T-79.154 **Syksy 2004**
Logiikka tietotekniikassa: erityiskysymyksiä II
Laskuharjoitus 6
Ratkaisut

1. Ratkaisussa käytetään predikaatteja:

   - $at(V, L, I)$ — imuri V on huoneessa L hetkellä I.
   - $clean(L, I)$ — huone L on siisti hetkellä I.
   - $move(V, F, T, I)$ — imuri V siirretään huoneesta F huoneeseen T hetkellä I.
   - $suction(V, L, I)$ — imuri V siivoaa huoneen L hetkellä I.
   - $next\_to(L_1, L_2)$ — huone L_2 on huoneen L_1 vieressä.

   Suunniteluoperaattorien mallintamisessa käytetään periaatetta, jonka mukaan tapahtuma implikoi sekä esi- että jälkiehtonsa.

   Operaattorien mallintamiseen tarvitaan seuraavat lauseskeemat:

   (a) **Suction**

   - Jos imuri v siivoaa huoneen l, se muuttuu siistiksi:

     $$\forall v \forall l \forall i(at(v, l, i) \wedge suction(v, l, i) \rightarrow clean(l, i + 1))$$

   - Imuri voi siivota vain huoneen, jossa se on:

     $$\forall v \forall l \forall i(suction(v, l, i) \rightarrow at(v, l, i))$$

   - Imuri ei voi liikkua siivotessaan:

     $$\forall v \forall l \forall i(suction(v, l, i) \rightarrow at(v, l, i + 1))$$

   - Puhtaita huoneita ei siivota:

     $$\forall v \forall l \forall i(suction(v, l, i) \rightarrow \neg clean(l, i))$$

   (b) **Move**

   - Imuri voi siirtyä pois ainoastaan huoneesta, jossa se on:

     $$\forall v \forall f \forall t \forall i(move(v, f, t, i) \rightarrow at(v, f, i))$$

- Imuri siirtyy kohteeseensa:

$$\forall v \forall f \forall t \forall i (move(v, f, t, i) \rightarrow at(v, t, i + 1))$$

- Imuri voi siirtyä ainoastaan viereiseen huoneeseen:

$$\forall v \forall f \forall t \forall i (move(v, f, t, i) \rightarrow next\_to(f, t))$$

Lisäksi tarvitaan koko joukko *kehysaksioomia*, jotka pitävät maailmankuvan konsistenttina ja yllätyksettömänä:

- Imuri ei voi olla yhtä aikaa kahdessa paikassa:

$$\forall v \forall l_1 \forall l_2 \forall i (l_1 \neq l_2 \rightarrow \neg at(v, l_1, i) \vee \neg at(v, l_2, i))$$

- Jos huone muuttuu siistiksi, on sen joku siivonnut:

$$\forall l \forall i (\neg clean(l, i) \wedge clean(l, i + 1) \rightarrow \exists v (suction(v, l, i)))$$

- Jos imuri ilmestyy jonnekin, se on siirtynyt normaalilla tavalla:

$$\forall v \forall t \forall i (\neg at(v, t, i) \wedge at(v, t, i + 1) \rightarrow \exists f (move(v, f, t, i)))$$

Suunnittelun alkutilassa kaikki huoneet ovat likaisia, imureita on vain yksi ja se on olohuoneessa:

$$\forall l (\neg clean(l, 0)) \wedge at(c, oh, 0)$$

Relaatio *next_to* täytyy määritellä kokonaan, eli lausejoukkoon täytyy lisätä $next\_to(x, y)$ aina, kun x:n ja y:n välillä on ovi, ja $\neg next\_to(x, y)$, jos ovea ei ole.

Maalitila on yksinkertaisesti:

$$\forall l (clean(l, g)),$$

missä g on askelten määrä suunnitelmassa.

2. **INSTANTIATE**

Seuraavassa esitetään ongelman koodaus instantiate-ohjelman ymmärtämällä syntaksilla:

```
#option --totalize-negations --dlp --true-negation

%% Pölynimurimaailma toteutettuna lauselogiikalla. Käyttö esim:
%%
%%   instantiate -ct=13 vac.sat instance.dat | gnt2
%%
%% tai vaihtoehtoisesti:
%%
%%   instantiate -ct=13 vac.sat instance.dat | dimacs > dimacs_output_file
%%

#state at(_, _, _).
#state clean(_, _).
#action move(_,_,_,_).
#action suction(_,_,_).

#type location(L).
#type location(L_1).
#type location(L_2).
#type vacuum(V).
#type time(I).
#type next_to(F, T).

time(1 .. t).

%%% SUCTION

%% Imuri voi siivota vain huoneen, jossa se on:
suction(V, L, I) -> at(V, L, I).

%% Puhtaita huoneita ei siivota:
suction(V, L, I) -> -clean(L, I).

%% Imuri ei voi liikkua siivotessaan:
suction(V, L, I) -> at(V, L, I+1).

%% Siivottu huone muuttuu siistiksi
at(V, L, I), suction(V, L, I) -> clean(L, I+1).


%%% MOVE
```

```
% Imuri voi siirtyä pois ainoastaan huoneesta, jossa se on:
move(V, F, T, I) -> at(V, F, I).

% Imuri siirtyy kohteeseensa:
move(V, F, T, I) -> at(V, T, I+1).


%% FRAME AXIOMS

% Imuri ei voi olla yhtä aikaa kahdessa paikassa:
L_1 != L_2 -> -at(V, L_1, I) | -at(V, L_2, I).

% Jos huone muuttuu siistiksi,  on sen joku siivonnut:
-clean(L, I), clean(L, I+1) -> { suction(Y, L, I) : vacuum(Y) }.

% Jos imuri ilmestyy jonnekin, se on siirtynyt normaalilla tavalla:
-at(V, Y, I), location(Y),  at(V, Y, I+1) -> { move(V, X, Y, I) :
                                               next_to(X, Y) }.

% Siivottu huone pysyy siivottuna
clean(L, I) -> clean(L, I+1).

% Jos imuri ei tee mitään, se pysyy paikoillaan.
at(V, L, I), no_op(V, I) -> at(V, L, I+1).

% % Aina tapahtuu jotain:
vacuum(V) -> { move(V, X, Y, I) : next_to(X, Y),
               suction(V, Z, I) : location(Z),
               no_op(V, I) }.

% Alkutila, kaikki huoneet likaisia:
-clean(L, 1).


% Lopputila
compute { clean(X, t+1) : location(X) }.
```