PLANNING DEDUTTIVO

Esercizi in Prolog

PLANNING DEDUTTIVO – Esercizio 1

- Prendiamo l'esempio visto a lezione e usiamo la formulazione di Kowalski:

Stato iniziale

holds(on(a,d),s0).

holds(on(b,e),s0).

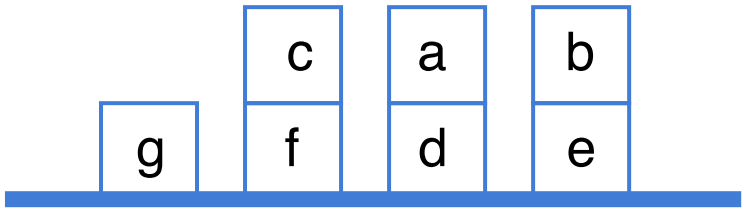
holds(on(c,f),s0).

holds(clear(a),s0).

holds(clear(b),s0).

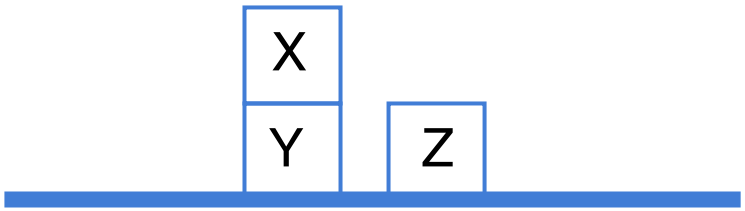
holds(clear(c),s0).

holds(clear(g),s0).

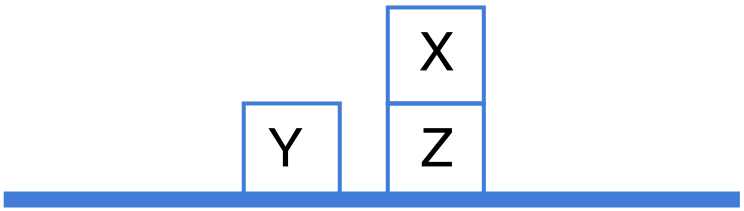


PLANNING DEDUTTIVO

Consideriamo l'unica azione move(X,Y,Z)



on(X,Y)
clear(X)
clear(Z)



clear(Y)
on(X,Z)

PLANNING DEDUTTIVO

%Effetti dell'azione move(X,Y,Z): *(Post)*

holds(clear(Y),do(move(X,Y,Z),S)).

holds(on(X,Z),do(move(X,Y,Z),S)).

% Clausola che esprime le precondizioni dell'azione move(X,Y,Z): *(Pre)*

```
pact(move(X,Y,Z),S):-
    holds(clear(X),S), holds(clear(Z),S),
    holds(on(X,Y),S), X\==Z.
```

PLANNING DEDUTTIVO

%Clausola per esprimere le condizioni di frame: *(FA)*

```
holds(V,do(move(X,Y,Z),S)):-
    holds(V,S),
    V\=clear(Z),
    V\=on(X,Y).
```

%Clausola per esprimere la raggiungibilità di uno stato:

```
poss(s0).
```

(Meta-assioma)

```
poss(do(A,S)):-
    poss(S),
    pact(A,S).
```

Disuguaglianza (SICStus Prolog)

- **T1 \= T2,** verifica se T1 e T2 non possono unificre;
- **T1 \== T2,** verifica se T1 e T2 non sono uguali (identici); ha successo se i due termini (non valutati) non sono identici

- In SICStus Prolog per le espressioni, l'operatore diverso è indicato come: **=\=**

- **E1 =\= E2** ha successo se le due espressioni (che vengono valutate) non hanno lo stesso valore

- Quindi
    - 2*2 \== 4 è vero
    - 2*2 =\= 4 è falso

PLANNING DEDUTTIVO

- NOTA: abbiamo una clausola che esprime condizioni di frame per ogni AZIONE

- Goal:

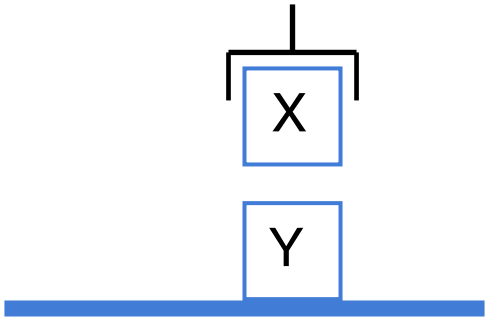
:- poss(S),holds(on(a,b),S),holds(on(b,g),S).

- Attivare il trace per monitorare la risoluzione
- Verificare la costruzione di altri piani con altre query, ad esempio quella proposta come esercizio:

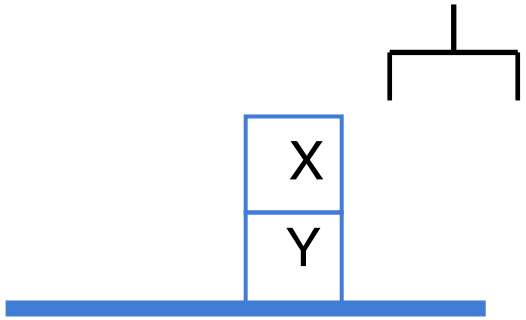
:- poss(S),holds(on(a,b),S),holds(on(c,a),S).

PLANNING DEDUTTIVO – Esercizio 2

- Usare il mondo a blocchi, ma cambiare e modellare le azioni

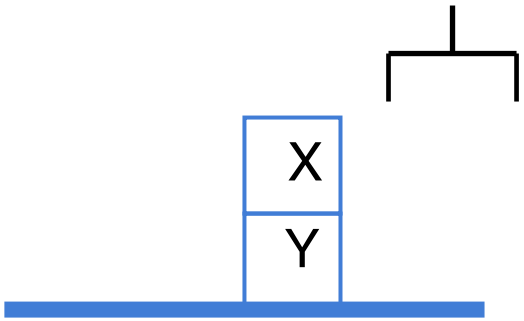


clear(Y), holding(X)

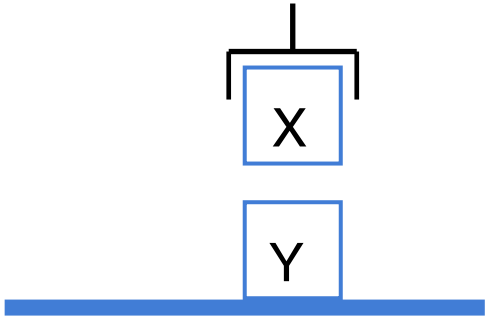


on(X,Y), handempty, clear(X)

STACK(X,Y)



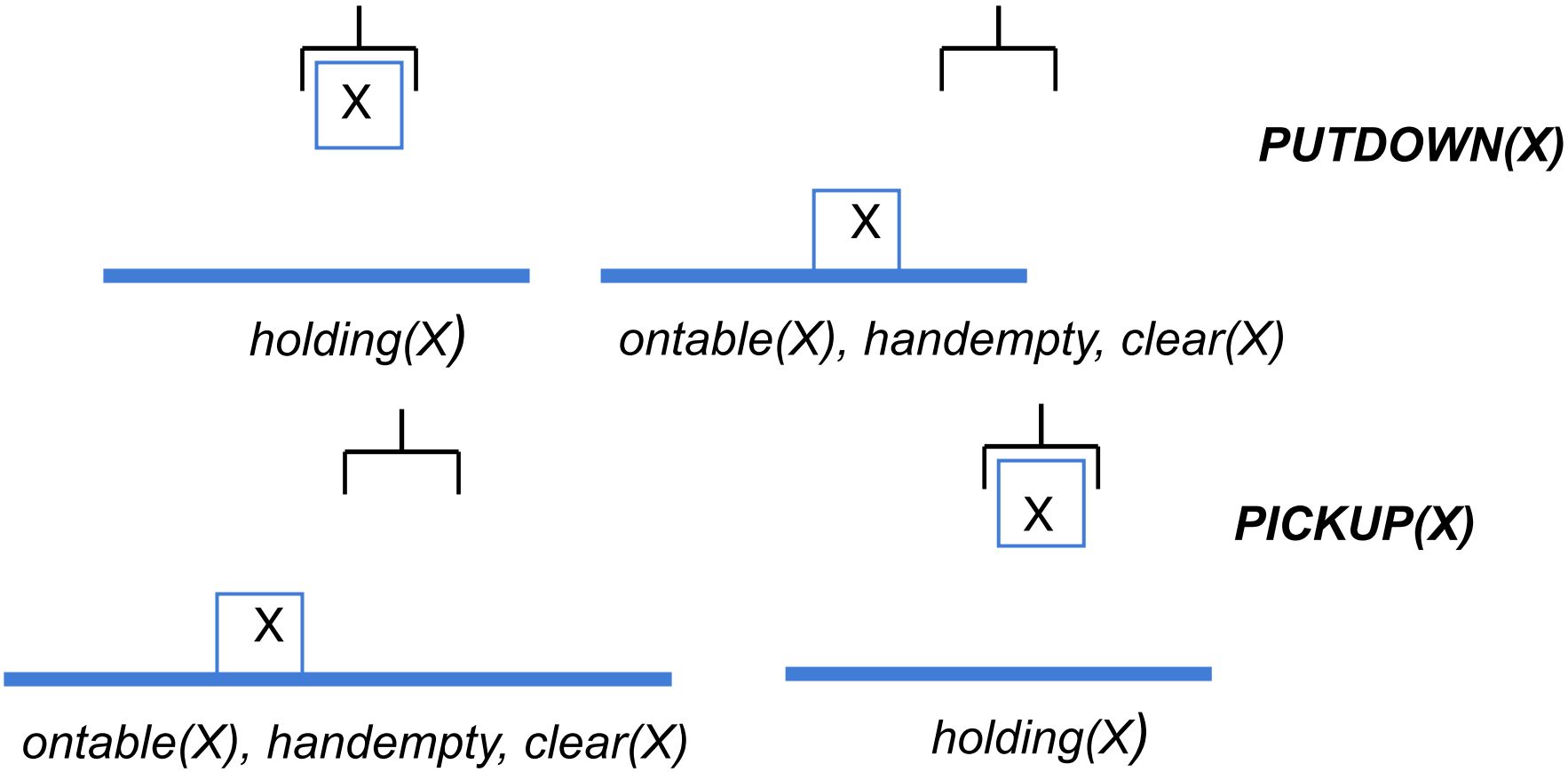
on(X,Y), handempty, clear(X)



clear(Y), holding(X)

UNSTACK(X,Y)

PLANNING DEDUTTIVO – Esercizio 2



PLANNING DEDUTTIVO – Esercizio 2

%Effetti dell'azione stack(X,Y): *(Post)*

holds(clear(X),do(stack(X,Y),S)).

holds(on(X,Y),do(stack(X,Y),S)).

holds(handempty, do(stack(X,Y),S)).

% Clausola che esprime le precondizioni dell'azione stack(X,Y): *(Pre)*

pact(stack(X,Y),S):-

holds(clear(Y),S),

holds(holding(X),S).

PLANNING DEDUTTIVO – Esercizio 2

%Clausola per esprimere le condizioni di frame: *(FA)*

```
holds(V,do(stack(X,Y),S)):-
    holds(V,S),
    V\=clear(Y),
    V\=holding(X).
```

%Clausola per esprimere la raggiungibilità di uno stato:

```
poss(s0).
```

(Meta-assioma)

```
poss(do(A,S)):-
    poss(S),
    pact(A,S).
```

PLANNING DEDUTTIVO – Esercizio 2

- Stesso stato iniziale di prima

- Goal:

:- poss(S),holds(on(a,b),S),holds(on(b,g),S).

- Attivare il trace per monitorare la risoluzione

- Verificare la costruzione di altri piani con altre query, ad esempio quella proposta come esercizio:

:- poss(S),holds(on(a,b),S),holds(on(c,a),S).

PLANNING DEDUTTIVO – Esercizio 3

Si modellino ora le seguenti azioni

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)
```

PLANNING DEDUTTIVO – Esercizio 3

Con il seguente stato iniziale e goal:

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

Stato goal: **`at(carico1,roma)`**

PLANNING DEDUTTIVO – Esercizio 3

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

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

Stato goal: `at(carico1,roma)`

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

Esercizio 3 – load(O,C,L)

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

%Effetti dell'azione load(O,C,L): *(Post)*

holds(in(O,C),do(load(O,C,L),S)).

% Clausola che esprime le precondizioni dell'azione load(O,C,L): *(Pre)*

```
pact(load(O,C,L),S):-
        holds(at(O,L),S),
    holds(at(C,L),S).
```

%Clausola per esprimere le condizioni di frame: *(FA)*

```
holds(V,do(load(O,C,L),S)):-
   holds(V,S), V\==at(O,L).
```