

## Suunnittelu toteutuvuusongelmana

- $S_U$ unnitteluongelmat (AI planning problems) muodostavat laskennallisesti haastavan sovellutusalueen.
- Erikoisalgoritmit vaikeasti muunneltavissa uusille sovellutusalueille.
- $S_U$ unnitteluongelman kuvaaminen loogisena päättelyongelmana tarjoaa enemmän joustavuutta (tietämyksen esittäminen).
- Kuvaus saattaa kasvaa suureksi (ns. kehysongelmasta johtuen).
- $S_U$ unnittelun kuvaaminen lauselogiikan toteutuvuusongelmana antanut laskennallisesti lupaavia tuloksia.

## Suunnitteluongelma

$S_U$ unnitteluongelma on nelikkö  $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$ , missä

- Jokaisen *äärellisen tyyppin*  $T \in \mathcal{T}$  alkioit on nimetty yksikäsitteisesti äärellisellä joukolla  $\{a, b, c, \dots\}$  *nimivakioita*.
- Jokainen *operaattori*  $O(x, y, z, \dots) \in \mathcal{O}$  rakentuu
  1. muuttujiensa tyyppimäärittelyistä  $x : T_1, y : T_2, z : T_3, \dots$ , ja
  2. *esiehtojen* ja *jälkiehtojen* joukoista  $\text{Pre}(O)$  ja  $\text{Post}(O)$ , jotka ovat tyyppitetyistä predikaattisymboleista  $P, Q, \dots$ , em. tyyppitetyistä muuttujista  $x, y, z, \dots$  ja tyyppien nimivakioista  $a, b, c, \dots$  rakentuvien atomikaavojen joukkoja.
- *Alkutilanne*  $S_0$  ja *lopputilanne*  $S_1$  ovat (muuttujattomia) atomilauseiden joukkoja.

## Luennon sisältöä

- Määritellään *suunnitteluongelma* yleisessä tapauksessa sekä suunnitteluongelman ratkaisut eli *suunnitelmat*.
- $S_U$ unnitelmien pituuden rajoittaminen.
- Rajoitetun ongelman esittäminen toteutuvuusongelmana.
- Joitain tekniikoita esityksen tehostamiseksi.
- Aiheeseen liittyviä artikkeleita:
  - H. Kautz and B. Selman [ECAI, 1992]: *Planning as Satisfiability*.
  - H. Kautz and B. Selman [AAAI, 1996]: *Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search*.
- Esimerkkitoteutuksen (SATPLAN) esittely (2. kotitehtävä).

## Esimerkki: palikkamaailma

- Tyyppi BLOCK : nimivakiot  $\{a, b, c, d\}$ .
- Operaattori  $\text{MOVE}(x, y, z)$ :
  1. muuttujien  $x, y$  ja  $z$  tyyppinä BLOCK,
  2.  $\text{Pre}(\text{MOVE}(x, y, z)) = \{\text{clear}(x), \text{on}(x, y), \text{clear}(z)\}$  ja
  3.  $\text{Post}(\text{MOVE}(x, y, z)) = \{\text{on}(x, z), \text{clear}(x), \text{clear}(y)\}$ .
- Alkutila  $S_0 = \{\text{on}(a, b), \text{on}(b, c), \text{clear}(a), \text{clear}(d)\}$ .
- Tavoitteen määrittelee lopetusehtojen joukko  $S_1 = \{\text{on}(b, a)\}$ .

## Ratkaisuna suunnitelmat

- Operaattori  $O(x, y, z, \dots) \in \mathcal{O}$  voidaan instantioida korvaamalla muuttujia  $x, y, z, \dots$  vastaavien tyyppien  $T_1, T_2, T_3, \dots$  nimivakioilla  $c_1, c_2, c_3, \dots$   
(kysymyksessä siten *substituutio*  $\sigma = \{x/c_1, y/c_2, z/c_3, \dots\}$ ).
- Yksittäinen *toiminto*  $O\sigma$  (eli operaattorin  $O(\vec{x}) \in \mathcal{O}$  muuttujaton instanssi) voidaan suorittaa, mikäli tilalle  $S$  pätee  $\text{Pre}(O)\sigma \subseteq S$ .  
Toiminnon suorittamisen seurauksena päädytään tilaan  $S' = (S - \text{Pre}(O)\sigma) \cup \text{Post}(O)\sigma$ .  
Tästä käytetään jatkossa merkintää  $S \xrightarrow{O\sigma} S'$ .
- $S_{\text{U}}$  suunnitteluongelman  $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$  *ratkaisu* eli *suunnitelma* on toimintojen sekvenssi  $O_1\sigma_1, \dots, O_n\sigma_n$ , joka muuttaa alkutilan  $S_0$  tilaksi  $S$  (eli  $S_0 \xrightarrow{O_1\sigma_1} \dots \xrightarrow{O_n\sigma_n} S$ ) siten, että  $S_1 \subseteq S$ .

© 2002 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

## Suunnittelun laaskennallisesta vaativuudesta

- $S_{\text{U}}$  suunnitteluongelman ratkaiseminen (eli annetut kriteerit täyttävän suunnitelman hakeminen) on yleisessä tapauksessa varsin hankalaa.
- Tarkastellaan seuraavia päätösongelmia:
  1. PLANSAT: onko syötteenä saadulla suunnitteluongelmalla  $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$  ratkaisua?
  2. PLANMIN: onko syötteenä saadulla suunnitteluongelmalla  $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$  korkeintaan  $k$ :n mittaista ratkaisua?  
(Tässä tapauksessa  $k$  sisältyy syötteeseen).
- PLANSAT ja PLANMIN ovat PSPACE-täydellisiä ongelmia.  
T. Bylander: *The Computational Complexity of Propositional STRIPS Planning* [AIJ 69(1-2), 165–204, 1994].

© 2002 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

## Esimerkki (jatkoa)

- Sekvenssi  $\text{MOVE}(a, b, d), \text{MOVE}(b, c, a)$  antaa suunnitelman:

$$\begin{array}{l} \text{MOVE}(a, b, d) \quad \{\text{on}(a, b), \text{on}(b, c), \text{clear}(a), \text{clear}(d)\} \\ \xrightarrow{\text{MOVE}(a, b, d)} \quad \{\text{on}(a, d), \text{on}(b, c), \text{clear}(a), \text{clear}(b)\} \\ \xrightarrow{\text{MOVE}(b, c, a)} \quad \{\text{on}(b, a), \text{on}(a, d), \text{clear}(c), \text{clear}(b)\}. \end{array}$$

- Em. toimintojen esi- ja jälkiehdot ovat seuraavat:

$$\begin{array}{l} \text{Pre}(\text{MOVE}(a, b, d)) = \{\text{clear}(a), \text{on}(a, b), \text{clear}(d)\} \\ \text{Post}(\text{MOVE}(a, b, d)) = \{\text{on}(a, d), \text{clear}(a), \text{clear}(b)\} \\ \text{Pre}(\text{MOVE}(b, c, a)) = \{\text{clear}(b), \text{on}(b, c), \text{clear}(a)\} \\ \text{Post}(\text{MOVE}(b, c, a)) = \{\text{on}(b, a), \text{clear}(b), \text{clear}(c)\} \end{array}$$

© 2002 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

- Pahimmassa tapauksessa suunnitelman pituus voi olla eksponentiaalinen suunnitteluongelman kokoon nähden.
- Yleiselle suunnitteluongelmalle voidaan hakea ratkaisua käyttämällä ratkaisuproseduuria rajoitetulle ongelmalle.
  1. Haetaan systemaattisesti pidempiä ja pidempiä suunnitelmia.  
 $k = 1, 2, 3, 4, 5, 6, 7, 8$  (suunnitelma löytyi)
  2. Tehdään puolitushakua suunnitelman pituuden  $k$  suhteen.  
 $k = 1, 2, 4, 8$  (suunnitelma löytyi)  
 $k = 6, 7$  (ei suunnitelmia, joten minipituus on 8)

**Huomio.** Kompleksisuutta voidaan alentaa rajoittamalla suunnitelmien pituus suunnitteluongelman kokoon nähden polynomiseksi (syötteen annettava yläraja  $k$  esitetään 1-kantaisena).



Rajoitettu suunnitteluongelma on enää NP-täydellinen.

© 2002 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

## Rajoitettu suunnittelu toteutuvuusongelmana

- Kompleksisuustulosten nojalla rajoitetun suunnitteluongelman instanssi voidaan redusoida polynomisessa ajassa lauselogiikan toteutuvuusongelman instanssiksi.
- Rajoitetun suunnitteluongelman  $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$  esitys toteutuvuusongelmana on joukko lauselogiikan lauseita  $PS(k)$  s.e. ongelmalla  $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$  on enintään  $k$ :n mittainen ratkaisu  $\iff PS(k)$  on toteutuva.
- Esitystä sanotaan *konstrukttiiviseksi*, jos on olemassa polynomisen ajan algoritmi, joka eristää  $PS(k)$ :n mallista rajoitetulle suunnitteluongelmalle  $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$  ratkaisun (suunnitelman).

## Kuvauksen rakenne

Suunnitteluongelman esitys jakautuu seuraaviin osiin:

1. Predikaattien totuusarvojen määrittäminen hetkellä  $t = 0$ :

$$\{P(\vec{c}, 0) \mid P(\vec{c}) \in S_0\} \cup \{\neg P(\vec{c}, 0) \mid P(\vec{c}) \notin S_0\}.$$

2. Predikaattien totuusarvot hetkellä  $t = n$ :  $\{P(\vec{c}, n) \mid P(\vec{c}) \in S_1\}$ .
3. Jokaisella ajanhetkellä  $t$  suoritetaan täsmälleen yksi toiminto  $O_t \sigma_t$ .
4. Toiminnon  $O_t \sigma_t$  esiehdot  $\text{Pre}(O_t) \sigma_t$  ovat tosia hetkellä  $t$ . Samoin lisätyt atomilauseet  $(\text{Post}(O_t) \sigma_t - \text{Pre}(O_t) \sigma_t)$  sekä poistettujen atomilauseiden  $(\text{Pre}(O_t) \sigma_t - \text{Post}(O_t) \sigma_t)$  negaatiot hetkellä  $t + 1$ .
5. *Kehysaksioimat*: Jos toiminto  $O_t \sigma_t$  ei muuta atomilauseen  $P(\vec{c})$  totuusarvoa ajanhetkellä  $t$ , se säilyy ennallaan ajanhetkeen  $t + 1$ .

## Lineaariset suunnitelmat

- Annetaan määritelmän mukaisille suunnitteluongelman ratkaisuille  $O_1 \sigma_1, \dots, O_n \sigma_n$  rajoitteet lauselogiikan lausein.
- Keskeisenä ideana on käyttää *lineaarista* ajan käsitettä: jokaisena ajanhetkenä  $0, 1, \dots, n - 1$  suoritetaan täsmälleen yksi toiminto.
- Jokainen predikaatti ja operaattori varustetaan ylimääräisellä aika-argumentilla  $t$ :  $\text{on}(x, y, t)$ ,  $\text{MOVE}(x, y, z, t)$ .
- Predikaateilla aika-argumentit  $t$  ovat väliltä  $0, \dots, n$  ja operaattoreilla väliltä  $0, \dots, n - 1$ .
- Lisäksi tarvitaan NOOP-operaattori (ei muuta predikaatteja) siltä varalta, että ratkaisu saavutetaan ennen ajanhetkeä  $n$ .

## Esimerkki: palikkamaailman esitys

1. Ajanhetken  $t = 0$  kuvaus:

$$\{\text{on}(a, b, 0), \text{on}(b, c, 0), \text{clear}(a, 0), \text{clear}(d, 0)\} \cup \\ \{\neg \text{on}(b, a, 0), \neg \text{on}(a, c, 0), \dots, \neg \text{clear}(b, 0), \neg \text{clear}(c, 0)\}.$$

2. Ajanhetken  $t = n$  kuvaus:  $\{\text{on}(b, a, n)\}$ .
3. Jokaisella ajanhetkellä  $t$  suoritetaan täsmälleen yksi toiminto:
  - (i)  $\neg \text{MOVE}(x, y, z, t) \vee \neg \text{MOVE}(x', y', z', t)$ , missä  $x \neq x'$  tai  $y \neq y'$  tai  $z \neq z'$ .
  - (ii)  $\neg \text{MOVE}(x, y, z, t) \vee \neg \text{NOOP}(t)$
  - (iii)  $\text{NOOP}(t) \vee \exists x \exists y \exists z \text{ MOVE}(x, y, z, t)$

**Huomio.**  $\exists x P(x)$  tarkoittaa disjunktioita  $P(a_1) \vee \dots \vee P(a_n)$ , missä vakiot  $a_1, \dots, a_n$  nimeävät muuttujan  $x$  tyypin.

## 4. Toimintojen suorittamisen ehdot ja vaikutukset:

$$\text{MOVE}(x, y, z, t) \rightarrow$$

$$\text{clear}(x, t) \wedge \text{on}(x, y, t) \wedge \text{clear}(z, t) \wedge$$

$$\text{on}(x, z, t + 1) \wedge \text{clear}(y, t + 1) \wedge \text{clear}(x, t + 1) \wedge$$

$$\neg \text{clear}(z, t + 1) \wedge \neg \text{on}(x, y, t + 1)$$

## 5. Jokaista operaattori/predikaatti-paria koskevat kehysaksiomat:

(i)  $\text{MOVE}(x, y, z, t) \rightarrow (\text{clear}(x', t) \leftrightarrow \text{clear}(x', t + 1))$ ,  
missä  $y \neq x'$  tai  $z \neq x'$ .

(ii)  $\text{MOVE}(x, y, z, t) \rightarrow (\text{on}(x', y', t) \leftrightarrow \text{on}(x', y', t + 1))$ ,  
missä  $x \neq x'$  tai  $z \neq y'$ .

(iii)  $\text{NOOP}(t) \rightarrow (\text{on}(x, y, t) \leftrightarrow \text{on}(x, y, t + 1))$

(iv)  $\text{NOOP}(t) \rightarrow (\text{clear}(x, t) \leftrightarrow \text{clear}(x, t + 1))$

**Tekniikoita kuvauksen tehostamiseksi**

- Kehysaksiomat ja yksikäsitteisyysaksiomat johtavat suunnitteluongelman kuvauksen koon nopeaan kasvuun.
- Kuvauksen kokoa voidaan rajoittaa:
  1. jakamalla operaattoreita osiin ja
  2. käyttämällä selittäviä kehysaksiomia.
- Lisäksi suunnitelmien löytämistä voidaan helpottaa luopumalla suunnitelmien lineaarisuudesta: sallitaan toisistaan riippumattomien operaattorien rinnakkainen suorittaminen

**Esimerkki (jatkoa)**

Tarkastellaan tapausta  $n = 1$  (vain yhtä operaattoria sovelletaan).

## 1. Kohdan 3 (yksikäsitteisyys) lauseita tarvitaan, koska muutoin esim.

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

olisi muiden lauseiden malli (muutoksia ilman toimintoa).

## 2. Kehysaksiomia tarvitaan, koska muutoin esim.

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

olisi muiden lauseiden malli:  $\text{on}(b, c, 1)$  on epätosi.

**Operaattorin jakaminen**

- Moniargumenttiset operaattorit kasvattavat kuvauksen kokoa. Kuvauksen koon kasvua voidaan hillitä jakamalla moniargumenttiset operaattorit osiin seuraavalla tavalla:

$$\text{MOVE}(x, y, z, t) \leftrightarrow \text{OBJ}(x, t) \wedge \text{SRC}(y, t) \wedge \text{DST}(z, t).$$

- Tämän avulla kuvaus voidaan kirjoittaa uudella tavalla:

$$\text{OBJ}(x, t) \wedge \text{SRC}(y, t) \wedge \text{DST}(z, t) \rightarrow$$

$$\text{clear}(x, t) \wedge \text{on}(x, y, t) \wedge \text{clear}(z, t) \wedge$$

$$\text{on}(x, z, t + 1) \wedge \text{clear}(y, t + 1) \wedge \text{clear}(x, t + 1) \wedge$$

$$\neg \text{clear}(z, t + 1) \wedge \neg \text{on}(x, y, t + 1).$$

$$\text{SRC}(x, t) \rightarrow (\text{clear}(x', t) \leftrightarrow \text{clear}(x', t + 1)), \text{ missä } x \neq x'.$$

$$\text{DST}(z, t) \rightarrow (\text{clear}(x', t) \leftrightarrow \text{clear}(x', t + 1)), \text{ missä } z \neq x'.$$

...

## Selittävät kehysaksiomat

- On myös toinen tapa kirjoittaa kehysaksiomat predikaatille  $P(\vec{x})$ :

$$P(\vec{x}, t) \wedge \neg P(\vec{x}, t + 1) \rightarrow \bigvee \{O\sigma \mid P(\vec{x}) \in (\text{Pre}(O)\sigma - \text{Post}(O)\sigma)\}$$

$$\neg P(\vec{x}, t) \wedge P(\vec{x}, t + 1) \rightarrow \bigvee \{O\sigma \mid P(\vec{x}) \in (\text{Post}(O)\sigma - \text{Pre}(O)\sigma)\}$$

**Esimerkki.** Palikkamaailman tapauksessa esim.

$$\text{clear}(z, t) \wedge \neg \text{clear}(z, t + 1) \rightarrow \exists x \exists y \text{MOVE}(x, y, z, t)$$

(tai  $\text{clear}(z, t) \wedge \neg \text{clear}(z, t + 1) \rightarrow \text{DST}(z, t)$ ) ja

$$\neg \text{clear}(y, t) \wedge \text{clear}(y, t + 1) \rightarrow \exists x \exists z \text{MOVE}(x, y, z, t)$$

(tai  $\text{clear}(y, t) \wedge \neg \text{clear}(y, t + 1) \rightarrow \text{SRC}(y, t)$ ).

- Riittää, että jokaisena ajanhetkenä  $0, 1, \dots, n - 1$  suoritetaan korkeintaan yksi toiminto: NOOP-operaattoria ei tarvita.

## SATPLAN

- Koostuu H. Kautzin ja B. Selmanin kehittämistä työkaluista suunnitteluongelmien ratkaisemiseksi toteutuvuusongelman kautta.
- Helpottaa käännöksen kirjoittamista tarjoamalla C-kirjaston, jolla voi generoida muuttujallisista klausuuliskeemoista lauselogiikan klausuuleja, jotka automaattisesti muunnetaan SAT-toteutusten (ntab, relsat, satz, chaff, walksat) edellyttämään muotoon.
- Käytössä ns. DIMACS-formaatti, jossa klausuuleille (kuten esim.  $\neg A \vee \neg B \vee C$ ) annetaan numeroesitys (-1 -2 3 0).
- Löytyy Atk-keskuksen koneista asennettuna hakemistosta `~tssyrjan/Tik-79.154/satplan` (tarkempi ohjeita tiedostossa README sekä alihakemistossa doc).

## Osittaisjärjestyskäännös

- Sallitaan useita rinnakkaisia ja toisistaan riippumattomia toimintoja samalla ajanhetkellä.
- Tarvitaan vähemmän ajanhetkiä kuin varsinaisessa suunnitelmassa on askelia, joten kuvauksen koko pienenee.
- Saadaan mm. käyttämällä selittäviä kehysaksiomia ja vaatimalla, että toisistaan riippuvat toiminnot eivät voi tapahtua samanaikaisesti.

**Esimerkki.**  $\neg \text{MOVE}(x, y, z, t) \vee \neg \text{MOVE}(x', y', z', t)$ , missä  $x \neq x'$  tai  $z \neq z'$ .

- Suunnitelma saadaan mallista järjestämällä samalla hetkellä tapahtuvat toiminnot mihin järjestykseen tahansa.

**Esimerkki.** SATPLAN edellyttää seuraaventyypistä syntaksia:

```
/* ∀x ∈ block – fixed : ∀z ∈ block, z ≠ x :
  (¬object(x, i) ∨ ¬destination(z, i) ∨ on(x, z, i + 1)) */
All(x, block, !member(x, fixed),
  All(z, block, !eql(x, z),
    Disj(Not(L2("object", x, i);
      Not(L2("destination", z, i);
        L3("on", x, z, i + 1)))));
/* ∃x ∈ block – fixed : object(x, i) */
Disj(All(x, block, !member(x, fixed),
  L2("object", x, i));
```

## SATPLAN-järjestelmän tuottamat tiedostot

- `bw_orig.ops`: Operaattorien (blocks world) ja tilojen määritelmät.
- `anomaly.facts`  
Määrittelee muuttujien arvoalueet, suunnitelman pituuden, sekä alku- ja lopputilat.
- `anomaly.cnf`: klausuuliesitys DIMACS-formaatissa.
- `anomaly.map`: symbolitaulu
- `anomaly.out`: tulosteet (totuusarvojakelu ja statistiikkaa)
- `anomaly.interp`: suunnitelma selväkielisenä