Suunnittelu toteutuvuusongelmana

Käsiteltäviä asioita:

1. Suunnitteluongelmat
2. Suunniteluongelman laskennallinen vaativuus
3. Rajoitettu suunnittelu toteutuvuusongelmana
4. Tekniikoita kuvauksen tehostamiseksi

1. Suunnitteluongelmat

- Suunnitteluogelmat (AI planning problems) muodostavat laskennallisesti haastavan sovellutusalueen.
- Erikoisalgoritmit vaikeasti muunneltavissa uusille sovellutusalueille.
- Suunnitteluongelman kuvaaminen logisena päättelyongelmana tarjoaa enemmän joustavuutta (tietämyksen esittäminen).
- Kuvaus saattaa kasvaa suureksi (ns. kehysongelmasta johtuen).
- Suunnittelun kuvaaminen lauselogiikan toteutuvuusongelmana antanut laskennallisesti lupaavia tuloksia.

Formaali määritelmä

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

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

Esimerkki: palikkamaailma

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

Ratkaisuna suunnitelmat

- Operaattori $O(x,y,z,\ldots) \in \mathcal{O}$ voidaan instantioida korvaamalla muuttujia $x,y,z,\ldots$ vastaavien tyyppien $T_1,T_2,T_3,\ldots$ nimivakioilla $c_1,c_2,c_3,\ldots$
(kysymyksessä siten *substituutio* $\sigma = \{x/c_1, y/c_2, z/c_3, \ldots\}$).
- Yksittäinen *toiminto* $O\sigma$ (eli operaattorin $O(\vec{x}) \in \mathcal{O}$ muuttujaton instanssi) voidaan suorittaa, mikäli tilalle S pätee $\mathrm{Pre}(O)\sigma \subseteq S$. Toiminnon suorittamisen seurauksena päädytään tilaan $S' = (S - \mathrm{Pre}(O)\sigma) \cup \mathrm{Post}(O)\sigma$.

  Tästä käytetään jatkossa merkintää $S \stackrel{O\sigma}{\Longrightarrow} S'$.
- Suunnitteluongelman $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$ *ratkaisu* eli *suunnitelma* on toimintojen sekvenssi $O_1\sigma_1, \ldots, O_n\sigma_n$, joka muuttaa alkutilan S_0 tilaksi S (eli $S_0 \stackrel{O_1\sigma_1}{\Longrightarrow} \cdots \stackrel{O_n\sigma_n}{\Longrightarrow} S$) siten, että $S_1 \subseteq S$.

Esimerkki (jatkoa)

- Sekvenssi MOVE(a,b,d), MOVE(b,c,a) antaa suunnitelman:

$$
\begin{aligned}
& \{\mathrm{on}(a,b), \mathrm{on}(b,c), \mathrm{clear}(a), \mathrm{clear}(d)\} \\
\stackrel{\mathrm{MOVE}(a,b,d)}{\Longrightarrow} & \{\mathrm{on}(a,d), \mathrm{on}(b,c), \mathrm{clear}(a), \mathrm{clear}(b)\} \\
\stackrel{\mathrm{MOVE}(b,c,a)}{\Longrightarrow} & \{\mathrm{on}(b,a), \mathrm{on}(a,d), \mathrm{clear}(c), \mathrm{clear}(b)\}.
\end{aligned}
$$

- Em. toimintojen esi- ja jälkiehdot ovat seuraavat:

$$
\begin{aligned}
\mathrm{Pre}(\mathrm{MOVE}(a,b,d)) &= \{\mathrm{clear}(a), \mathrm{on}(a,b), \mathrm{clear}(d)\} \\
\mathrm{Post}(\mathrm{MOVE}(a,b,d)) &= \{\mathrm{on}(a,d), \mathrm{clear}(a), \mathrm{clear}(b)\} \\
\mathrm{Pre}(\mathrm{MOVE}(b,c,a)) &= \{\mathrm{clear}(b), \mathrm{on}(b,c), \mathrm{clear}(a)\} \\
\mathrm{Post}(\mathrm{MOVE}(b,c,a)) &= \{\mathrm{on}(b,a), \mathrm{clear}(b), \mathrm{clear}(c)\}
\end{aligned}
$$

2. Suunnittelun laskennallinen vaativuus

- Suunnitteluongelman ratkaiseminen (eli annetut kriteerit täyttävän suunnitelman hakeminen) on yleisessä tapauksessa varsin hankalaa.
- Tarkastellaan seuraavia päätösongelmia:
  1. PLANSAT: onko syötteeksi saadulla suunnitteluongelmalla $\langle \mathcal{T}, \mathcal{O}, S_0, S_1 \rangle$ ratkaisua?
  2. PLANMIN: onko syötteeksi 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].

- 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 mininipituus on 8)

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

☞ Rajoitettu suunnitteluongelma on enää NP-täydellinen.

3. Rajoitettu suunnittelu toteutuvuusongelmana

- Kompleksisuustulosten nojalla rajoitetun suunnitteluongelman instanssi voidaan redusoida polynomisessa ajassa lauselogiikan toteutuvuusongelman instanssiksi.
- Rajoitetun suunnitteluongelman $\langle \mathcal{T}, O, S_0, S_1 \rangle$ esitys toteutuvuusongelmana on joukko lauselogiikan lauseita $PS(k)$ s.e.

  ongelmalla $\langle \mathcal{T}, O, S_0, S_1 \rangle$ on enintään k:n mittainen ratkaisu

  $\iff$ $PS(k)$ on toteutuva.
- Esitystä sanotaan *konstruktiviseksi*, jos on olemassa polynomisen ajan algoritmi, joka eristää lausejoukon $PS(k)$ mallista rajoitetulle suunnitteluongelmalle $\langle \mathcal{T}, O, S_0, S_1 \rangle$ ratkaisun (suunnitelman).

Lineaariset suunnitelmat

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

Kuvauksen rakenne

Suunnitteluongelman esitys jakautuu seuraaviin osiin:

1. Predikaattien totuusarvojen määrääminen 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 $\mathrm{Pre}(O_t)\sigma_t$ ovat tosia hetkellä t. Samoin lisätyt atomilauseet $(\mathrm{Post}(O_t)\sigma_t - \mathrm{Pre}(O_t)\sigma_t)$ sekä poistettujen atomilauseiden $(\mathrm{Pre}(O_t)\sigma_t - \mathrm{Post}(O_t)\sigma_t)$ negaatiot hetkellä $t+1$.
5. *Kehysaksiomat*: Jos toiminto $O_t\sigma_t$ ei muuta atomilauseen $P(\vec{c})$ totuusarvoa ajanhetkellä t, se säilyy ennallaan ajanhetkeen $t+1$.

Esimerkki: palikkamaailman esitys

1. Ajanhetken $t = 0$ kuvaus:

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

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

Huomio. $\exists x P(x)$ tarkoittaa disjunktiota $P(a_1) \vee \cdots \vee P(a_n)$, missä vakiot $a_1, \ldots, a_n$ nimeävät muuttujan x tyyppiin kuuluvat alkiot.

4. Toimintojen suorittamisen ehdot ja vaikutukset:

$$\begin{aligned}&\mathsf{MOVE}(x,y,z,t) \rightarrow\\ &\mathsf{clear}(x,t) \land \mathsf{on}(x,y,t) \land \mathsf{clear}(z,t) \land\\ &\mathsf{on}(x,z,t+1) \land \mathsf{clear}(y,t+1) \land \mathsf{clear}(x,t+1) \land\\ &\neg\mathsf{clear}(z,t+1) \land \neg\mathsf{on}(x,y,t+1)\end{aligned}$$

5. Jokaista operaattori/predikaatti-paria koskevat kehysaksiomat:
   (i) $\mathsf{MOVE}(x,y,z,t) \rightarrow (\mathsf{clear}(x',t) \leftrightarrow \mathsf{clear}(x',t+1))$, missä $x' \neq y$ ja $x' \neq z$.
   (ii) $\mathsf{MOVE}(x,y,z,t) \rightarrow (\mathsf{on}(x',y',t) \leftrightarrow \mathsf{on}(x',y',t+1))$, missä $x' \neq x$ tai vaihtoehtoisesti $y' \neq y$ ja $y' \neq z$.
   (iii) $\mathsf{NOOP}(t) \rightarrow (\mathsf{on}(x,y,t) \leftrightarrow \mathsf{on}(x,y,t+1))$.
   (iv) $\mathsf{NOOP}(t) \rightarrow (\mathsf{clear}(x,t) \leftrightarrow \mathsf{clear}(x,t+1))$.

Esimerkki (jatkoa)

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

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

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

   olisi muiden lauseiden malli (muutoksia ilman toimintoa).

2. Kehysaksiomia tarvitaan, koska muutoin esim.

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

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

4. 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

Operaattorin jakaminen

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

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

- Nyt MOVE-operaattorin kuvaus voidaan kirjoittaa tiiviimmin:

$$\begin{aligned}&\mathsf{OBJ}(x,t) \rightarrow \mathsf{clear}(x,t) \land \mathsf{clear}(x,t+1)\\ &\mathsf{SRC}(y,t) \rightarrow \mathsf{clear}(y,t+1)\\ &\mathsf{DST}(z,t) \rightarrow \mathsf{clear}(z,t) \land \neg\mathsf{clear}(z,t+1)\\ &\mathsf{OBJ}(x,t) \land \mathsf{SRC}(y,t) \rightarrow \mathsf{on}(x,y,t) \land \neg\mathsf{on}(x,y,t+1)\\ &\mathsf{OBJ}(x,t) \land \mathsf{DST}(z,t) \rightarrow \mathsf{on}(x,z,t+1)\\ &\mathsf{SRC}(y,t) \land \mathsf{DST}(z,t) \land x \neq y \land x \neq z \rightarrow (\mathsf{clear}(x,t) \leftrightarrow \mathsf{clear}(x,t+1))\\ &\ldots\end{aligned}$$

Selittävät kehysaksiomat

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

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

Esimerkki. Palikkamaailman tapauksessa esim.

$$\mathsf{clear}(z,t) \wedge \neg\mathsf{clear}(z,t+1) \rightarrow \exists x \exists y \mathsf{MOVE}(x,y,z,t)$$
$$(\text{tai } \mathsf{clear}(z,t) \wedge \neg\mathsf{clear}(z,t+1) \rightarrow \mathsf{DST}(z,t)) \text{ ja}$$
$$\neg\mathsf{clear}(y,t) \wedge \mathsf{clear}(y,t+1) \rightarrow \exists x \exists z \mathsf{MOVE}(x,y,z,t)$$
$$(\text{tai } \mathsf{clear}(y,t) \wedge \neg\mathsf{clear}(y,t+1) \rightarrow \mathsf{SRC}(y,t)).$$

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

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

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

Oppimistavoitteet

- Ymmärrät suunnitteluongelman ja sen ratkaisujen määritelmät
- Tunnet suunnittelun laskennallisen vaativuuden perustulokset
- Osaat ratkaista laajemman suunnitteluongelman lauselogiikan toteutuvuusongelman kautta (2. kotitehtävä)