**T-79.154** **Syksy 2003**
**Logiikka tietotekniikassa: erityiskysymyksiä II**
**Laskuharjoitus 9**
**Ratkaisut**

1. 
```
%% vacuum.lp -- a domain description file for planning in the vacuum
%%              domain.

%% predicates:
% at(V, L, I) -- a vacuum cleaner V is at a place L at the time step
%                I.
% clean(L, I) -- a location L is clean at a time step I.
%
%% Actions:
% move(V, F, T, I) -- move the vacuum cleaner V from a place F to a
%                     place T at a time step I.
%
% suction(V, L, I) -- a vacuum cleaner V cleans the location L at time
%                    step I.

%The basic encoding of the actions is such that the preconditions of an
%action imply that the action can be performed.
%
%    { action } :- preconditions.
%
% An action implies its effects.
%
%    effects :- action.
%
% The blocking of conflicting actions is done by adding a constraint
% that says that if an action doesn't change its own precondition,
% then the precondition has to hold also at the next instant:
%
%      precondition(I+1) :- action, precondition(I),
%                           keeps_precondition(action).

%% SUCTION

% preconditions
{ suction(V, L, I) } :-
```

```
    vacuum(V),
    location(L),
    time(I),
    at(V, L, I),
    not clean(L, I).

% effects
clean(L,I+1) :-
    vacuum(V),
    location(L),
    time(I),
    suction(V,L,I).

% because a vacuum vacuum doesn't move while cleaning, the vacuum
% cleaner has to be at the same location in the next time step
at(V, L, I+1) :-
    at(V, L, I),
    vacuum(V),
    location(L),
    time(I),
    suction(V, L, I).

%% MOVE

%action
{ move(V, F, T, I) } :-
    vacuum(V),
    next_to(F, T),
    time(I),
    at(V, F, I).

% effect
at(V, T, I+1) :-
    vacuum(V),
    next_to(F, T),
    time(I),
    move(V, F, T, I).

moves(V, I) :-
    vacuum(V),
    next_to(F, T),
```

```
    time(I),
    move(V, F, T, I).

%% Frame axioms:

% a vacuum cleaner may not be in two places at the same time
 :- 2 { at(V, L, I) : location(L) },
    vacuum(V),
    time(I).

% a vacuum cleaner stays at the same spot if it doesn't move
at(V, L, I+1) :-
    vacuum(V),
    location(L),
    time(I),
    at(V, L, I),
    not moves(V, I).

% a once cleaned room stays cleaned
clean(L, I+1) :-
    location(L),
    time(I),
    clean(L, I).

%% we want to have n time steps.
time(1..n).

% goal state
compute 1 {  clean(L, n+1) : location(L) } .
```

2. 
```
%% The idea of the grocery world is similar to the vacuum world. That
%% is, preconditions of an action imply that the action may be
%% performed and an action implies its effects:

% { action } :- preconditions.
% effect :- action.

% First define the time and the end moment
time(1..n).
const end_time = n+1.
```

```
%% MOVE
% action
{ move(F, T, I) } :-
    next(F, T),
    time(I),
    at(F, I).

% effect
at(T, I+1) :-
    next(F, T),
    time(I),
    move(F, T, I).

moving(I) :-
    next(F, T),
    time(I),
    move(F, T, I).

%% PICK

% action
{ pick(Item, I) } :-
    in_list(Item),
    time(I),
    not has(Item, I),
    not paid(I),
    at(L, I),
    located(Item, L).

% effect
has(Item, I+1) :-
    in_list(Item),
    time(I),
    pick(Item, I).

% we may not move while picking items
at(L, I+1) :-
    in_list(Item),
    located(Item, L),
    at(L, I),
```

```
    time(I),
    pick(Item, I).

%% PAY
{ pay(I) } :-
    located(cashier, L),
    at(L, I),
    time(I).

% effect
paid(I+1) :-
    time(I),
    pay(I).

at(L, I+1) :-
    located(cashier,L),
    at(L, I),
    time(I),
    paid(I).



%%% FRAME AXIOMS

% we may be only in one place at a time
 :- 2 { at(L, I) : location(L) },
    time(I).

% our position stays the same if we are not moving
at(L, I+1) :-
    at(L, I),
    location(L),
    time(I),
    not moving(I).

% we don't drop picked items
has(Item, I+1) :-
    has(Item, I),
    in_list(Item),
    time(I).
```

```
% once we pay we stay paid
paid(I+1) :-
    paid(I),
    time(I).
```