

Lause (Cook-Levin) *Kieli*

$SAT = \{ \varphi \mid \varphi \text{ on toteutuva lausekalkyylin kaava} \}$
on *NP-täydellinen*.

- Pitää osoittaa siis, että $A \leq_m^p SAT$ mielivaltaisella $A \in NP$
- Ainoa, mitä A :sta tiedetään on, että sillä on polynomisessa ajassa toimiva epädeterministinen tunnistaja N
- Palautusfunktio muuntaa koneen N mahdollisen syötteen w lausekalkyylin kaavaksi φ_w , joka kuvaa N :n mahdollisia laskentoja syötteellä w
- φ_w on toteutuva joss $w \in L(N) = A$
- Kutakin N :n mahdollista laskentaa vastaa yksi φ_w :n muuttujien totuusarvoasetus
- Kaava φ_w muodostetaan ilmaisemaan ne ehdot, joilla annettu totuusarvoasetus vastaa N :n hyväksyvää laskentaa

Kertaus

1. Luokan P ongelmat ovat käytännössä ratkeavia
2. $P \subseteq NP$. Lisäksi NP :ssä on ongelmia, joille ei tunneta polynomista ratkaisualgoritmia
3. Kaikki luokan NP ongelmat voidaan palauttaa NP -täydelliseen ongelmaan polynomisesti. Jos jokin NP -täydellinen ongelma kuuluu P :hen, niin $P = NP$.
4. Ongelman $A \in NP$ todistaminen NP -täydelliseksi:
 - a) Valitse samankaltainen tunnetusti NP -täydellinen ongelma B
 - b) Muodosta polynominen palautus $f: B \leq_m^p A$; lemmän 7.36 perusteella myös A on NP -täydellinen

Seuraus: *CSAT*, *3SAT* ja *VC* ovat *NP*-täydellisiä ongelmia.

Riippumaton joukko, IS: Annettuna suuntaamaton verkko G ja luonnollinen luku k . Onko G :ssä vähintään k solmua, joiden välillä ei kulje yhtään kaarta?

Seuraavan lemmän perusteella on helppo muodostaa palautukset $VC \leq_m^p IS$ ja $IS \leq_m^p KLIKKI$

Lemma Olkoon $G = (V, E)$ suuntaamaton verkko ja $V' \subseteq V$. Tällöin seuraavat ehdot ovat ekvivalentit:

1. V' on G :n solmupeite,
2. $V \setminus V'$ on riippumaton solmujoukko ja
3. $V \setminus V'$ on klikki G :n komplementtiverkossa $\check{G} = (V, (V \times V) \setminus E)$ \square

$VC \leq_m^p IS$:

Valitaan kuvaus $f: \langle G, k \rangle \mapsto \langle G, |V| - k \rangle$.

Selvästi tämä muunnos voidaan laskea polynomisessa ajassa.

Nyt edellisen lemmän perusteella

$$\langle G, k \rangle \in VC \Leftrightarrow \langle G, |V| - k \rangle \in IS.$$

Täten $f: VC \leq_m^p IS$.

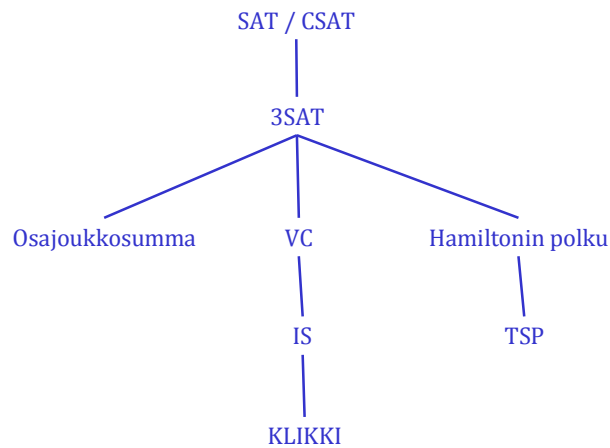
$IS \leq_m^p KLIKKI$:

Valitaan kuvaus $f: \langle G, k \rangle \mapsto \langle \check{G}, k \rangle$.

Kuvaus voidaan laskea polynomisessa ajassa. Edellisen lemmän perusteella

$$\langle G, k \rangle \in IS \Leftrightarrow \langle \check{G}, k \rangle \in KLIKKI.$$

Täten $f: IS \leq_m^p KLIKKI$.



8. Tilavaativuus

- Standardimallisen Turingin koneen

$M = (Q, \Sigma, \Gamma, \delta, q_0, \text{accept}, \text{reject})$ laskennan tilavaativuus syötteellä w on

$$\text{space}_M(w) = \max\{ |uav| : q_0 w \ggg_M u q av, q \in Q, u, a, v \in \Gamma^* \}$$

- Epädeterministisen Turingin koneen

$N = (Q, \Sigma, \Gamma, \delta, q_0, \text{accept}, \text{reject})$ laskennan tilavaativuus:

- $\text{space}_N(w) =$ eniten tilaa vievän laskennan $q_0 w \ggg_N \dots$ vaatimien nauhapaikkojen määrä

- Olkoon $s: \mathbb{N} \rightarrow \mathbb{R}^+$ mv. funktio
- Formaalien kielten deterministinen tilavaativuusluokka on:

$$DSPACE(s) = \{ A \mid A \text{ voidaan tunnistaa tilassa } s \}$$
- Epädeterministinen tilavaativuusluokka puolestaan on

$$NSPACE(s) = \{ A \mid A \text{ voidaan tunnistaa epädet. tilassa } s \}$$
- Tilaa voidaan uudelleenkäyttää, kun taas aikaa ei
- Esim. SAT:in tapauksen ratkaiseminen vaatii vain lineaarisen tilan, vaikka se luultavasti ei NP-täydellisenä ongelmana ole polynomisen ajan puitteissa ratkaistavissa

- Tilavaativuuden yhdisteluokat ovat

$$\begin{aligned} PSPACE &= \bigcup \{ DSPACE(s) \mid s \text{ on polynomi} \} \\ &= \bigcup_{k \geq 0} DSPACE(n^k) \end{aligned}$$

$$EXSPACE = \bigcup_{k \geq 0} DSPACE(2^{n^k})$$

$$NSPACE = \bigcup_{k \geq 0} NSPACE(n^k)$$

$$NEXSPACE = \bigcup_{k \geq 0} NSPACE(2^{n^k})$$

Lemma *Kaikilla* $t(n), s(n) \geq n$:

1. $DTIME(t(n)) \subseteq DSPACE(t(n))$ ja
2. $DSPACE(s(n)) \subseteq \bigcup_{k \geq 0} DTIME(k^{s(n)})$

Todistus.

1. $t(n)$:ssä askelessa Turingin kone voi kirjoittaa korkeintaan $t(n)$ merkkiä työnauhalleen.
2. Koneella, joka toimii tilassa $s(n)$, on n :n merkin syötteellä enintään $k^{s(n)}$ mahdollista tilannetta (k vakio). Antamalla laskennan jatkua korkeintaan $k^{s(n)}$ askelta, saadaan kone toimimaan aina ajassa $k^{s(n)}$. \square

Seuraus: $P \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE$

Savitchin lause

Lause 8.5 *Olkoon* $f: \mathbb{N} \rightarrow \mathbb{R}^+$ *mv. funktio s.e.* $f(n) \geq n$. *Tällöin*
 $NSPACE(f(n)) \subseteq DSPACE(f^2(n))$.

Todistus. Epädeterminististä konetta on siis simuloitava deterministisellä. Suoraviivaisin tapa on käydä kukin laskentapuun haara kerrallaan läpi. Haara, joka käyttää $f(n)$ tilan voi kuitenkin sisältää $2^{O(f(n))}$ askelta ja kussakin askelessa voi olla epädeterministinen valinta. Jotta seuraava haara voidaan saavuttaa, on vaihtoehtoista pidettävä kirjaa. Vaaditaan siis pahimmillaan $2^{O(f(n))}$ tila, joten tämä menetelmä ei käy.

Tarkastellaankin *johtamisongelmaa*: voiko epädeterministinen kone N päästä tilanteesta c_1 tilanteeseen c_2 askelten määrällä t .

Asettamalla c_1 N :n alkutilanteeksi ja c_2 sen hyväksyväksi lopputilaksi, ja ratkaisemalla ongelma deterministisesti käyttämättä liikaa tilaa, voidaan todeta lauseen pätevän.

Etsitään N :n keskimäinen tilanne c_m s.e.

1. c_1 johtaa tilanteeseen c_m askelten määrällä $t/2$ ja
2. c_m johtaa tilanteeseen c_2 askelten määrällä $t/2$.

Rekursiivisten kutsujen vaatima tila voidaan uudelleenkäyttää.

Rekursiopinon tallettamiseen tarvitaan tilaa. Joka tasolla käytetään $O(f(n))$ tila tilanteen tallettamiseen. Rekursiopinon maksimikorkeus on logaritminen epädeterministisen koneen pisimmän laskentapolun pituuden suhteen, eli $\log(2^{O(f(n))}) = O(f(n))$. Deterministinen simulointi vaatii siis $O(f^2(n))$ tilan. \square

Seuraus

1. $\text{NPSPACE} = \text{PSPACE}$
2. $\text{NEXPSPACE} = \text{EXSPACE}$ \square

- Nyt tiedämme yhdistevaativuusluokista lineaarisen järjestyksen

$$P \subseteq NP \subseteq \left\{ \begin{array}{l} \text{PSPACE} = \\ \text{NPSPACE} \end{array} \right\} \subseteq \text{EXPTIME} \subseteq$$

$$\text{NEXPTIME} \subseteq \left\{ \begin{array}{l} \text{EXSPACE} = \\ \text{NEXPSPACE} \end{array} \right.$$

Lause

1. $P \neq EXPTIME$
2. $PSPACE \neq EXPSPACE$ \square

- "Eksponentiaalisen etäällä" toisistaan olevat vaativuusluokat tiedetään toisistaan eroaviksi
 $P \neq EXPTIME$, $NP \neq NEXPTIME$ ja $PSPACE \neq EXPSPACE$
- Sen sijaan ei tiedetä pätevätkö
 $P \neq NP$, $NP \neq PSPACE$, $PSPACE \neq EXPTIME$, ...
- Uskotaan, että kaikki nämä epäyhtälöt ovat tosia, mutta tuloksia ei osata todistaa
- Voidaan osoittaa, että ei ole olemassa "vaikeinta" rekursiivista kieltä

Lause

1. Jos $A \in REC$, niin on olemassa rekursiivinen funktio $t(n)$, jolla $A \in DTIME(t(n))$.
2. Kaikilla rekursiivisilla funktioilla $t(n)$ on olemassa kieli $A \in REC \setminus DTIME(t(n))$.

Funktio $f(n) = f'(n, n)$ on rekursiivinen

$$f(0, n) = 2^{2^{n-2}} \text{ kpl}$$

$$f(m+1, n) = 2^{2^{n-2}} f'(m, n) \text{ kpl}$$

$$f(0) = 1, f(1) = 4, f(2) = 2^{2^{2-2}} = 65\,536 \text{ kpl}$$

Ed. lause: $A \in REC \setminus DTIME(f(n)) \neq \emptyset$

Siis on olemassa "ratkeavia" ongelmia, jotka n merkin syötteillä vaativat enemmän aikaa kuin funktion $f(n)$ verran.

8.1 PSPACE-täydellisyys

- Kieli B on *PSPACE-täydellinen*, jos
 1. $B \in \text{PSPACE}$ ja
 2. $A \leq_m^p B$ kaikilla $A \in \text{PSPACE}$
- Jos B täyttää vain ehdon 2, niin sitä nimitetään *PSPACE-kovaksi*
- Täysin kvantifioidussa Boolean kaavassa kaikki muuttujat ovat jonkin kvantorin vaikutuspiirissä, esim.

$$\varphi = \forall x \exists y [(x \vee y) \wedge (\neg x \vee \neg y)]$$
- TQBF on tosien täysin kvantifioitujen Boolean kaavojen kieli

Lause 8.9 TQBF on PSPACE-täydellinen.

- Mv. $A \in \text{PSPACE}$ pitää palauttaa polynomisesti TQBF:en lähtien liikkeelle A :n polynomisessa tilassa toimivasta tunnistajakoneesta M
- Cook-Levin lauseen tapainen todistus ei kuitenkaan tule kyseeseen, koska kone M voi edelleen vaatia eksponentiaalisen ajan
- Tarvitaan Savitchin lauseen todistustekniikan kaltainen lähestyminen
- Syöte w kuvataan kvantifioiduksi kaavaksi φ , joka on tosi jos ja vain jos M hyväksyy w :n
- Muuttujajoukot c_1 ja c_2 kuvaavat tilanteita



- Muodostetaan kaava $\varphi(c_1, c_2, t)$, joka on tosi jos ja vain jos (joss) M voi siirtyä c_1 :stä c_2 :een korkeintaan $t > 0$ askelella
- Palautus tapahtuu kun muodostetaan kaava $\varphi(c_{\text{start}}, c_{\text{accept}}, h)$, missä $h = 2^{df(n)}$
- d on siten valittu vakio, että M :llä ei ole n :n mittaisella syötteellä mahdollisia tilanteita kuin korkeintaan $2^{df(n)}$
- $f(n) = n^k$
- Kaavan tulee koodata koneen M työnauhan sisältö
- Jokaiseen nauhan soluun liittyy useita muuttujia, yksi jokaista nauha-aakkoston merkkiä kohden ja yksi jokaista M :n tilaa vastaten



- Koska tilanteessa on n^k solua, niin on muuttujiakin $O(n^k)$
- Kun $t = 1$, niin
 - Joko $c_1 = c_2$ tai
 - c_2 seuraa c_1 :stä yhdessä M :n siirtymäaskelella
- Kaavan $\varphi(c_1, c_2, t)$ muodostaminen on helppoa
 - Kunkin c_1 :tä kuvaavan muuttujan arvo on sama kuin vastaavan c_2 :ta kuvaavan muuttujan arvo
 - (Kuten Cook-Levin lauseen todistuksessa) kirjoittamalla lauseet, jotka varmistavat, että c_1 :n kuvauksen kolmikot voivat johtaa suoraan vastaaviin c_2 :n kuvauksen kolmikiin



- Kun $t > 1$, niin $\varphi(c_1, c_2, t)$ muodostetaan rekursiivisesti
- Suoraviivainen ratkaisu olisi määritellä

$$\varphi(c_1, c_2, t) = \exists m_1 [\varphi(c_1, m_1, t/2) \wedge \varphi(m_1, c_2, t/2)],$$
 missä m_1 on M :n tilanne
- Tarkkaan ottaen $\exists m_1$ tarkoittaa $\exists x_1, \dots, x_l$, missä $l = O(n^k)$ ja x_1, \dots, x_l ovat tilanteen m_1 koodaavat muuttujat
- $\varphi(c_1, c_2, t)$ on oikea kaava, sen arvo on tosi kun M voi päästä c_1 :stä c_2 :een t :ssä askeleessa
- Sen pituus kuitenkin kasvaa rekursion myötä liian pitkäksi (eksponentiaaliseksi)



- Kaavan $\varphi(c_1, c_2, t)$ pituuden lyhentämiseksi muutetaan se muotoon

$$\exists m_1 \forall (c_3, c_4) \in \{ (c_1, m_1), (m_1, c_2) \}: [\varphi(c_3, c_4, t/2)]$$
- Selvästikin tämä kaava ilmaisee saman ehdon kuin aiemminkin, mutta nyt yhdellä rekursiivisella kaavalla
- Kaavan syntaktisesti korrektiin muotoon muuttamiseksi $\forall x \in \{y, z\}: [\dots]$ voidaan korvata ekvivalentilla $\forall x [(x = y \vee x = z) \Rightarrow \dots]$
- Kukin rekursiotaso kasvattaa kaavan pituutta $O(f(n))$:n verran
- Rekursiotasoja on $\log(2^{d(f(n))}) = O(f(n))$, joten kaikkiaan muodostuvan kaavan pituus on $O(f^2(n))$



- Kirjassa esitetään kahden "keinotekoisien" pelin PSPACE-täydellisyys
- Niille ei siis ole polynomi aikaista algoritmia, jollei $P = PSPACE$
- 8×8 -laudalla pelattava shakki ei kuitenkaan suoraan taivu samanlaiseen todistustekniikkaan
- Tilanteita on "vain" äärellinen määrä
- Periaatteessa voitaisiin siis tabuloida kaikki mahdolliset tilanteet ja paras siirto kussakin tilanteessa
- Täten Turingin koneen (tai äärellisen automaatin) kontrolliin voi tallettaa vastaavan tiedon ja koneella voi pelata optimaalisesti lineaarisessa ajassa



- Taulukon koko vain valitettavasti on linnunrataamme suurempi
- Kiinteän kokoisten ongelmien vaativuutta ei voi käsitellä esitetyillä tekniikoilla
- Ne pätevät vain vaativuuden kasvunopeudelle ongelman tapauksen pituuden kasvaessa
- Yleistämällä pelit $n \times n$ -laudoille, voidaan niiden optimaalisen pelaamisen vaativuutta tarkastella asympotoottisin keinoin
- Mm. shakki ja GO on osoitettu ainakin PSPACE-koviksi tässä katsantokannassa