Todistuskompleksisuudesta Boolen piirien toteutuvuustarkastuksessa

Matti Järvisalo
Teknillinen korkeakoulu
Tietojenkäsittelyteorian laboratorio
matti.jarvisalo@hut.fi

1 Johdanto

Lauselogiikan toteutuvuusongelmassa [17] kysytään, onko annetulle lauselogiikan lauseelle olemassa sen toteuttavaa totuusjakelua. Toteutuvuusongelma on tyypillinen **NP**-*täydellinen* ongelma [5], jolle ei tunneta yleistä polynomiaikaista ratkaisualgoritmia. Muun muassa suunnittelu-, mallintarkastus-, testaus- ja verifiointiongelmia [12, 4, 13, 3] voidaan ratkaista toteutuvuusongelmatapauksina. Tehokkaille toteutuvuustarkastimille on näin ollen suuri kysyntä. Toteutuvuustarkastimia onkin kehitetty ja käytetty onnistuneesti monien erilaisten ongelmien ratkaisemiseen.[1] Sekä täydelliset, systemaattiset että satunnaistetut paikalliseen hakuun perustuvat toteutuvuustarkastusmenetelmät (katso esimerkiksi [18, 15, 14, 1]) ovatkin kehittyneet huomattavasti viimeisten noin 10 vuoden aikana. Yleisesti tehokkaita menetelmiä ei silti ole kyetty kehittämään.

Tehokkaat täydelliset toteutuvuustarkastimet perustuvat tyypillisesti *Davis-Putnam-Logemann-Loveland-menetelmään* (DPLL) [8], joka edellyttää syötteen olevan *konjunktiivisessa normaalimuodossa* (KNM) [16], eli ongelmatapauksen tulee koostua joukosta disjunktioita. KNM:n käyttö ongelman mallinnuskielenä on hyvin vaivalloista. Tämän vuoksi tyypillisesti käytetään ensin yleisempää esitysmuotoa, joka tämän jälkeen käännetään KNM:oon. Polynomiaikainen käännös KNM:oon edellyttää apumuuttujien käyttöönottoa. Ongelmana on, että toteutuvuustarkastimien laskenta-aika kasvaa pahimmassa tapauksesa eksponentiaalisesti muuttujien määrän suhteen. Lisäksi käännös saattaa hävittää instanssin sisäisen rakenteen, jota muutoin olisi ollut mahdollista käyttää hyväksi hakuprosessissa.

Tässä työssä esitellään yleisemmälle, *Boolen piireiksi* [17] kutsutulle esitysmuodolle kehitetty taulumenetelmä. Menetelmää voidaan pitää DPLL-menetelmän yleistyksenä Boolen piireille. Boolen piirit ovat hyvin luonnollinen ja ongelman sisäisen rakenteen säilyttävä esitysmuoto monille käytännön ongelmille. Työ käsittelee laskennalliselta kannalta oleellisen *haarauttavan säännön*[2] käyttöä. Haarauttavan säännön käyttö johtaa pahimmassa tapauksessa eksponentiaaliseen määrään turhaa työtä toteutuvuushaun aikana. Työssä tarkastellaan, miten haarauttavan säännön rajoittaminen vaikuttaa toteutuvuustarkastuksen tehokkuuteen. Erityisesti tarkastellaan, kuinka lyhyitä todistuksia on mahdollista tuottaa haarauttavan säännön rajoitukset huomioonottaen.

Työssä osoitetaan, että haarauttavan säännön käytön rajoittaminen millä tahansa tarkastelluista tavoista kasvattaa pienimmän mahdollisen todistuksen kokoa pahimmassa tapauksessa eksponentiaalisesti. Tulokset pätevät DPLL-menetelmään yleisesti käytetyllä käännöksellä Boolen piireistä tuotettujen konjuktiivisessa normaalimuodossa olevien lauseiden osalta.

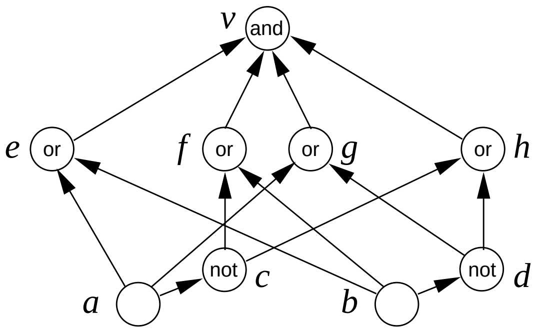
2 Boolen piireistä

Boolen piirit ovat suunnattuja, asyklisiä verkkoja, joiden solmuja kutsutaan *porteiksi*. Portit voidaan jakaa kolmeen joukkoon: *tulosportteihin*, *väliportteihin* ja *syöteportteihin*. Jokaiseen tulos- ja väliporttiin liittyy Boolen funktio. Esimerkki Boolen piiristä on kuvassa 1. Kuvan piirissä v on tulosportti, c, d, e, f, g, h väliportteja ja a, b syöteportteja.

Piirin kaaret kuvaavat porttien funktionaalista riippuvuutta toisistaan. Esimerkiksi kuvan 1 piirissä porttien b, c, f välinen riippuvuus on muotoa

[1] Katso [10, 21] katsauksina erilaisiin toteutuvuustarkastustekniikoihin.
[2] Englanniksi *splitting rule* tai *cut rule*.

$f = \mathsf{or}(b, c)$, ja edelleen porttien a, c välillä on riippuvuus $c = \mathsf{not}(a)$.



Kuva 1: Esimerkki Boolen piiristä.

Boolen piirien semantiikka on ilmeinen: kiinnitettäessä totuusarvot syöteportteihin määrittyvät muiden porttien totuusarvot yksikäsitteisesti. *Rajoitetussa Boolen piirissä* sallitaan rajoitteet piirin porttien totuusarvoiksi. *Boolen piirien toteutuvuusongelmassa* kysytään, onko annetun rajoitetun Boolen piirin syöteporteille olemassa sellaista totuusarvojakelua, joka ei aiheuta ristiriitaa piirin rajoitteiden kanssa. Jos tällainen totuusjakelu on olemassa, sanotaan kyseistä piiriä *toteutuvaksi*, ja muulloin *toteutumattomaksi*. Kuten lauselogiikan tapauksessa, myös Boolen piirien toteutuvuusongelma on **NP**-täydellinen. Esimerkiksi kuvan 1 piirille ei ole sen toteuttavaa totuusjakelua rajoitteella "portti v on *tosi*". Toisaalta rajoitteilla "portti e on *epätosi*" ja "portti g on *tosi*" totuusjakelu $a \to$ *epätosi*, $b \to$ *epätosi* toteuttaa piirin.

3 Taulumenetelmä

Eräs systemaattinen tapa ratkaista Boolen piirien toteutuvuusongelman instansseja on *taulumenetelmän* käyttäminen [7]. Taulumenetelmässä rakennetaan *tauluksi* kutsuttava binääripuu. Taulun juureen kuvataan tarkasteltavan Boolen piirin funktionaaliset riippuvuudet ja piirissä olevat rajoitteet. Jokaista *todeksi* (*epätodeksi*) rajoitettua porttia v kohden lisätään merkintä $\mathbf{T}v$ ($\mathbf{E}v$). Esimerkkinä kuvan 1 piiri rajoitteella "portti v on *tosi*" on kuvattuna kuvan 2 taulun merkinnöiksi 1–8.

Taulumenetelmä koostuu *säännöistä*. Taulun juuren luomisen jälkeen taulua rakennetaan käyttäen taulumenetelmän sääntöjä. Taulun jokainen haara edustaa totuusjakelua, joka mahdollisesti toteuttaa tarkasteltavan piirin. Jos taulun haarassa on merkinnät $\mathbf{T}v$ ja $\mathbf{E}v$ jollekin portille v, sanotaan haaraa *ristiriitaiseksi*. Jos taulun kaikki haarat ovat ristiriitaiset, sanotaan taulua *hylkäykseksi* (refutaatioksi). Jos rajoitetulle Boolen piirille on mahdollista rakentaa hylkäys, on piiri *toteutumaton*. Hylkäys on siis todistus piirin toteutumattomuudesta.

Tässä työssä tarkasteltava **BC**-*menetelmä* koostuu luonnollisista *päättelysäännöistä* koskien $\mathsf{not}-$, or- ja and-portteja, sekä *taulun haarauttavasta säännöstä*.

Päättelysäännöt ovat seuraavat. Jos $v = \mathsf{not}(v')$ ja haarassa on $\mathbf{T}v$ ($\mathbf{E}v$), voidaan päätella $\mathbf{E}v'$ ($\mathbf{T}v'$), ja päinvastoin. Jos $v = \mathsf{or}(v_1, \ldots, v_k)$, niin voidaan päätellä seuraavaa.

- Jos haarassa on $\mathbf{E}v$, niin voidaan päätella $\mathbf{E}v_i$ kaikilla $1 \leq i \leq k$, ja päinvastoin.
- Jos haarassa on $\mathbf{T}v_i$ jollakin $1 \leq i \leq k$, niin voidaan päätellä $\mathbf{T}v$.
- Jos haarassa on $\mathbf{T}v$ ja
$$\mathbf{E}v_1, \ldots, \mathbf{E}v_{i-1}, \mathbf{E}v_{i+1}, \ldots, \mathbf{E}v_k,$$
niin voidaan päätellä $\mathbf{T}v_i$.

Jos $v = \mathsf{and}(v_1, \ldots, v_k)$, niin voidaan päätellä seuraavaa.

- Jos haarassa on $\mathbf{T}v$, niin voidaan päätella $\mathbf{T}v_i$ kaikilla $1 \leq i \leq k$, ja päinvastoin.
- Jos haarassa on $\mathbf{E}v_i$ jollakin $1 \leq i \leq k$, niin voidaan päätellä $\mathbf{E}v$.
- Jos haarassa on $\mathbf{E}v$ ja
$$\mathbf{T}v_1, \ldots, \mathbf{T}v_{i-1}, \mathbf{T}v_{i+1}, \ldots, \mathbf{T}v_k,$$
niin voidaan päätellä $\mathbf{E}v_i$.

Haarauttavan säännön ideana on sallia *tapausanalyysi*, jossa tutkitaan, miten päättelyä voidaan jatkaa, kun portille, jolle ei ole vielä merkintää haarassa, valitaan totuusarvo. Käytettäessä haarauttavaa sääntöä piirin porttiin v jakautuu siis käsiteltävä haara kahteen haaraan, joista toiseen merkitään $\mathbf{T}v$ ja toiseen $\mathbf{E}v$.

1.	$v = \mathsf{and}(e, f, g, h)$	
2.	$e = \mathsf{or}(a, b)$	
3.	$f = \mathsf{or}(b, c)$	
4.	$g = \mathsf{or}(a, d)$	
5.	$h = \mathsf{or}(c, d)$	
6.	$c = \mathsf{not}(a)$	
7.	$d = \mathsf{not}(b)$	
8.	$\mathbf{T}v$	
9.	$\mathbf{T}e$	$(1, 8)$
10.	$\mathbf{T}f$	$(1, 8)$
11.	$\mathbf{T}g$	$(1, 8)$
12.	$\mathbf{T}h$	$(1, 8)$

13.	$\mathbf{T}a$	(haar.)	14.	$\mathbf{E}a$	(haar.)
15.	$\mathbf{E}c$	$(6, 13)$	20.	$\mathbf{T}b$	$(2, 9, 14)$
16.	$\mathbf{T}b$	$(3, 10, 15)$	21.	$\mathbf{T}d$	$(4, 11, 14)$
17.	$\mathbf{E}d$	$(7, 16)$	22.	$\mathbf{E}d$	$(7, 21)$
18.	$\mathbf{E}h$	$(5, 15, 17)$	23.	$\times$	$(21, 22)$
19.	$\times$	$(12, 18)$			

Kuva 2: **BC**-hylkäys kuvan 1 Boolen piirille.

Esimerkkinä taulun rakentamisesta on kuvassa 2 hylkäys kuvan 1 Boolen piirille.

4 Ongelmanasettelu

Haarauttavan säännön käytössä ongelmallista on, että pahimmassa tapauksessa rakennettavaan tauluun tulee eksponentiaalinen määrä merkintöjä tarkasteltavan piirin koon suhteen, kun taas päättelysäännöillä luotavissa olevien merkintöjen määrä on lineaarinen. Tästä syystä olisikin mielekästä rajoittaa haarauttavan säännön käyttö johonkin luonnolliseen piirin porttien *pieneen* osajoukkoon. Työn ongelmanasettelu on seuraava.

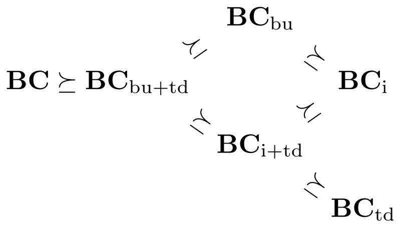
> *Miten rajoitukset haarauttavan säännön käytölle vaikuttavat hylkäysten kokoon?*

Hylkäyksen koko määritellään merkintöjen määräksi hylkäyksessä, missä hylkäyksen juuri lasketaan yhdeksi merkinnäksi. Mittana hylkäyksen koolle käytetään *todistuskompleksisuutta* [2], joka määritellään pienimmän mahdollisen hylkäyksen kooksi tietylle piirille. Todistuskompleksisuuteen pohjautuen voidaan sanoa, että taulumenetelmä T *polynomisesti simuloi* [6] taulumenetelmää T' (merkitään $T \succeq T'$), jos jokaiselle toteutumattomalle Boolen piirille $\mathcal{C}$ pätee, että jos $\mathcal{C}$:lle on mahdollista rakentaa menetelmässä T' hylkäys kokoa n, niin $\mathcal{C}$:lle on mahdollista rakentaa menetelmässä T hylkäys, jonka koko on enintään $p(n)$, missä p on polynomi. Jos $T \succeq T'$ pätee, mutta $T' \succeq T$ ei päde, merkitään $T \succ T'$. Jos sekä $T \succeq T'$ että $T' \succeq T$ eivät päde, merkitään $T \not\equiv T'$.

Tässä työssä tarkastellaan seuraavia **BC**-menetelmän muunnelmia, joissa haarauttavan säännön käyttöä on rajoitettu.

- $\mathbf{BC}_{\mathrm{i}}$: Haarauttavan säännön käyttö on rajoitettu syöteportteihin.
- $\mathbf{BC}_{\mathrm{td}}$: Haarauttavan säännön käyttö on rajoitettu tulosportteihin ja sellaisiin portteihin, joiden jollekin isälle on merkintä haarassa.
- $\mathbf{BC}_{\mathrm{i+td}}$: Haarauttavan säännön käyttö on rajoitettu syöte- ja tulosportteihin sekä sellaisiin portteihin, joiden jollekin isälle on haarassa merkintä.
- $\mathbf{BC}_{\mathrm{bu}}$: Haarauttavan säännön käyttö on rajoitettu syöteportteihin ja sellaisiin portteihin, joiden jollekin lapselle on merkintä haarassa.
- $\mathbf{BC}_{\mathrm{bu+td}}$: Haarauttavan säännön käyttö on rajoitettu syöte- ja tuloportteihin sekä sellaisiin portteihin, joiden jollekin lapselle tai isälle on merkintä haarassa.

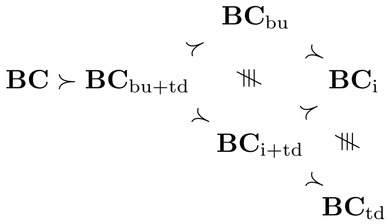
Suoraviivaisesti rajoitustavan sekä relaation $\succeq$ transitiivisuuden perusteella saatava polynomiseen simuloitavuuteen perustuva **BC**-menetelmän muunnelmien välinen järjestys on esitetty kuvassa 3.



Kuva 3: Suoraviivaisesti nähtävä polynomiseen simuloitavuuteen perustuva **BC**-menetelmän muunnelmien välinen järjestys.

5 Tulokset

Tulokset osoittavat, että jokainen tarkasteltu tapa rajoittaa haarauttavan säännön käyttöä johtaa eksponentiaaliseen todistuskompleksisuuden kasvuun. Lisäksi kaikkien haarauttavan säännön suhteen rajoitettujen menetelmien välillä on eksponentiaalisia eroja todistuskompleksisuuden suhteen. Yhteenveto tuloksista on esitetty kuvassa 4.



Kuva 4: Yhteenveto työn tuloksista. Selkeyden vuoksi kuvasta puuttuu merkintä $\mathbf{BC}_{\mathrm{bu}} \not\equiv \mathbf{BC}_{\mathrm{td}}$.

On huomattavaa, että tulokset pätevät DPLL-menetelmään yleisesti käytetyllä *Tseitinin käännöksellä* [20] Boolen piireistä tuotettujen konjuktiivisessa normaalimuodossa olevien lauseiden osalta. Näin ollen työn tulokset osoittavat, että harkitut luonnolliset, paikalliset haarauttavan säännön rajoitukset DPLL-menetelmään perustuvissa toteutuvuustarkastimissa kasvattavat pahimmassa tapauksessa todistuksen kokoa eksponentiaalisesti, toisin kuin usein on oletettu kokeellisiin tuloksiin vedoten [19, 9]. Tulokset perustuvat tiettyihin piirikonstruktioihin, kuten kyyhkyslakkaperiaatteen kuvaukseen Boolen piirinä, sekä **BC**-menetelmän *resoluutiorajoittuneisuuteen.* Todistukset tuloksille löytyvät kirjoittajan diplomityöstä [11].

6 Yhteenveto

Työssä tutkitaan, miten haarauttavan säännön käytön rajoittaminen vaikuttaa todistusten pituuksiin Boolen piirin toteutuvuustarkastuksessa. Tulokset osoittavat, että kaikki tarkastellut rajoitustavat aiheuttavat pahimmassa tapauksessa pienimmän mahdollisen todistuksen kokoon eksponentiaalisesti kasvun. Myös rajoitustapojen välillä osoitetaan olevan eksponentiaalisia eroja. Tulokset osoittavat, että paikalliset haarauttavan säännön rajoitukset DPLL-menetelmään perustuvissa toteutuvuustarkastimissa kasvattavat pahimmassa tapauksessa todistuksen kokoa eksponentiaalisesti.

Viitteet

[1] R.J. Bayardo, R.C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. *Proc. 14th National Conference on Artificial Intelligence*, s. 203–208, 1997.

[2] P. Beame, T. Pitassi. Propositional proof complexity: past, present, and future. *Bulletin of the EATCS*, 65:66–89, 1998.

[3] A. Biere and W. Kunz. SAT and ATPG: Boolean engines for formal hardware verification. *Proc. 20th IEEE/ACM International Conference on Computer Aided Design.* IEEE Press, 2002.

[4] E. Clarke, A. Biere, R. Raimi, Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.

[5] S.A. Cook. The complexity of theorem-proving procedures. *Conference record of 3rd annual ACM Symposium on Theory of Computing*, s. 151–158. ACM Press, 1971.

[6] S.A. Cook, R.A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.

[7] M. D'Agostino, D.M. Gabbay, R. Hähnle, J. Posegga. *Handbook of Tableau Methods.* Kluwer Academic Publishers, 1999.

[8] M. Davis, G. Logemann, D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

[9] E. Giunchiglia, A. Massarotto, and R. Sebastiani. Act, and the rest will follow: Exploiting determinism in planning as satisfiability. *Proc. 15th National Conference on Artificial Intelligence and of the 10th Conference on Innovative Applications of Artificial Intelligence*, s. 948–953. AAAI Press, 1998.

[10] J. Gu, P.W. Purdom, J. Franco, B.W. Wah. Algorithms for the satisfiability (SAT) problem: a survey. *Satisfiability Problem: Theory and Applications, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, 35, s. 19–152. AMS, 1997.

[11] M. Järvisalo. Proof Complexity of Cut-Based Tableaux for Boolean Circuit Satisfiability Checking. Research Report A90, Laboratory for Theoretical Computer Science, Helsinki University of Technology, 2004.

[12] H. Kautz and B. Selman. Planning as satisfiability. *Proc. 10th European Conference on Artificial Intelligence*, s. 359–363, 1992.

[13] T. Larrabee. Test pattern generation using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, 11(1):6–22, 1992.

[14] C.M. Li, Anbulagan. Heuristics based on unit propagation for satisfiability problems. *Proc. 15th International Joint Conference on Artificial Intelligence*, 1997.

[15] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, S. Malik. Chaff: Engineering an efficient SAT solver. *Proc. 38th Design Automation Conference*, s. 530–535. ACM, 2001.

[16] A. Nerode and R.A. Shore. *Logic for Applications.* Springer, USA, 1997.

[17] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, USA, 1994.

[18] Bart Selman, Henry Kautz, Bram Cohen. Local search strategies for satisfiability testing. *Second DIMACS implementation challenge: cliques, coloring and satisfiability, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26, s. 521–532. AMS, 1996.

[19] O. Shtrichman. Tuning SAT checkers for Bounded Model Checking. *Computer Aided Verification; 12th International Conference*, LNCS 1855, s. 480–494, Springer, 2002.

[20] G.S. Tseitin. On the complexity of derivation in propositional calculus. *Automation of Reasoning 2: Classical Papers on Computational Logic 1967-1970*, s. 466–483. Springer, 1983.

[21] L. Zhang, S. Malik. The quest for efficient Boolean satisfiability solvers. *CADE-18*, LNCS 2392 , s. 295–313. Springer, 2002.