Lauselogiikan soveltaminen

- Toteutuvuusongelmaan voidaan palauttaa useat muut logiikkaan liittyvät kysymyksenasettelut.

  Esimerkki. Lause on pätevä joss sen negaatio ei ole toteutuva.

  Esimerkki. Lause ϕ seuraa loogisesti lausejoukosta Σ joss $\Sigma \cup \{\neg\phi\}$ ei ole toteutuva.

  Esimerkki. Lauseet ϕ and ψ ovat ekvivalentteja joss $\neg(\phi \leftrightarrow \psi)$ ei ole toteutuva.

- Mitkä ongelmat voidaan palauttaa toteutuvuusongelmaan tehokkaasti?

  ☞ Kaikki **NP**-luokkaan kuuluvat ongelmat!

NP-luokka

- **NP**-luokka on päätösongelmien kokoelma
- Päätösongelma muodostuu äärettömästä joukosta tapauksia ja kyllä/ei-kysymyksestä.

  Esimerkki. 3-väritysongelma
  Tapaukset: graafit
  Kysymys: voidaan annetun graafin solmut värittää 3 värillä siten, että naapureilla ei ole sama väri?

- Päätösongelma L kuuluu **NP**-luokkaan joss on olemassa polynomisesti tasapainotettu ja polynomisesti ratkeava relaatio R siten, että ongelma "kyllä"-tapausten joukko $L = \{x \mid (x,y) \in R \text{ jollekin } y\}$.

  (R on polynomisesti ratkeava, jos on olemassa polynomisen ajan algoritmi, joka päättää päteekö $(x,y) \in R$ vai ei;
  R on pol.tasapainotettu, jos kaikille $(x,y) \in R$, $|y| \leq |x|^k$ jollakin $k \geq 1$.)

NP-luokka (II)

- **NP**-luokan päätösongelmilla on tiiviit todisteet: jokaisella "kyllä"-tapauksella x on tiivis (polynomisen kokoinen) todiste y, joka voidaan varmentaa polynomisessa ajassa.
- **Esimerkki.** Lauselogiikan toteutuvuusongelma (SAT) kuuluu **NP**-luokkaan.

  Tapaukset: Lauselogiikan lauseet
  Kysymys: Onko annettu lause toteutuva?

  Todiste: Totuusjakelu

  $R = \{(x,y) \mid x \text{ on lause ja y totuusjakelu s.e. } y \models x\}$

  "Kyllä"-tapausten joukko:

  SAT $= \{x \mid (x,y) \in R \text{ jollekin } y\}$.

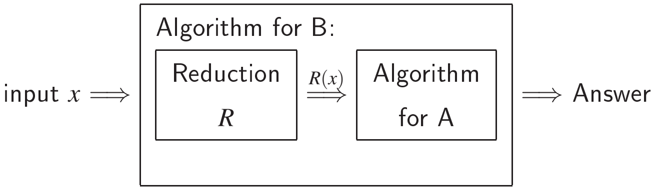
NP-luokka (III)

Useimpien käytännön tietojenkäsittelytehtävien päätösversiot kuuluvat **NP**-luokkaan:

- Tehtävänä on löytää rakenne (todiste), joka täyttää tietyt vaatimukset (polku, reitti, suunnitelma, osien sijoittelu, yhtälön ratkaisu, ...).
- Rakenne (todiste) ei ole valtava ongelman syötteeseen nähden (polynomisesti tasapainotettu).
- Annetut ehdot voidaan tarkastaa polynomisessa ajassa (polynomisesti ratkeava).

Ongelmien vaativuusjärjestys

- **NP**-luokka on ääretön kokoelma päätösongelmia, joiden laskennallinen vaativuus tuntuu vaihtelevan.
- Ongelmat voidaan järjestää vaativuusjärjestykseen ($B \leq A$: A on vähintään yhtä vaikea kuin B) reduktio-kuvausten avulla.
- B palautuu ongelmaan A ($B \leq A$) jos on olemassa (polynomisessa ajassa laskettava) kuvaus R s.e. jokaiselle B:n syötteelle x, $R(x)$ on A:n syöte ja x:n vastaus ongelmassa B on "kyllä" joss $R(x)$:n vastaus ongelmassa A on "kyllä".



Ongelmien vaativuusjärjestys (II)

- **NP**-luokan ongelma A on **NP**-täydellinen, jos jokainen **NP**-luokan ongelma B palautuu siihen ($B \leq A$).
- Lauselogiikan toteutuvuusongelma (SAT) on **NP**-täydellinen eli kaikki **NP**-luokan ongelmat voidaan palauttaa polynomisessa ajassa SAT-ongelmaan.

Esimerkki. 3-väritysongelman palautus SAT-ongelmaan.
Graafin $G = \langle S, K \rangle$ jokaista solmua $a \in S$ kohti 3 atomilausetta: A_p, A_s, A_v
Reduktion $R(G)$ on seuraava lausejoukko:

1. Jokaista solmua $a \in S$ kohti lause: $A_p \vee A_s \vee A_v$.
2. Jokaista kaarta $\langle a, b \rangle \in K$ kohti lauseet: $\neg A_p \vee \neg B_p$, $\neg A_s \vee \neg B_s$ ja$\neg A_v \vee \neg B_v$.

☞ Graafilla on 3-väritys jos ja vain jos lausejoukko on toteutuva.

Ongelmien vaativuusjärjestys (III)

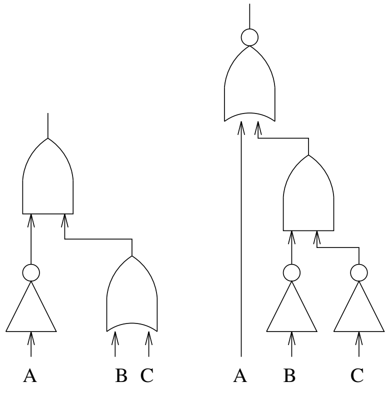
- Osalle **NP**-luokan ongelmista löytyy polynomisen ajan ratkaisumenetelmä .
  Esimerkki. Löytyykö graafissa polku kahden annetun pisteen välille
  2-SAT: $(l_1^1 \vee l_2^1) \wedge \cdots \wedge (l_1^m \vee l_2^m)$
  HORNSAT: $(a \vee \neg b_1^1 \vee \cdots \vee \neg b_k^1) \wedge \cdots \wedge (\neg b_1^m \vee \cdots \vee \neg b_l^m)$
- Osalle (erityisesti **NP**-täydelliset ongelmat) polynomista algoritmia ei tunneta.
  Esimerkki. 3-SAT: $(l_1^1 \vee l_2^1 \vee l_3^1) \wedge \cdots \wedge (l_1^m \vee l_2^m \vee l_3^m)$ on jo **NP**-täydellinen.
- Huom! Jos yhdelle **NP**-täydelliselle ongelmalle löydettäisiin pol.ajan ratkaisumenetelmä, kaikki **NP**-luokan ongelmat voitaisiin ratkaista pol.ajassa.
- Menetelmät ratkaista **NP**-täydellisiä ongelmia:
  likimääräiset algoritmit, erikoistapaukset, heuristiikat, paikallinen haku, . . .

Eräs visio

- Uusi ongelmanratkaisuparadigma laskennallisesti vaikeille ongelmille: kuvataan ongelman ratkaisut korkean tason määrittelykielellä, joka käännetään lauselogiikkaan siten, että saadun lauseen mallit antavat ratkaisut alkuperäiseen ongelmaan.
- Toteutetaan tehokas toteutuvuustarkastin, jossa paljon erikoisosaamista vaikeiden ongelmien ratkaisemisesta.
- Tämä lähestymistapa jo käytössä muutamilla alueilla.

Digitaalipiirien verifiointi

- ➤ Piirin määrittely ja toteutus esitetään logiikan lauseina.
- ➤ Toteutuksen oikeellisuus verifioidaan tarkastamalla loogisen ekvivalenssi määrittelyn kanssa.



$\neg A \wedge (B \vee C)$ vs. $\neg(A \vee (\neg B \wedge \neg C))$

ATPG

- ➤ Automaattisessa testien generoinnissa (automatic test pattern generation) haetaan testisarjoja, jolla voidaan tunnistaa annetun vikamallin vikoja.
- ➤ Monet laitteistotestauksen kysymykset palautuvat SAT-ongelmaan.
- ➤ **Esimerkki.** "Jumitus"-vikamalli: tietty portti on jumittunut tiettyyn arvoon ja haluttaisiin testi, jolla tämä todetaan.

  Jumitus-arvoon-1: löydä syöte, jolla portin pitäisi saada arvo "0".

  ☞ Toteutuvuusongelma:

  Olkoon ϕ lauselogiikan lause, joka ilmaisee portin laskeman Boolen funktion.

  Jumitus-arvoon-1 -testi: totuusjakelu, jossa $\neg\phi$ on tosi.

Symbolinen mallintarkastus

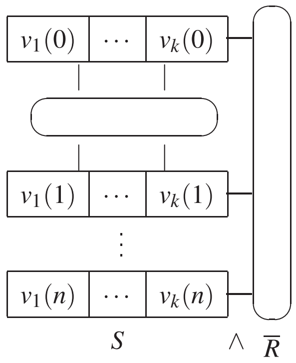
- ➤ Ajatuksena on esittää tutkittavan järjestelmän (mallin) koko tila-avaruus symbolisesti: saavutettavien tilojen joukkoa esitetään lauselogiikan lauseena $R(\vec{x})$, missä $\vec{x}$ on järjestelmän tilabittejä vastaava atomilausejoukko. (Lauseiden esitys usein BDD-muodossa).
- ➤ Jokainen lauseen $R(\vec{x})$ toteuttava totuusjakelu vastaa saavutettavaa tilaa.
- ➤ Olkoon $T(\vec{x},\vec{x}')$ järjestelmän tilasiirtymärelaation esitys ja $I(\vec{x})$ mahd. alkutilojen esitys lauseina. Tällöin $R(\vec{x})$ voidaan rakentaa seuraavasti:

  $R(\vec{x}) := I(\vec{x})$; (Huom. $\exists xp \equiv p(\top/x) \vee p(\bot/x)$)
  repeat
  $R'(\vec{x}) := R(\vec{x})$;
  $R(\vec{x}) := \exists\vec{x}(T(\vec{x},\vec{x}') \wedge R(\vec{x}))[\vec{x}/\vec{x}']$;
  until $R(\vec{x}) \equiv R'(\vec{x})$
- ➤ Esim. ei-toivottavan tilan löytäminen voidaan palauttaa ongelmaan, onko lause $R(\vec{x}) \wedge bad(\vec{x})$ toteutuva, missä $bad(\vec{x})$ on huonojen tilojen esitys.

Rajoitettu mallintarkastus

(Bounded Model Checking)

- ➤ Järjestelmän käyttäytyminen avataan rajatun askelmäärän n päähän (lause S).
- ➤ Tarkastettavan vaatimuksen R negaatio (lause $\overline{R}$) .
- ➤ $S \wedge \overline{R}$ on toteutuva joss järjestelmällä on korkeintaan n-pituinen suoritus, joka rikkoo vaatimusta R.
- ➤ Lyhyiden virheellisten suoritusten hakumenetelmä.



Muita sovellutuksia

- Suunnitteluongelmat

  Samalla periaatteella kuin rajoitettu mallintarkastus.

  Haetaan lyhyitä suunnitelmia ensin: kasvatetaan suunnitelman sallittua pituutta n vähitellen.

- Rajoiteohjelmointiongelmat
- Kombinatoriset ongelmat

Vaikeat toteutuvuustarkastustapaukset

- SAT-ongelman **NP**-täydellisyys: ei tunneta algoritmia, jonka ajoaika *huonoimmassakin tapauksessa* kasvaa vain polynomisesti syötteen kokoon nähden.
- Monille lauseille toteutuvuustarkastus on laskennallisesti varsin kevyttä.
- Näin on esim. useille satunnaisesti generoiduille lauseperheille.
- Vaikeat tapaukset riippuvat yleensä käytetystä algoritmista.
- Monille täydellisille menetelmille voidaan antaa (eksponentiaalinen) *suoritusajan alaraja* tietyntyyppisille *toteutumattomille* lauseille: menetelmän mukaisen todistuspuun koolle löytyy alaraja.
- Vaikeiden toteutuvien lauseperheiden generointi on haastavampaa.

Vaikeita toteutuvuustarkastustapauksia

- Satunnaisesti generoidut ongelmat

  Kiinteänmittaiset lauseet faasimuutospisteessä

- Tapaukset, jotka saadaan palauttamalla jonkin toisen sovellusalueen vaikea ongelmatapaus toteutuvuusongelmaksi.

  Esim. Vaikea graafin väritysongelma toteutuvuusongelmaksi.

- Vaikeita toteutuvia tapauksia

  Esim. yksisuuntaisen funktion kääntäminen: esim. tekijöihinjako

- Teollisuusesimerkit

  Syötteen koko iso mutta toisaalta syötteellä paljon rakennetta.

Satunnaiset k-SAT-ongelmat

Laskennallisesti hankalia SAT-ongelmia saadaan generoimalla satunnaisesti samanmittaisia (k literaalia sisältäviä) klausuuleja.

- Parametrit: atomisten lauseiden määrä v,

  literaalien määrä k jokaisessa klausuulissa

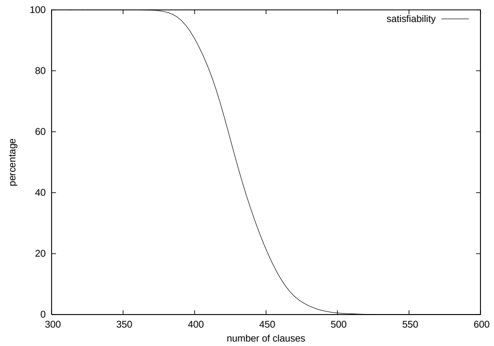
  klausuulien määrä c.

- Proseduuri: toistetaan c kertaa seuraava:
  1. Valitaan satunnaisesti k kpl atomisista lauseista $A_1, \ldots, A_v$.
  2. Muodostetaan näistä k:n literaalin klausuuli siten, että jokainen valituista muuttuja A_i kirjataan
     - positiiviseksi literaaliksi A_i todennäköisyydellä 0.5 ja
     - muutoin negatiiviseksi literaaliksi $\neg A_i$.

Faasimuutosilmiö

- Kaikki tällaiset satunnaiset k-SAT -ongelmat eivät ole vaikeita toteutuvuustarkastaa mutta hankalia klausuulijoukkoja löytyy *faasimuutospisteessä*.
- Satunnaisilla k-SAT -ongelmilla on seuraavaa faasimuutosominaisuus: jos tietyllä atomilauseiden määrällä v generoidaan satunnaisia k-SAT -tapauksia, joissa klausuulien määrä c kasvaa, tapahtuu tietyllä suhteella $\frac{c}{v}$ äkillinen muutos, jossa todennäköisyys sille, että generoitu klausuuli on toteutuva muuttuu nopeasti arvosta 1 arvoon 0.
- Faasimuutospisteeksi kutsutaan sitä suhteen $\frac{c}{v}$ arvoa, jolla tämä todennäköisyys on 0.5.

Faasimuutosilmiö (II)



- Esimerkiksi satunnaisilla 3-SAT-ongelmilla faasimuutospiste saavutetaan suhteen $\frac{c}{v} \approx 4.3$ ($\frac{c}{v} = 4.258 + 58.26v^{-5/3}$ [Crawford and Auton, 1996]).
- Tällaiset ongelmat ovat usein vaikeita täydellisille menetelmille mutta helpompia paikalliseen hakuun perustuville epätäydellisille menetelmille.

Yksisuuntaisen funktion kääntäminen

- Vaikea toteutuva ongelmaluokka saadaan yksisuuntaisten funktioiden käänteisfunktiosta.
- Ideana on kuvata yksisuuntaisen funktion laskenta Boolen piirinä. Tällöin käänteisfunktio voidaan nähdä toteutuvuusongelmana, jossa pitää löytää totuusjakelu atomeille (syöte) niin, että piiri laskee halutun tuloksen.
- Esimerkkejä DES, AES, tekijöihinjako

Algoritmien vertailu

Algoritmien kokeellinen vertailu ei ole ongelmatonta, koska tyypillisesti koejärjestelyissä löytyy vaikeasti hallittavissa olevia eroja, esim.:

- laitteisto
- ohjelmointitaito
- “virittäminen”
- käytetty työmäärä
- testitapausten valinta

  ...

Kokeet

Tavoitteena on testata ko. järjestelmää ainakin seuraavilla testitapauksilla:

- Satunnaiset 3-SAT -ongelmat:

  Käytetään testitapausten generointiin MakeWFF-ohjelmaa. Tavoitteena selvittää, kuinka suuret tapaukset (montako atomia) saadaan ratkaistua "luotettavasti kohtuullisessa ajassa" faasimuutospisteessä.

- Tekijöihinjako-ongelma

  Käytetään testitapausten generointiin genfacbm-ohjelmaa Tavoitteena selvittää, kuinka iso luku (montako bittiä) saadaan jaettua tekijöihin/osoitettua alkuluvuksi "luotettavasti kohtuullisessa ajassa".

- Käytännön esimerkit

  Mitkä valmiista mallintarkastustapauksista tutkittava järjestelmä pystyy ratkaisemaan "luotettavasti kohtuullisessa ajassa"?

Käytännön ohjeita

- **Muistettavaa**

  Toistettavuus, mittaustarkkuus, mittausjärjestelyt, dokumentointi

  Algoritmit usein heuristisia ja satunnaistettuja: ajoajat voivat vaihdella rajusti

- Lisää ohjeita kotisivulla http://www.tcs.hut.fi/Studies/T-79.194/

- Vastaanotot: to 13.2. klo 11.30-12.30; to 20.2. klo 12-13, to 27.2. klo 12-13.

- Yhteenvetotilaisuus to 6.3. klo 12-14: esitellään lyhyesti koetulokset (ei vaikuta arvosteluun).

- Koeajot: kass.tcs.hut.fi

  Ylläpito: Matti Järvisalo (mjj@tcs.hut.fi)