideri il problema di raggiungere New York partendo da Bologna. Sono disponibili alcune tratte aeree descritte nello stato iniziale.

```
go(Partenza,Arrivo)
PREC:at(Partenza), tratta(Partenza,Arrivo)
ADD LIST: at(Arrivo)
DELETE LIST: at(Partenza)

scalo(Partenza,Arrivo,Scalo)
PREC:at(Partenza),tratta(Partenza,Scalo),
     tratta(Scalo,Arrivo)
ADD LIST: at(Arrivo)
DELETE LIST: at(Partenza)
```

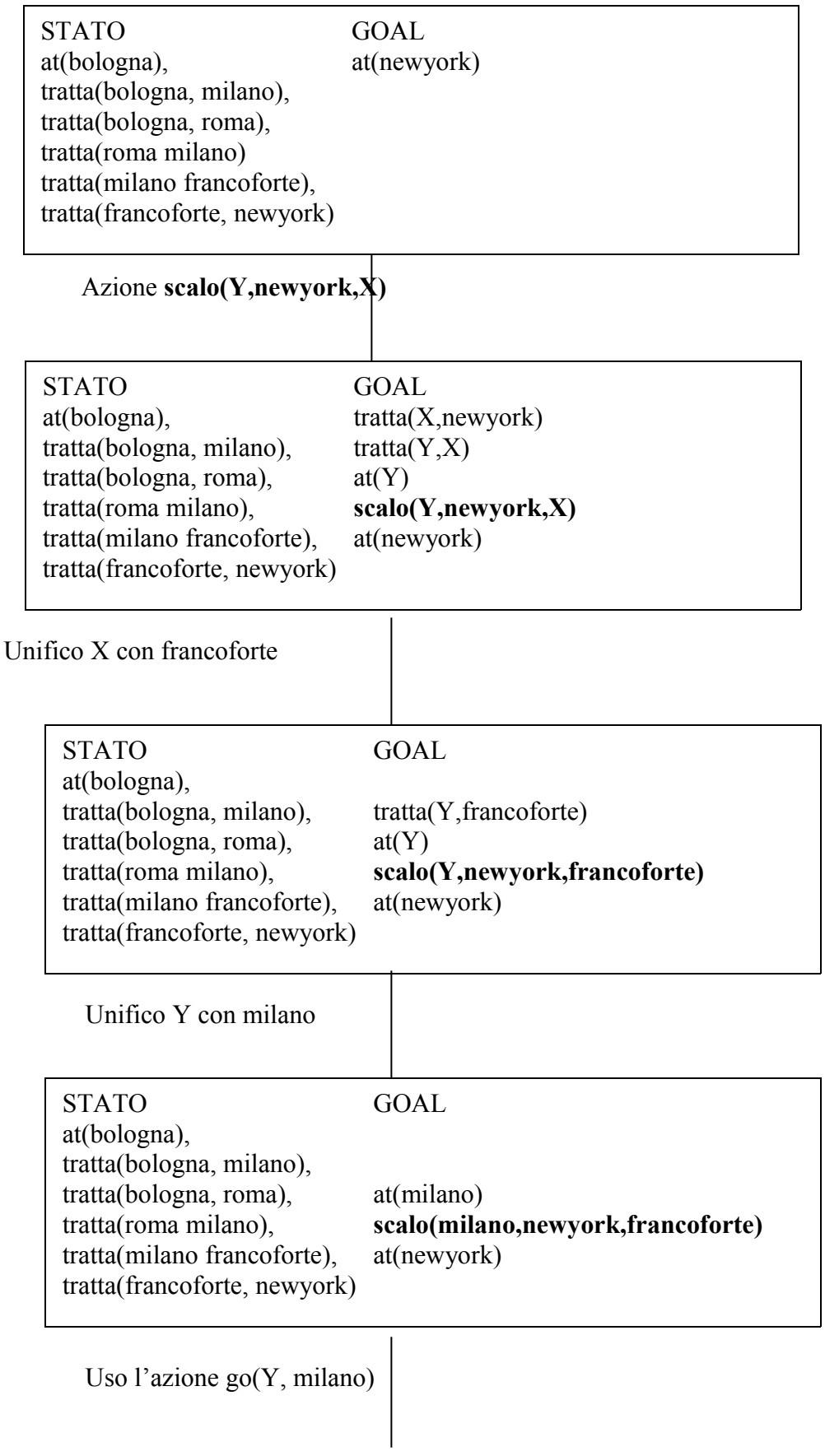
Stato iniziale:

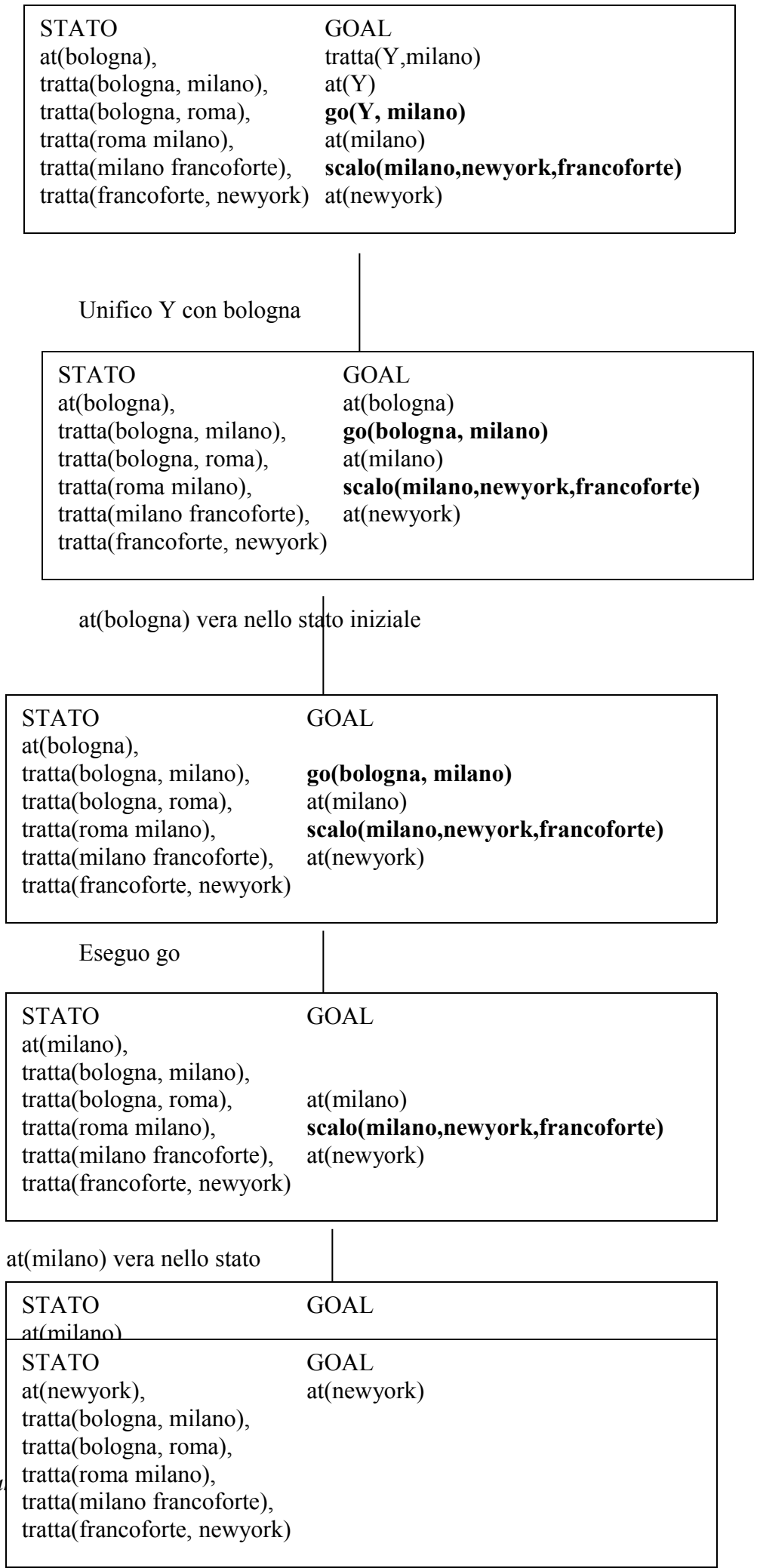
```
at(bologna), tratta(bologna,milano),
tratta(bologna, roma), tratta(roma milano)
tratta(milano francoforte),
tratta(francoforte, newyork)
```

Stato goal: **`at(newyork)`**

Si mostrino i passi compiuti dall'algoritmo STRIPS per risolvere il problema. Si mostri UNA SOLA STRADA nello spazio di ricerca che porti a una soluzione USANDO UNA AZIONE go e UNA scalo.

SOLUZIONE


STATO
GOAL
at(bologna),
at(newyork)
tratta(bologna, milano),
tratta(bologna, roma),
tratta(roma milano)
tratta(milano francoforte),
tratta(francoforte, newyork)
Azione **scalo(Y,newyork,X)**
STATO
GOAL
at(bologna),
tratta(X,newyork)
tratta(bologna, milano),
tratta(Y,X)
tratta(bologna, roma),
at(Y)
tratta(roma milano),
scalo(Y,newyork,X)
tratta(milano francoforte),
at(newyork)
tratta(francoforte, newyork)
Unifico X con francoforte
STATO
GOAL
at(bologna),
tratta(bologna, milano),
tratta(Y,francoforte)
tratta(bologna, roma),
at(Y)
tratta(roma milano),
scalo(Y,newyork,francoforte)
tratta(milano francoforte),
at(newyork)
tratta(francoforte, newyork)
Unifico Y con milano
STATO
GOAL
at(bologna),
tratta(bologna, milano),
tratta(bologna, roma),
at(milano)
tratta(roma milano),
scalo(milano,newyork,francoforte)
tratta(milano francoforte),
at(newyork)
tratta(francoforte, newyork)
Uso l'azione go(Y, milano)


STATO
at(bologna),
tratta(bologna, milano),
tratta(bologna, roma),
tratta(roma milano),
tratta(milano francoforte),
tratta(francoforte, newyork)
GOAL
tratta(Y,milano)
at(Y)
go(Y, milano)
at(milano)
scalo(milano,newyork,francoforte)
at(newyork)
Unifico Y con bologna
STATO
at(bologna),
tratta(bologna, milano),
tratta(bologna, roma),
tratta(roma milano),
tratta(milano francoforte),
tratta(francoforte, newyork)
GOAL
at(bologna)
go(bologna, milano)
at(milano)
scalo(milano,newyork,francoforte)
at(newyork)
at(bologna) vera nello stato iniziale
STATO
at(bologna),
tratta(bologna, milano),
tratta(bologna, roma),
tratta(roma milano),
tratta(milano francoforte),
tratta(francoforte, newyork)
GOAL
go(bologna, milano)
at(milano)
scalo(milano,newyork,francoforte)
at(newyork)
Eseguo go
STATO
at(milano),
tratta(bologna, milano),
tratta(bologna, roma),
tratta(roma milano),
tratta(milano francoforte),
tratta(francoforte, newyork)
GOAL
at(milano)
scalo(milano,newyork,francoforte)
at(newyork)
at(milano) vera nello stato
STATO
at(milano)
GOAL
STATO
at(newyork),
tratta(bologna, milano),
tratta(bologna, roma),
tratta(roma milano),
tratta(milano francoforte),
tratta(francoforte, newyork)
GOAL
at(newyork)

Eseguo scalo e ottengo il goal soddisfatto

23 Marzo 2006

Si consideri il seguente problema: un carrello è nella posizione ‘a’ ed è pieno di pietre. Inoltre, vi è della ghiaia nella posizione 'b' connessa ad 'a'.
Il goal che si vuole raggiungere è che le pietre siano tutte in 'a' e che la ghiaia sia caricata nel carrello nella posizione 'b', avendo a disposizione le seguenti azioni

Scaricamento carrello
unload(Wagon,Item)
PREC: full(Wagon,Item), in(Wagon,X)
EFFECT: empty(Wagon), in(Item,X)

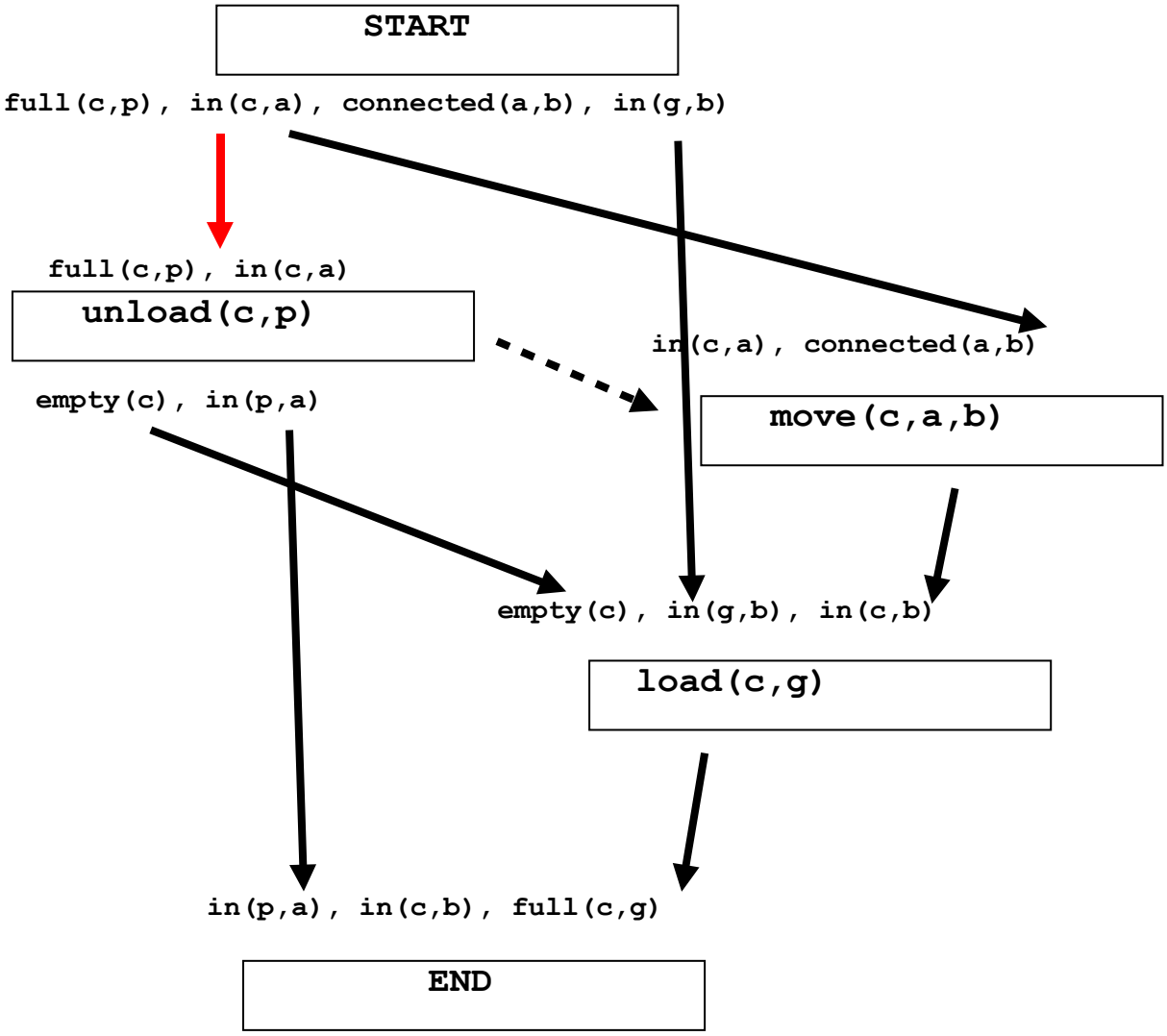
Caricamento carrello
load(Wagon,Item)
PREC:empty(Wagon), in(Wagon,X), in(Item,X)
EFFECT: full(Wagon,Item), ¬in(Item,X)

Spostamento del carrello tra due posizioni connesse
move(Wagon,Loc1,Loc2)
PREC: in(Wagon,Loc1), connected(Loc1,Loc2)
EFFECT: in(Wagon,Loc2), ¬in(Wagon,Loc1),

Stato iniziale:
in(c,a), full(c,p), connected(a,b), in(g,b)

Stato goal: **in(p,a), in(c,b), full(c,g)**

Si mostrino i passi compiuti dall’algoritmo POP per risolvere il problema. Si mostrino i causal link e le eventuali minacce.



Questo piano contiene un threat: infatti il *causal link* in rosso (in(c,a)) è minacciato dagli effetti dell'azione move(c,b,a) che come effetto contiene not in(c,a).
In questo caso si può applicare la Demotion e far sì che move(c,b,a) segua unload(c,p) (vincolo di successione tratteggiato in figura).

13 Dicembre 2006

Si consideri il seguente problema: un'auto è nella posizione '1' ed ha a bordo il solo autista (a). Inoltre, vi è una persona (b) nella posizione '2' connessa ad '1'.
Il goal che si vuole raggiungere è che le due persone siano tutte in '1', fuori dell'auto, avendo a disposizione le seguenti azioni

Scaricamento persona
unload(Car,Person)
PREC: on(Car,Person), in(Car,X)
EFFECT: ¬on(Car,Person), in(Person,X)

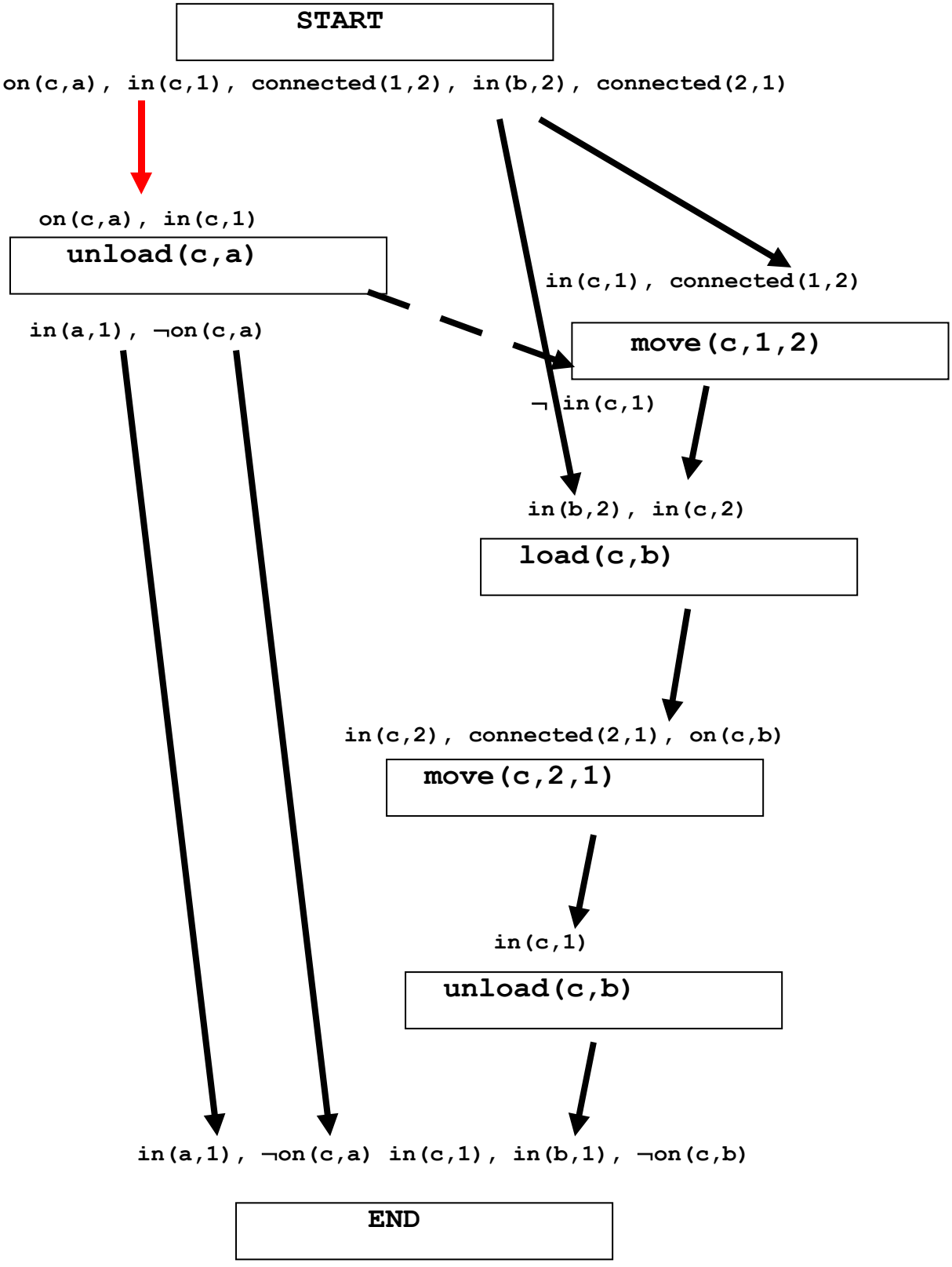
Caricamento persona
load(Car,Person)
PREC: in(Car,X), in(Person,X)
EFFECT: on(Car,Person), ¬in(Person,X)

Spostamento dell'auto tra due posizioni connesse
move(Car,Loc1,Loc2)
PREC: in(Car,Loc1), connected(Loc1,Loc2)
EFFECT: in(Car,Loc2), ¬in(Car,Loc1),

Stato iniziale:
in(c,1), on(c,a), connected(1,2), connected(2,1), in(b,2)

Stato goal: **in(a,1), in(b,1), in(c,1), ¬on(c,a), ¬on(c,b)**

Si mostrino i passi compiuti dall'algoritmo POP per risolvere il problema. Si mostrino i causal link e le eventuali minacce.



Questo piano contiene un threat: infatti il *causal link* in rosso (in(c,1)) è minacciato dagli effetti dell'azione move(c,1,2) che come effetto contiene not in(c,1).

In questo caso si può applicare la Demotion e far sì che unload(c,a) preceda move(c,1,2) (vincolo di precedenza tratteggiato in figura) e di fatto tutte le azioni successive fino all'azione END.

13 Dicembre 2007

Si consideri il seguente problema: un carrello (‘c’) è nella posizione ‘2’ con dei sacchi di cemento (‘s’) caricati. In posizione ‘3’ si trovano dei mattoni (‘m’). La posizione '2' e la posizione ‘3’ sono connesse ad '1'.
Il goal che si vuole raggiungere è che il carrello sia in ‘1’ e mattoni e cemento scaricati in posizione ‘1’, avendo a disposizione le seguenti azioni

Scaricamento materiale
unload(C,M)
PREC: on(C,M), in(C,X)
EFFECT: ¬on(C,M), in(M,X)

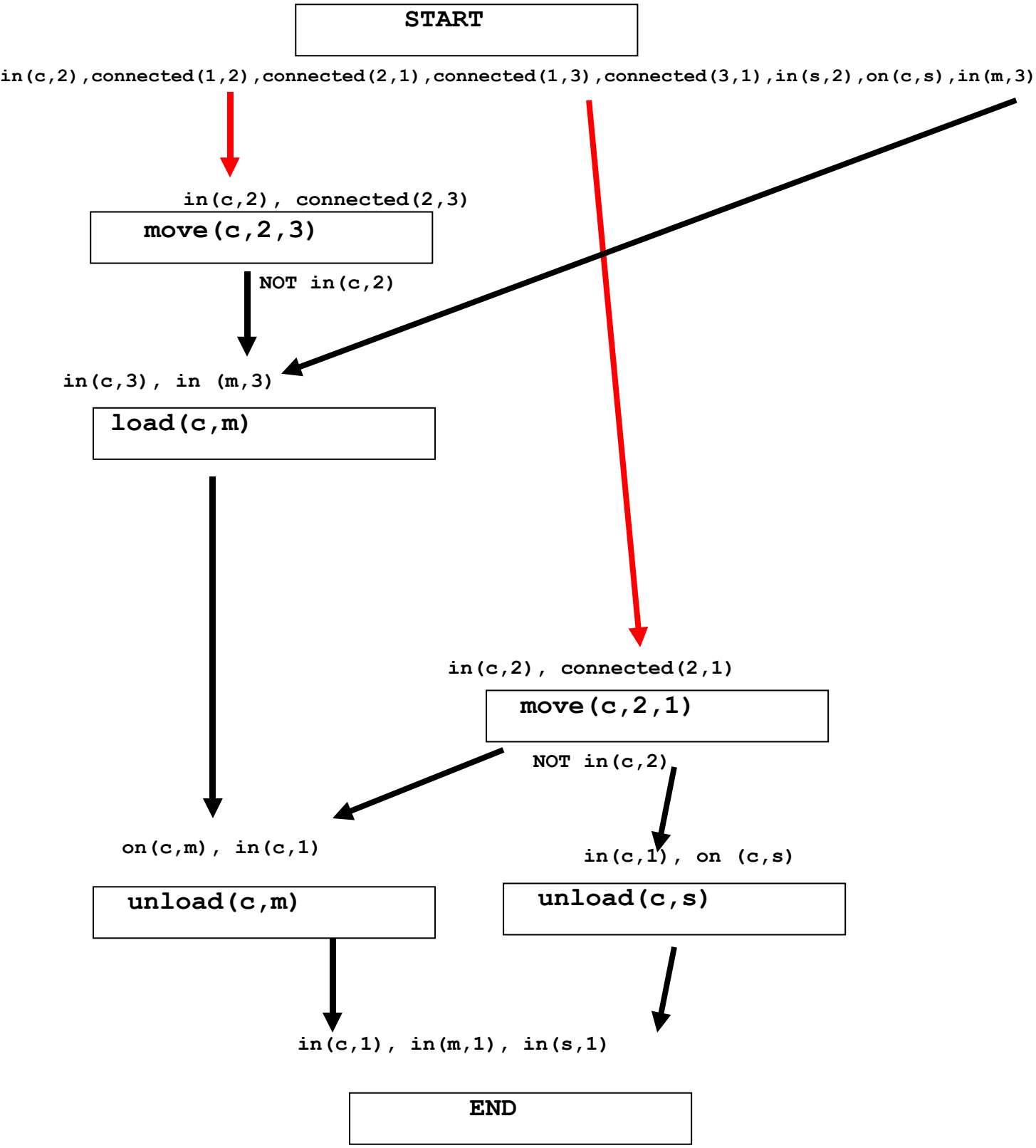
Caricamento materiale
load(C,M)
PREC: in(C,X), in(M,X)
EFFECT: on(C,M), ¬in(M,X)

Spostamento del carrello tra due posizioni connesse
move(C,Loc1,Loc2)
PREC: in(C,Loc1), connected(Loc1,Loc2)
EFFECT: in(C,Loc2), ¬in(C,Loc1),

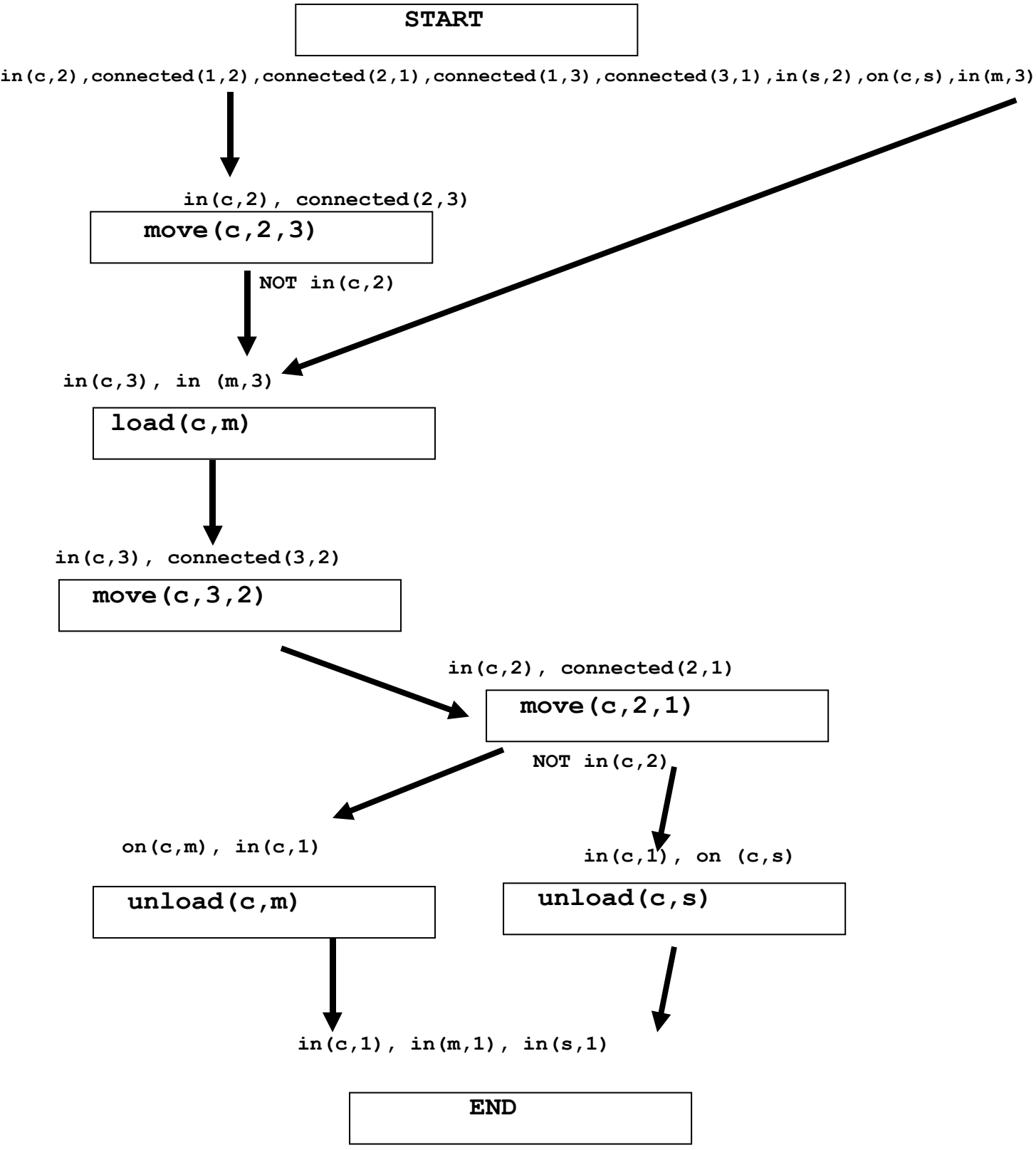
Stato iniziale:
in(c,2), connected(1,2), connected(2,1), connected(1,3), connected(3,1), in(s,2), on(c,s), in(m,3)

Stato goal: **in(c,1), in(s,1), in(m,1)**

Si mostrino i passi compiuti dall’algoritmo POP per risolvere il problema.



Questo piano contiene un threat: infatti i due *causal link* in rosso sono minacciato dagli effetti delle due azioni move che come effetto contengono not in(c,2).
In questo caso non si possono applicare la demotion e la promotion. Serve una azione che ristabilisca in (c,2)


START
in(c,2),connected(1,2),connected(2,1),connected(1,3),connected(3,1),in(s,2),on(c,s),in(m,3)
in(c,2), connected(2,3)
move(c,2,3)
NOT in(c,2)
in(c,3), in (m,3)
load(c,m)
in(c,3), connected(3,2)
move(c,3,2)
in(c,2), connected(2,1)
move(c,2,1)
NOT in(c,2)
on(c,m), in(c,1)
in(c,1), on (c,s)
unload(c,m)
unload(c,s)
in(c,1), in(m,1), in(s,1)
END