Fondamenti di Intelligenza Artificiale M
a.a. 2010/2011

Note sul compito d'esame

Composizione di un possibile testo d'esame

Il testo d'esame di "*Fondamenti di Intelligenza Artificiale M*", dal 2009/2010 riprende in parte la struttura del testo d'esame del precedente corso "*Fondamenti di Intelligenza Artificiale L-S*", ed in parte e' composto da esercizi riguardanti la parte di programma "nuova" (cioè non presente nel corso precedente).

Esempi con soluzione si possono trovare nel sito del Corso AA 2009/2010

http://www.lia.deis.unibo.it/Courses/AI/fundamentalsAI2009-10/

Argomenti d'esame

Al fine di prepararsi all'esame, è consigliato svolgere a titolo di esercizio tutti i testi d'esami relativi ai corsi passati. Inoltre, è opportuno prepararsi anche sui nuovi argomenti trattati. In particolare, sarà ragionevole attendersi esercizi e/o domande di teoria rispetto al vecchio corso di Fondamenti di Intelligenza Artificiale L-S su:

- Semantic Web e Description Logic
- Meta-interpreti in Prolog
- Planning e Sistemi a regole
- Domande aperte su tutti gli argomenti trattati nel corso.

Esempi di possibili esercizi – Description Logic e SW

- Introdurre brevemente la differenza tra Open World Assumption e Close World Assumption, illustrando tale differenza con una piccola knowledge base di esempio.
- Illustrare brevemente i possibili modelli semantici visti nel corso per rappresentare la conoscenza.
- Spiegare brevemente, nell'ambito dell'iniziativa Semantic Web, il ruolo degli standard RDF e OWL.
- Nell'ambito di Semantic Web, introdurre brevissimamente la funzione dello standard SPARQL, eventualmente con qualche esempio di query.
- Considerando gli operatori (EXISTS, FORALL, AND, FILLS) e i simboli logici di Description Logic visti a lezione, rappresentare la seguente conoscenza: "Le conferenze importanti sono tutte quelle che hanno almeno 3 iscritti. Le conferenze italiane sono quelle dove tutti gli iscritti sono italiani. Alla conferenza "xyz" si sono iscritti Federico, Marco e Paolo. Tutti e tre sono italiani." Cosa è possibile inferire in Description Logic da questa base di conoscenza?

Esempi di possibili esercizi – Meta-interpreti in Prolog

Si scriva un meta-interprete depth-first per Prolog puro che quando cerca di selezionare una clausola la cui testa unifica con un dato goal G per fare il passo di risoluzione, presenti all'utente la lista di tutte le clausole ancora non tentate che unificano con G e lasci a lui la scelta. In questo modo le clausole non sono più tentate nell'ordine testuale di Prolog.

Esempi di possibili esercizi – Meta-interpreti in Prolog

```
metadf(true):-!.
metadf((A,B)):-!,metadf(A),metadf(B).
metadf(X):-
    findall(G,unificano(X,G),L),!,
    choice(X,L).

unificano(X,G) :- clause(X,Body), G=[X,Body].
```

% Ora viene costruita una lista L che contiene liste del ti-% po [Head,Body] di clausole la cui testa unifica con X.

Esempi di possibili esercizi – Meta-interpreti in Prolog

```
% Nel caso di lista vuota l'alternativa fallisce.
choice( _, [ ]):- !,
    write('Questa alternativa fallisce'),
    fail.

% Se esiste un'unica clausola è inutile chiedere
% all'utente di effettuare una scelta.
choice(X, [  [H, Body]  ]) :-  !,
    choice1( X, H, [  [H,Body]  ],  Body).

% Nel caso di più clausole l'utente effettua una scelta.
choice( X, L) :-
    scelta(  L, [ H, Body]  ),
    choice1( X, H, L, Body).
```

Esempi di possibili esercizi – Meta-interpreti in Prolog

```
% choice1 la prima volta che viene chiamato provvede all'unificazione della testa della clausola
selezionata con il sottogoal da dimostrare poi cerca di dimostrare il body.
choice1( X, H, _, Body):-
        H=X,
        metadf(Body).

% In caso di fallimento è necessario tentare le strade ancora aperte permettendo all'utente di scegliere
nuovamente ma eliminando dalle possibili scelte quelle già effettuate.
choice1(X,H,L,Body):-
        delete([H,Body],L,L1),
        choice(X,L1).

scelta(L,[H,Body]):-
        write('scegli una clausola dalla lista'),
        nl,
        write(L),
        read([H,Body]).
NB: La lista che viene presentata all'utente contiene le clausole che unificano con la clausola da
        dimostrare, già unificate.
```

Esempi di possibili esercizi – Meta-interpreti in Prolog

Si scriva un meta-interprete PROLOG che assegna un limite massimo alla profondità dell'albero AND-OR che esplora. Tale profondità **D** può essere specificata dall'utente all'atto dell'invocazione del meta-interprete. Nel caso in cui la profondità dell'albero superi **D**, il meta-interprete non fallisce, ma produce come uscita il ramo corrente dell'albero che stava esplorando.

Più in dettaglio, il meta-interprete è invocato con goal della forma:

:- solve(Goal, D,Overflow).

Dove Goal è il goal iniziale e D una profondità massima.

Il predicato solve ha successo:

1. Nel caso in cui il Goal ha successo con una profondità massima che non supera D .

   In questo caso la variabile Overflow verrà istanziata a **no_overflow**;
2. Nel caso in cui si sia superata la profondità massima D. In questo caso la variabile Overflow verrà istanziata ad una struttura del tipo **overflow([A|B])**, dove la lista [A|B] contiene i sottogoal sul corrente ramo di ricerca.

Esempi di possibili esercizi – Meta-interpreti in Prolog

Esempio: dato il programma:

```
p(X) :- q(X), r(X).
q(1).
r(X) :- p(X).
```

otterremo per :- solve(p(X), 5, Overflow),
il risultato:
X=1,
Overflow = overflow([p(1), r(1), p(1), r(1), p(1)])

Esempi di possibili esercizi – Meta-interpreti in Prolog

```
% Goal ha successo a una profondità minore di D
solve(true, D1, no_overflow)  :-  D1>0.

% Raggiunta profondità massima
solve( A, 0, overflow( [ ] )    ).

% Caso dell'and di due sottogoal
solve( (A,B),  D,  Overflow)  :-
      D>0,
      solve( A,  D,  OverflowA),
      solve_conj( OverflowA,  B,  D,  Overflow).

solve( Goal,  D,  Overflow)  :-
      D>0,
      clause( Goal, B),
      D1 is D - 1,
      solve( B,  D1,  OverflowB),
      return_overflow(  OverflowB,  Goal,  Overflow).
```

Esempi di possibili esercizi – Meta-interpreti in Prolog

```
solve_conj( overflow(S),  B, D, overflow(S)).
solve_conj( no_overflow, B, D, Overflow):-
    solve(  B,  D,  Overflow).


% return_overflow torna no_overflow nel caso in cui il Goal ha successo
    con una profondità che non supera D. In caso contrario provvede a
    costruire la lista di goal e sottogoal richiesta
return_overflow( no_overflow,  A,  no_overflow).
return_overflow( overflow(S), A, overflow([ A | S])).
```

Esempi di possibili esercizi – Planning

È dato lo stato iniziale descritto dalle seguenti formule atomiche:
[ontable(a,p1), ontable(d,p3), on(c,d), clear(a), clear(c), empty(p2), handempty]
(a,c,d rappresentano dei blocchi e p1,p2,p3 sono le uniche 3 posizioni occupabili del tavolo

Le azioni sono modellate opportunamente come segue:

- **pickup(X,Pos)**
  PRECOND: ontable(X,Pos), clear(X), handempty
  DELETE: ontable(X,Pos), clear(X), handempty
  ADD: holding(X), empty(Pos)
- **putdown(X,Pos)**
  PRECOND: holding(X), empty(Pos)
  DELETE: holding(X), empty(Pos)
  ADD: ontable(X,Pos), clear(X), handempty

(Pos è una variabile che rappresenta la posizione del tavolo, X,Y rappresentano blocchi)

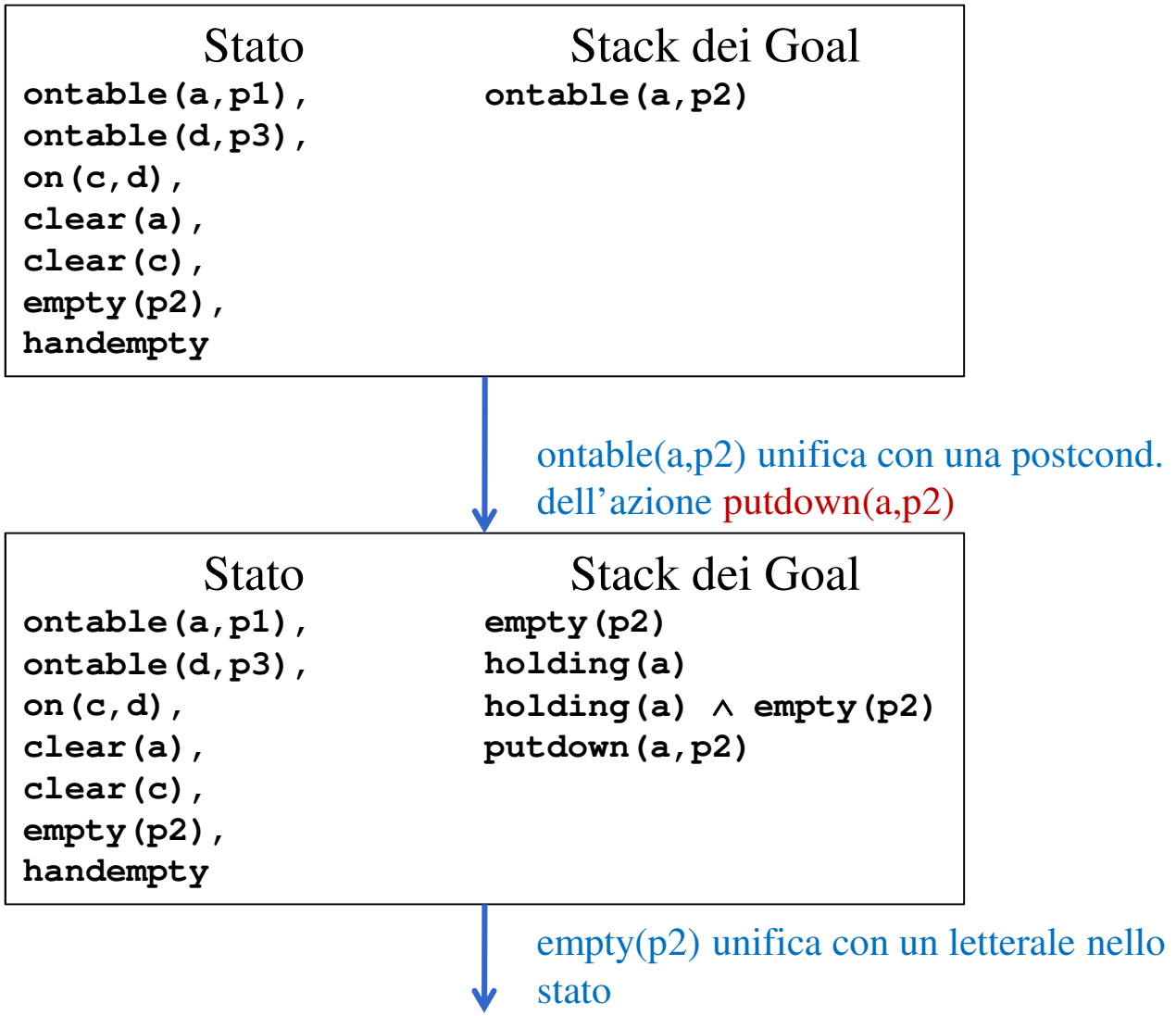
Esempi di possibili esercizi – Planning

- **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)

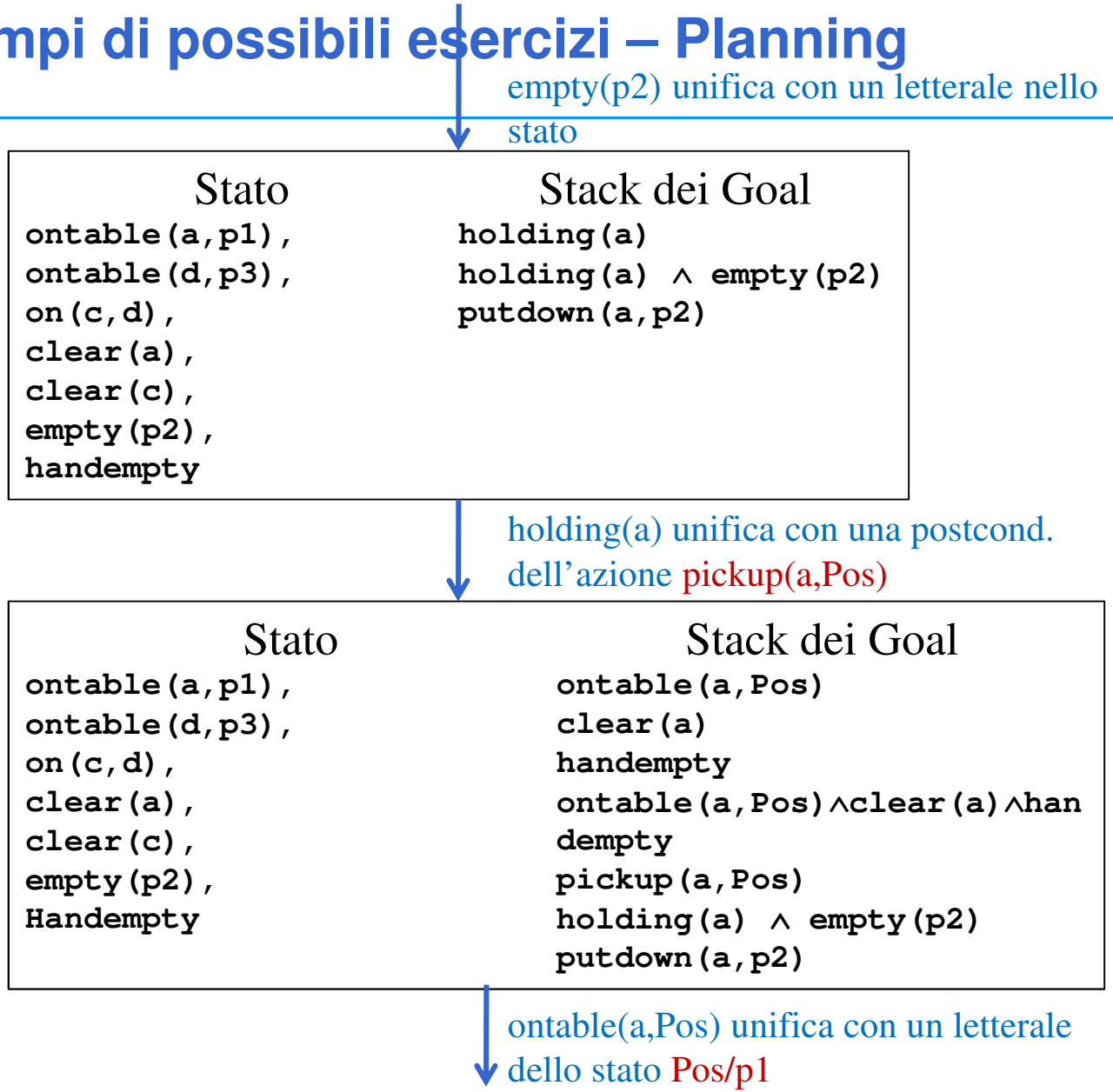
e il goal **ontable(a,p2)**

Si descriva come l'algoritmo lineare backward STRIPS trova un piano per questo goal (quindi si mostri SOLO una strada nell'albero di ricerca considerando come ordinamento tra precondizioni in and nello stack quello che considera prima le precondizioni che unificano direttamente con un letterale dello stato e poi quelle che per essere soddisfatte hanno bisogno di una azione). Si descriva lo stato e lo stack dei goal passo passo.

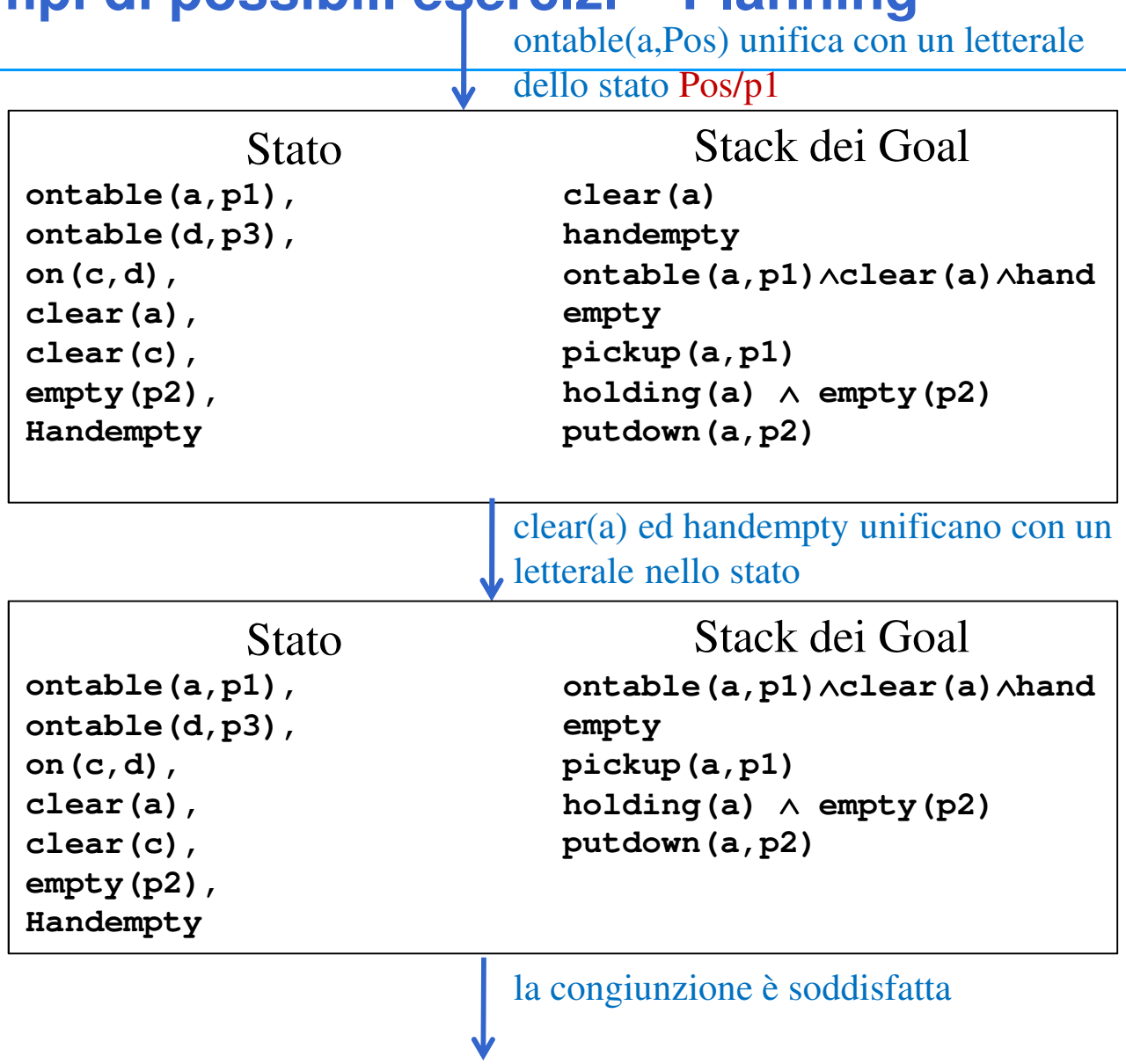
Esempi di possibili esercizi – Planning



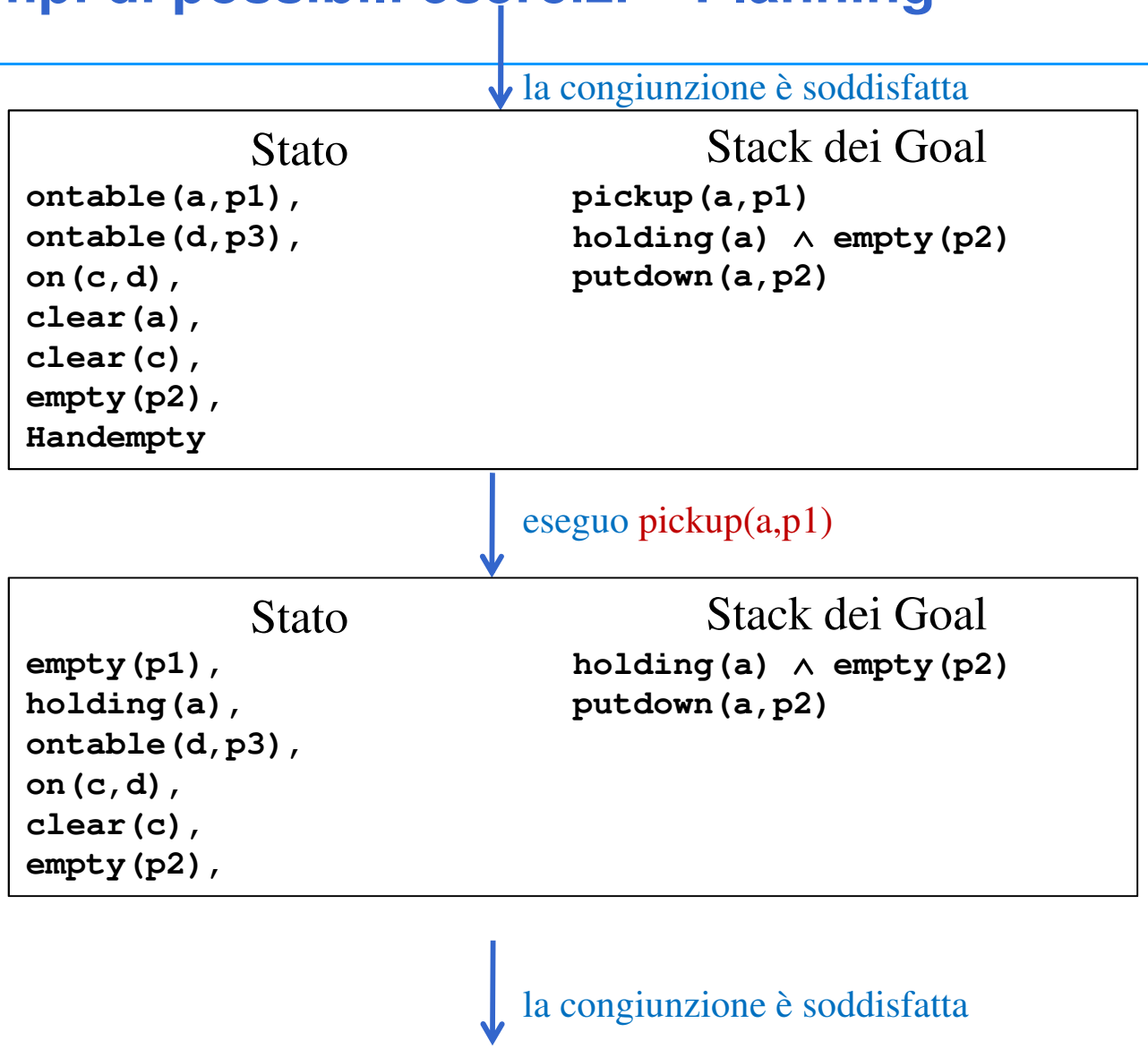
Esempi di possibili esercizi – Planning



Esempi di possibili esercizi – Planning



Esempi di possibili esercizi – Planning



Esempi di possibili esercizi – Planning

la congiunzione è soddisfatta

Stato	Stack dei Goal

`empty(p1),`
`holding(a),`
`ontable(d,p3),`
`on(c,d),`
`clear(c),`
`empty(p2),`
	`putdown(a,p2)`

Eseguo putdown(a,p2)

Stato	Stack dei Goal

`empty(p1),`
`handempty(a),`
`ontable(d,p3),`
`on(c,d),`
`clear(c),`
`ontable(a,p2)`	