## Suunnittelu toteutuvuusongelmana

- Suunnittelu on esimerkki laskennallisesti haastavasta sovellutusalueesta.
- Erikoisalgoritmit vaikeasti muunneltavissa uusille sovellutusalueille.
- Suunnitteluongelman ratkaiseminen kuvaamalla se sopivaan logiikkaan (päättely- tai toteutuvuusongelmaan) tarjoaa enemmän joustavuutta (lisäehdot, tietämyksen esittäminen).
- Kuvaus saattaa kasvaa suureksi (kehysongelma).
- Suunnittelun kuvaaminen lauselogiikan toteutuvuusongelmaksi antanut laskennallisesti lupaavia tuloksia.



## Suunnitteluongelma

- Olkoon käytössä joukko operaattoreita $Ops$

```
(operator
  MOVE
  (params (<X> OBJECT) (<Y> OBJECT) (<Z> OBJECT))
  (preconds  (clear <X>) (on <X> <Y>) (clear <Z>))
  (effects   (on <X> <Z>)  (clear <Y>) (clear <X>)))
```

- Domain-määrittelyt $Dom$ muuttujille.
- Alkutilanne $S_0$ ja lopputilanne $S_1$ (molemmat joukkoja muuttujattomia atomilauseita).



Sarja toimintoja (operaattorien $Ops$ instansseja) on suunnitteluongelman $\langle Ops, Dom, S_0, S_1\rangle$ *ratkaisu* joss se muuntaa alkutilan $S_0$ tilaksi, johon lopputila $S_1$ sisältyy.

**Esimerkki.** Operaattorit: MOVE

Domain-määrittelyt: $OBJECT : \{a, b, c, d\}$

$S_0 = \{on(a, b), on(b, c), clear(a), clear(d)\}$

$S_1 = \{on(b, a)\}$

Ratkaisu: $MOVE(a, b, d), MOVE(b, c, a)$

$S_0 = \{on(a, b), on(b, c), clear(a), clear(d)\}$

$\overset{MOVE(a,b,d)}{\Longrightarrow} \{on(a, d), on(b, c), clear(a), clear(b)\}$

$\overset{MOVE(b,c,a)}{\Longrightarrow} \{on(b, a), on(a, d), clear(c), clear(b)\}$



- *Rajoitetussa* suunnitteluongelmassa annetaan haetun suunnitelman pituudelle yläraja $n$.
- Yleinen suunnitteluongelma on PSPACE-täydellinen mutta rajoitettu suunnitteluongelma on NP-täydellinen.
- Rajoitetun suunnitteluongelman $\langle P, n\rangle$ käännös toteutuvuus-ongelmaksi on joukko lauselogiikan lauseita $T(P, n)$ s.e. ongelmalla $P$ on korkeintaan $n$ pituinen ratkaisu joss $T(P, n)$ on toteutuva.
- Käännöstä sanotaan konstruktiiviseksi joss on olemassa polynomisen ajan algoritmi, joka laskee $T(P, n)$:n mallista ongelman $\langle P, n\rangle$ ratkaisun (suunnitelman).
- Yleiselle suunnitteluongelmalle ratkaisu rajoitetun avulla esim. puolitushaulla:

  $n = 1, 2, 4, 8$ (rajoitettu suunnitelma löytyi) $6, 7$.

## Lineaarinen käännös

- Jokainen predikaatti ja operaattori varustetaan ylimääräisellä "aika"-argumentilla: $on(X, Y, I), move(X, Y, Z, I)$
- Predikaateille aika-argumentti: $0, \ldots, n$ operaattoreille: $0, \ldots, n-1$
- Tavoitteena lineaarinen ratkaisu: kunakin ajanhetkenä suoritetaan täsmälleen yksi toiminto.
- $noop$-operaattori (säilyttää predikaatin seuraavaan tilaan) siltä varalta, että ratkaisu on lyhyempi kuin $n$.



- Tehdään käännös seuraavasti:

  1. Alkutila on täysin määrätty:
  $\{p(\vec{t}, 0) \mid p(\vec{t}) \in S_0\} \cup \{\neg p(\vec{t}, 0) \mid p(\vec{t}) \notin S_0\}$
  (lopputila ei: otetaan sellaisenaan).

  2. Jos operaattori-instanssi pätee ajanhetkellä $I$, sen esiehdot pätevät hetkellä $I$ ja lisätyt atomilauseet ja poistettujen atomilauseiden negaatiot hetkellä $I+1$.

  3. Kehysaksioomat: Jos operaattori ei lisää eikä poista atomilausetta, säilyy atomilauseen totuusarvo.

  4. Täsmälleen yksi toiminto kullakin ajanhetkellä.



**Esimerkki.** Käännös $T(P, n)$:

1. Alkutila täydennettynä + lopputila:
$\{on(a, b, 0), on(b, c, 0), clear(a, 0), clear(d, 0)\} \cup$
$\{\neg on(b, a, 0), \neg on(a, c, 0), \ldots, \neg clear(b, 0), \neg clear(c, 0)\} \cup$
$\{on(b, a, n)\}$.

2. Operaattorien vaikutukset:
$move(X, Y, Z, I) \rightarrow (clear(X, I) \wedge on(X, Y, I) \wedge clear(Z, I)) \wedge$
$(on(X, Z, I+1) \wedge clear(Y, I+1) \wedge clear(X, I+1) \wedge$
$\neg clear(Z, I+1) \wedge \neg on(X, Y, I+1))$

3. Kehysaksioomat jokaiselle operaattori/predikaatti -parille:
(i) $move(X, Y, Z, I) \rightarrow (clear(X', I) \leftrightarrow clear(X', I+1))$
missä $Y \neq X'$ tai $Z \neq X'$
(ii) $move(X, Y, Z, I) \rightarrow (on(X', Y', I) \leftrightarrow on(X', Y', I+1))$
missä $X \neq X'$ tai $Z \neq Y'$



(iii) $noop(I) \rightarrow (on(X, Y, I) \leftrightarrow on(X, Y, I+1))$
(iv) $noop(I) \rightarrow (clear(X, I) \leftrightarrow clear(X, I+1))$

4. Täsmälleen yksi toiminto kullakin ajanhetkellä:

(i) $\neg move(X, Y, Z, I) \vee \neg move(X', Y', Z', I)$
missä $X \neq X'$ tai $Y \neq Y'$ tai $Z \neq Z'$

(ii) $\neg move(X, Y, Z, I) \vee \neg noop(I)$

(iii) $noop(I) \vee \exists X \exists Y \exists Z move(X, Y, Z, I)$

(Huom! $\exists X\, p(X) : p(d_1) \vee \cdots \vee p(d_m)$)

Kehysaksioomat ja yksikäsitteisyysaksioomat johtavat käännöksen koon nopeaan kasvuun.

**Esimerkki.** Olkoon $n = 1$.

3.) Kehysaksioomat:

$M = \{on(a,b,0), on(b,c,0), clear(a,0), clear(d,0), on(b,a,n),$
$move(a,b,d,0), on(a,d,1), clear(a,1), clear(b,1)\}$.

on lauseiden 1&2 malli: $on(b,c,1)$ ei ole totta!

4.) Ainakin yksi toiminto:

$M = \{on(a,b,0), on(b,c,0), clear(a,0), clear(d,0), on(b,a,n),$
$on(c,d,1)\}$.

on lauseiden 1&2&3 malli: muutos ilman toimintoa!



## Operaattorin jakaminen

- Moniargumenttiset operaattorit $\Rightarrow$ suuri käännös.
- Käännöksen kasvua voidaan hidastaa jakamalla moniargumenttinen operaattori:
  $move(X,Y,Z,I) \leftrightarrow o(X,I) \wedge s(Y,I) \wedge d(Z,I)$
- Nyt voidaan käännös kirjoittaa uudella tavalla. Esim.
  $o(X,I) \wedge s(Y,I) \wedge d(Z,I) \rightarrow$
  $(clear(X,I) \wedge on(X,Y,I) \wedge clear(Z,I)) \wedge$
  $(on(X,Z,I+1) \wedge clear(Y,I+1) \wedge clear(X,I+1) \wedge$
  $\neg clear(Z,I+1) \wedge \neg on(X,Y,I+1))$.

  $s(X,I) \rightarrow (clear(X',I) \leftrightarrow clear(X',I+1))$ missä $X \neq X'$.
  $d(Z,I) \rightarrow (clear(X',I) \leftrightarrow clear(X',I+1))$ missä $Z \neq X'$.
  $\ldots$



## Selittävät kehysaksioomat

- Toinen tapa kirjoittaa kehysaksioomat:
  $f_i \wedge \neg f_{i+1} \rightarrow \bigvee\{a_i \mid f \in Del(a)\}$
  $\neg f_i \wedge f_{i+1} \rightarrow \bigvee\{a_i \mid f \in Add(a)\}$
  Esim. $clear(Z,I) \wedge \neg clear(Z,I+1) \rightarrow move(X,Y,Z,I)$
  (tai $clear(Z,I) \wedge \neg clear(Z,I+1) \rightarrow d(Z,I)$)
  $\neg clear(Y,I) \wedge clear(Y,I+1) \rightarrow move(X,Y,Z,I)$
  (tai $clear(Y,I) \wedge \neg clear(Y,I+1) \rightarrow s(Y,I)$)
- Riittää vaatia, että korkeintaan yksi toiminto kunakin hetkenä.
- $noop$-operaattoria ei tarvita.



## Osittaisjärjestyskäännös

- Sallitaan useita tapahtumia samalla ajanhetkellä.
- Tarvitaan vähemmän ajanhetkiä kuin suunnitelmassa askelia.
  $\Rightarrow$ Käännöksen koko pienenee.
- Saadaan esim. käyttämällä selittäviä kehysaksioomeja ja vaatimalla, että konfliktissa olevat toiminnot eivät voi tapahtua samaan aikaan.
  Esim. $\neg move(X,Y,Z,I) \vee \neg move(X',Y',Z',I)$
  missä $X \neq X'$ tai $Z \neq Z'$
- Suunnitelma saadaan mallista järjestämällä samalla hetkellä tapahtuvat toiminnot mihin tahansa järjestykseen.