Symbolinen verifiointi BDD:llä

Antti Perälä

Helsinki 11. lokakuuta 2002

Rinnakkaisjärjestelmien mallinus ja verifiointi - seminaari

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Johdanto **1**

2 Järjestelmän mallinnus **1**

2.1 Kripke struktuuri 1

2.2 Transitiorelaatio R 2

2.2.1 Esimerkki 3

2.3 Ohjelman mallinnus 4

2.4 Rinnakkaisen ohjelman mallinnus 7

2.4.1 Esimerkki 8

3 Symbolinen verifiointi **9**

3.1 Kripke struktuuri BDD:llä 10

3.1.1 Esimerkki 11

3.2 Saavutettavat tilat 11

3.3 Algoritmi 12

4 Yhteenveto **13**

Viitteet **15**

1 Johdanto

Tässä esitelmässä käsitellään symbolista verifiontia käyttäen BDD kaavioita järjestelmän mallin kuvaamiseen. Aluksi esitellään järjestelmän mallinnusta Kripke struktuuriksi, joka on eräänlainen tilasiirtymä kaavio. Lopuksi esitetään menetelmä Kripke struktuurin muuttamisesta BDD kaavioiksi ja näiden käyttöä symbolisessa verifioinnissa.

2 Järjestelmän mallinnus

[CGP99] Järjestelmän mallinnuksessa muodostetaan järjestelmästä formaalimalli. Verifioinnin kannalta mallin tulisi esittää järjestelmän oikeellisuuteen liittyvät asiat, mutta jättää kaikki toiminnan siihen liittymättömät asiat pois. Aluksi järjestelmän toiminta on jaettava tiloiksi. Tila esittää järjestelmän laskennan vaihetta tietyllä ajan hetkellä, jossa järjestelmän tila on järjestelmän muuttujien arvot. Lisäksi tarvitaan tieto miten järjestelmän tila muuttuu järjestelmän toimintojen vaikutuksesta. Tämä muutos voidaan havainnollistaa kahden tilan avulla, ensimmäinen on järjestelmän tila ennen toiminnon tapahtumista ja toinen toiminnon jälkeen. Tätä paria kutsutaan järjestelmän transitioksi *transition*. Järjestelmän suoritus *computation* on ääretön jono tiloja, joissa jokaisesta tilasta on päästy seuraavaan tilaan jollakin transitiolla.

Järjestelmä voidaan mallintaa Kripke struktuuriksi *Kripke structure* kutsutulla tilasiirtymäkaaviolla *state transition graph*. Kripke struktuuri koostuu tiloista, transitioista tilojen välillä ja funktiosta, joka nimeää jokaisen tilan järjestelmän ominaisuuksien perusteella. Polut tässä rakenteessa kuvaavat järjestelmän suorituksia.

2.1 Kripke struktuuri

Olkoon AP joukko atomisia propositioita. Kripke struktuuri M yli AP on nelijoukko $M = (S, S_0, R, L)$ jossa

1. S on äärellinen joukko tiloja
2. $S_0 \subseteq S$ joukko alkutiloja
3. $R \subseteq S \times S$ transitiorelaatio, jonka pitää olla totaalinen eli jokaiselle $s \in S$ on tila $s' \in S$ niin, että $R(s, s')$
4. $L : S \to 2^{AP}$ funktio , joka nimeää jokaisen tilan joukolla atomisia propositioita, jotka ovat tosia tilassa

Polku rakenteessa M tilasta s on ääretön jono tiloja $\pi = s_0 s_1 s_2 ...$ niin, että $s_0 = s$ ja $R(s_i, s_{i+1})$ on voimassa kaikilla $i \geq 0$.

2.2 Transitiorelaatio R

Järjestelmän kuvaamiseen käytetään ensimmäisen kertaluvun logiikkaa. Näin ollen kaavoissa esiityvillä predikaatti- ja funktiosymboleilla on ennalta määrätty merkitys, joka on yleensä asia yhteydestä selvä. Olkoon $V = \{v_1, ..., v_n\}$ joukko järjestelmän muuttujia. Oletaan että V:hen kuuluvien muuttujien arvoalueet sisältyvät äärelliseen joukkoon D, jota kutsutaan universumiksi. Valuaatio *valuation* on funktio, joka liittää jokaiselle joukon V alkiolle arvon joukosta D. Järjestelmän tila voidaan määritellä antamalla kaikkien V:n alkioiden arvot. Toisin sanoen tila on valuaatio $s : V \to D$ joukolle muuttujia V:ssä. Annetulle valuaatiolle voidaan esittää kaava, joka on tosi kyseisellä valuaatiolla. Esimerkiksi, jos $V = \{v_1, v_2, v_3\}$ ja valuaatio $(v_1 \leftarrow 2, v_2 \leftarrow 3, v_3 \leftarrow 5)$ voidaan muodostaa kaava $(v_1 = 2 \wedge v_2 = 3 \wedge v_3 = 5)$. Yleisesti kaava voi olla tosi useassa eri valuaatiolla. Kun ajatella kaavan esittävän joukkoa valuaatioita, joissa kaikissa kaava on tosi, voidaan ensimmäisen kertaluvun logiikan kaavoilla kuvata tilajoukkoja. Erityisesti alkutilat voidaan kuvata $\mathcal{S}_0$ joukon V muuttujilla.

Tilojen esittämisen lisäksi tarvitaan menetelmä jolla voidaan esittää tilasiirtymiä l. transitioita ensimmäisen asteen logiikan kaavoilla. Tälläisen kaavan tulee esittää järjestettyjä tilapareja. Tämä ei onnistu yhdellä joukolla järjestelmän muuttujia V,

joten luodaan toinen joukko muuttujia V'. Ajatellaan joukon V sisältävän nykyiset muuttujat ja joukon V' sisältävän vastaavasti seuraavan tilan muuttujat. Jokaisella $v \in V$ on vastaava seuraavan tilan muuttuja $v' \in V'$. Valuaatiota joukkojen V ja V' voidaan kuvata järjestetyn tilaparin valitana tai transitiona, ja näitä valuaatiota voidaan esittää kaavoilla. Tilaparijoukkoa kutsutaan transitiorelaatioksi. Olkoon R transitiorelaatio, tällöin $R(V, V')$ tarkoittaa kaavaa, joka määrittelee sen. Järjestelmän ominaisuuksien spesifioinnin kirjoittamiseksi tarvitaan määritelmä joukon AP atomisille propositioille. Ne ovat yleensä muotoa $v = d$, jossa $v \in V$ ja $d \in D$. Propositio on tosi tilassa s, jos $s(v) = d$.

Kripke struktuuri voidaan muodostaa järjestelmää kuvaavista ensimmäisen kertaluvun logiikkan kaavoista $\mathcal{S}_0$ ja $\mathcal{R}$ seuraavalla tavalla:

- Joukon S tilat määräävät kaikki joukon V muuttujien valuaatiot
- Alkutilajoukon S_0 tilat määräävät valuaatiot V:lle, jotka toteuttavat kaavan $\mathcal{S}_0$
- Olkoon s ja s' kaksi tilaa, siten että $R(s, s')$, jos $\mathcal{R}$ on tosi kun kaikille $v \in V$ sijoitetaan $s(v)$ ja kaikille $v' \in V'$ sijoitetaan $s(v')$
- Nimeämisfunktio $L : S \to 2^{AP}$ määritellään niin, että $L(s)$ on osajoukko kaikista atomisista propositioista, jotka ovat tosia s:ssä. Jos v on totuusarvoinen muuttuja, niin $v \in L(s)$ ilmoittaa $s(v)$ olevan tosi ja vastaavasti jos $v \notin V$ silloin $s(v)$ on epätosi

Koska transitiorelaation R haluttiin olevan totaalinen, sitä joudutaan laajentamaan tilanteissa, joissa tilalla s ei ole seuraajia. Näissä tilanteissa muutetaan R niin, että $R(s, s)$ on voimassa.

2.2.1 Esimerkki

Kaksi muuttujaa x ja y, joiden arvoalue $D = \{0, 1\}$. Näin ollen muuttujien valuaatio on yksinkertaisesti pari $(d_1, d_2) \in D \times D$, jossa d_1 on x:n arvo ja d_2 vastaavasti y:n.

Järjestelmä koostuu yhdestä transitiosta

$$x := (x + y) \mod 2$$

ja alkuarvot $x = 1$ ja $y = 1$. Järjestelmää kuvaa kaksi ensimmäisen kertaluvun logiikan kaavaa. Järjestelmän alkutilat esitetään $\mathcal{S}_0(x, y) \equiv x = 1 \wedge y = 1$ ja transitiot $\mathcal{R}(x, y, x', y') \equiv x' = (x + y) \mod 2 \wedge y' = y$. Järjestelmän Kripke struktuuri $M = (S, S_0, R, L)$ muodostetaan näistä kaavoista seuraavasti:

- $S = D \times D$
- $S_0 = \{(1, 1)\}$
- $R = \{((1, 1), (0, 1)), ((0, 1), (1, 1)), ((1, 0), (1, 0)), ((0, 0), (0, 0))\}$
- $L((1, 1)) = \{x = 1, y = 1\}$, $L((0, 1)) = \{x = 0, y = 1\}$, $L((1, 0)) = \{x = 1, y = 0\}$ ja $L((0, 0)) = \{x = 0, y = 0\}$

Ainoa alkutilasta alkava polku on $(1, 1)(0, 1)(1, 1)(0, 1)$.... Tämä polku on myös järjestelmän ainoa suoritus.

2.3 Ohjelman mallinnus

Ohjelma koostuu sarjallisista komennoista. Merkitään $\mathcal{C}$ transitioproseduuria, joka ottaa vastaan sarjallisen ohjelman P ja muuttaa sen ensimmäisen kertaluvun logiikan kaavaksi $\mathcal{R}$, joka esittää ohjelman transitioita. Jokaisella lauseella oletetaan olevan yksikäsitteiset alku- ja loppupisteet. Kun vielä jokainen piste nimetään yksikäsitteisesti, voidaan transitioproseduuria yksinkertaistaa. Nimeämismuunnos muuntaa nimeämättömän ohjelman P nimetyksi ohjelmaksi $P^{\mathcal{L}}$. Sarjallisessa ohjelmassa lauseen loppupiste on identtinen seuraavan komennon alkipisteen kanssa eli riittää nimetä alkupisteet. Kun vielä nimetään ohjelman P alku- ja loppupisteet, saadaan yksikäsitteinen nimeäminen kaikille ohjelman lauseiden pisteille. Annetun lauseen P, nimetty lauseke $P^{\mathcal{L}}$ on:

- Jos P ei ole koottulause (eli P in $x := e$, `skip` jne.), niin $P^{\mathcal{L}} = P$
- Jos $P = P_1; P_2$ niin $P^{\mathcal{L}} = P_1^{\mathcal{L}}; l'' : P_2^{\mathcal{L}}$
- Jos $P =$`if` b `then` P_1 `else` P_2 `end if`, niin $P^{\mathcal{L}} =$ `if` b `then` $l_1 : P_1^{\mathcal{L}}$ `else` $l_2 : P_2^{\mathcal{L}}$ `end if`
- Jos $P =$ `while` b `do` P_1 `end while`, niin $P^{\mathcal{L}} =$ `while` b `do` $l_1 : P_1^{\mathcal{L}}$ `end while`

Jatkossa oletetaan, että P on nimetty lauseke ja P:n alku- ja loppupisteet ovat m ja m'. Ohjelmaan lisätään erikoismuuttuja pc, joka toimii ohjelman käskylaskurina *program counter*. Käskylaskuri saa arvokseen ohjelman pisteiden nimiä. Lisäksi pc:lle tarvitaan ylimääräinen arvo $\perp$, joka kuvaa tuntematonta arvoa. Ilmaus $pc = \perp$ kertoo, ettei ohjelma ei ole aktiivinen eikä sitä näin ollen voida suorittaa.

Olkoon taas V ohjelman muuttujat ennen transition suoritusta ja V' muuttujat transition suorituksen jälkeen, kuten edellä. Nyt pc' esittää käskylaskurin arvoa transition suorituksen jälkeen. Koska jokainen transitio muuttaa vain muutamaa muuttujaa, merkitään `same(`Y`)` kaavaa $\bigwedge_{y \in Y} (y' = y)$. Ohjelman P alkutilat voidaan esittää kaavalla $\mathcal{S}_0(V, pc) \equiv$ `pre(`V`)` $\wedge\, pc = m$, jossa `pre(`V`)` on jokin muuttujien $v \in V$ mahdollisista alkuarvoista.

Muunnosproseduuri *translation prosedure* $\mathcal{C}$ toiminta riippuu kolmesta parametrista: alkupisteestä l, lauseesta P ja loppupisteestä l'. Proseduuri määritellään rekursiivisesti. $\mathcal{C}(l, P, l')$ kuvaa transitiojoukon P disjunktioksi kaikkien transitioiden joukosta. Yksittäisen transition disjunktio määrää käskylaskurin arvon ja totuusarvoisen kaavan arvon, joiden perusteella päätellään mitkä transitiot voidaan suorittaa.

- Sijoitus:
  $\mathcal{C}(l, v \leftarrow e, l') \equiv pc = l \wedge pc' = l' \wedge v' = e \wedge$ `same(`$V \setminus \{v\}$`)`
- Skip:
  $\mathcal{C}(l,$ `skip`$, l') \equiv pc = l \wedge pc' = l' \wedge$ `same(`V`)`
- Peräkkäissuoritus:
  $\mathcal{C}(l, P_1; l'' : P_2, l') \equiv \mathcal{C}(l, P_1, l'') \vee \mathcal{C}(l'', P_2, l')$

Kaava transitioille $P_1; l'' : P_2$ on disjunktio transitioiden P_1 ja P_2 kaavoista. Koska lauseiden P_1 ja P_2 välissä oleva piste on l'' suoritetaan lause P_1 ennen lausetta P_2.

- Ehtolause:
  $\mathcal{C}(l, \texttt{if}\ b\ \texttt{then}\ l_1 : P_1\ \texttt{else}\ l_2 : P_2\ \texttt{end if}, l')$ on disjunktio seuraavista neljästä kaavasta:

  - $pc = l \wedge pc' = l_1 \wedge b \wedge \texttt{same}(V)$
  - $pc = l \wedge pc' = l_2 \wedge \neg b \wedge \texttt{same}(V)$
  - $\mathcal{C}(l_1, P_1, l')$
  - $\mathcal{C}(l_2, P_2, l')$

  Disjunktion ensimmäinen osa vastaa tapausta, jossa b on tosi. Tällöin suoritetaan lause P_1. Toisessa osassa b on puolestaan epätosi ja suoritetaan lause P_2. Molemmat disjunktion osat kuvaavat transition, jossa vain käskylaskurin arvo muuttuu. Kaksi viimeistä osaa kuvaavat lauseiden P_1 ja P_2 suorituksen ja l' on molempien loppupiste.

- Toistolause: $\mathcal{C}(l, \texttt{while}\ b\ \texttt{do}\ l_1 : P_1\ \texttt{end while}, l')$ on disjunktio seuraavista kolmesta kaavasta:

  - $pc = l \wedge pc' = l_1 \wedge b \wedge \texttt{same}(V)$
  - $pc = l \wedge pc' = l' \wedge \neg b \wedge \texttt{same}(V)$
  - $\mathcal{C}(l_1, P_1, l)$

  Disjunktion ensimmäinen osa vastaa tapausta, jossa b on tosi. Tällöin lause P_1 suoritetaan. Toisessa osassa b on epätosi, jolloin silmukan suoritus päättyy. Kolmas osa on lauseen P_1 suoritus, lauseen loppupiste on sama kuin silmukan alkupiste.

2.4 Rinnakkaisen ohjelman mallinnus

Rinnakkainen ohjelma koostuu joukosta prosesseja, joita voidaan suoritetaan rinnakkain. Yksittäinen prosessi on sarjallinen ohjelma, jonka mallinnus esitettiin edellä. Aluksi olkoon V_i joukko muuttujia, joiden arvoja prosessi P_i voi muuttaa. Joukkojen V_i ei tarvitse olla pistevieraita. Kuten edellä joukko V sisältää kaikki ohjelman muuttujat. Prosessin P_i käskylaskuri on pc_i ja joukko PC sisältää kaikki ohjelman käskylaskurit.

Rinnakkainen ohjelma P on muotoa $\texttt{cobegin}\ P_1 \,||\, P_2 \,||\cdots||P_n\texttt{coend}$, jossa $P_1, \ldots, P_n$ ovat prosesseja. Edellä esitettyä nimeämismuunnosta laajennetaan niin, että rinnaikkaisohjelma voidaan esittää samalla tavalla kuin sarjallinen ohjelma. Kun sarjallisessa ohjelmassa lauseen loppupisteen nimi on sama kuin seuraavan lauseen alkupiste, rinnakkaisessa ohjelmassa kaikkilla pisteille on yksikäsitteiset nimet. Kuten edellä m on ohjelman P alkupiste ja m' loppupiste.

- Jos $P = \texttt{cobegin}\ P_1 \,||\, P_2 \,||\cdots||P_n\texttt{coend}$, niin
  $P^{\mathcal{L}} = \texttt{cobegin}\ l_1 : P_1^{\mathcal{L}} l_1' \,||\, l_2 : P_2^{\mathcal{L}} l_2' \,||\cdots||l_n : P_n^{\mathcal{L}} l_n' \texttt{coend}$

Rinnakkaisen ohjelman P alkutiloja kuvaa kaava

$$\mathcal{S}_0(V, PC) \equiv \texttt{pre}(V) \wedge pc = m \wedge \bigwedge_{i=1}^{n} (pc_i = \bot),$$

jossa $pc_i = \bot$ kertoo ettei prosessi P_i ole aktiivinen eli sitä ei voida suorittaa tässä tilassa.

Muunnosproseduuri $\mathcal{C}$ laajennetaan rinnakkaisille ohjelmille seuraavasti: $\mathcal{C}(l, \texttt{cobegin}\ l_1 : P_1 l_1' \,||\, \cdots||l_n : P_n l_n' \texttt{coend}, l')$ muodostuu kolmen kaavan disjuktiosta:

- $pc = l \wedge pc_1' = l_1 \wedge \ldots \wedge pc_n' = l_n \wedge pc' = \bot$
- $pc = \bot \wedge pc_1 = l_1' \wedge \ldots \wedge pc_n = l_n' \wedge pc' = l' \wedge \bigwedge_{i=1}^{n} (pc_i' = \bot)$
- $\bigvee_{i=1}^{n} (\mathcal{C}(l_i, P_i, l_i') \wedge \texttt{same}(V \setminus V_i) \wedge \texttt{same}(PC \setminus \{pc_i\}))$

Ensimmäinen disjunktion osa kuvaa rinnakkaisten prosessien alustamisen. Siirtymä suoritetaan `cobegin`:n alkupisteestä kaikkien yksittäisten prosessien alkupisteisiin. Toinen osa kuvaa ohjelman lopputilaa. Kaikkien prosessien loppupisteistä siirrytään `coend` lauseen loppupisteeseen. Viimeinen disjunktion osa kuvaa rinnakkaisten prosessien vuorottelevaa suoritusta. Prosessin P_i transitiorelaatio yhdistetään kaavaan $\texttt{same}(V \backslash V_i) \wedge \texttt{same}(PC \backslash \{pc_i\})$. Tästä johtuen prosessin P_i transitiot voivat muuttaa muuttujia V_i. Lisäksi vain yhden prosessin transitio voidaan suorittaa kerralle.

2.4.1 Esimerkki

Esimerkkinä on yksinkertainen poissulkemisongelma, jossa kahdella prosessilla on yhteinen muuttuja. Kyseinen rinnakkaisohjelma on $P = m : \texttt{cobegin}\ P_0 \mid\mid P_1\ \texttt{coend}\ m'$, jossa prosessit P_0 ja P_1 ovat:

$P_0 ::$ $l_0 :$ `while` $True$ `do`

$NC_0 :$ `wait`$(turn = 0)$;

$CR_0 : turn := 1$;

`end while;`

$l'_0 :$

$P_1 ::$ $l_1 :$ `while` $True$ `do`

$NC_1 :$ `wait`$(turn = 1)$;

$CR_1 : turn := 0$;

`end while;`

$l'_1 :$

Ohjelman P käskylaskuri pc saa vain kolme arvo: m P:n alkupiste, m' P:n loppupiste ja $\perp$ kun P_0 ja P_1 ovat aktiivisia. Prosessin P_i käskylaskuri pc_i saa arvot: l_i, l'_i, NC_i, CR_i ja $\perp$. Prosessit jakavat muuttujan $turn$, joten $V = V_0 = V_1 = \{turn\}$ ja $PC = \{pc, pc_0, pc_1\}$. Kun käskylaskurin arvo on CR_i prosessi P_i on kriittisellä alueella. Molemmat prosessit eivät saa olla samaan aikaan kriittisellä alueella. Kun käskylaskurin arvo on NC_i prosessi ei ole kriittisellä alueella, tässä tapauksessa prosessi odottaa tilannetta $turn = i$. Jolloin prosessi saa luvan edetä kriittiselle

alueelle.

Ohjelman P alkutilat kuvataan kaavalla $\mathcal{S}_0(V, PV) \equiv pc = m \wedge pc_0 = \bot \wedge pc_1 = \bot$. Eli muuttujan *turn* alkuarvoa ei määrätä, se voi olla nolla tai yksi. Muunnosproseduurilla $\mathcal{C}$ saadaan P:n transitiorelaatio $\mathcal{R}(V, PC, V', PC')$, joka on disjunktio seuraavista neljästä kaavasta:

- $pc = m \ \wedge \ pc'_0 = l_0 \ \wedge \ pc'_1 = l_1 \ \wedge \ pc = \bot$
- $pc_0 = l'_0 \ \wedge \ pc_1 = l'_1 \ \wedge \ pc' = m' \ \wedge \ pc'_0 = \bot \ \wedge \ pc'_1 = \bot$
- $\mathcal{C}(l_0, P_0, l'_0) \ \wedge \ \mathtt{same}(V \setminus V_0) \ \wedge \ \mathtt{same}(PC \setminus \{pc_0\})$, joka on ekvivalentti kaavan $\mathcal{C}(l_0, P_0, l'_0) \ \wedge \ \mathtt{same}(pc, pc_1)$ kanssa
- $\mathcal{C}(l_1, P_1, l'_1) \ \wedge \ \mathtt{same}(V \setminus V_1) \ \wedge \ \mathtt{same}(PC \setminus \{pc_1\})$, joka on ekvivalentti kaavan $\mathcal{C}(l_1, P_1, l'_1) \ \wedge \ \mathtt{same}(pc, pc_0)$ kanssa

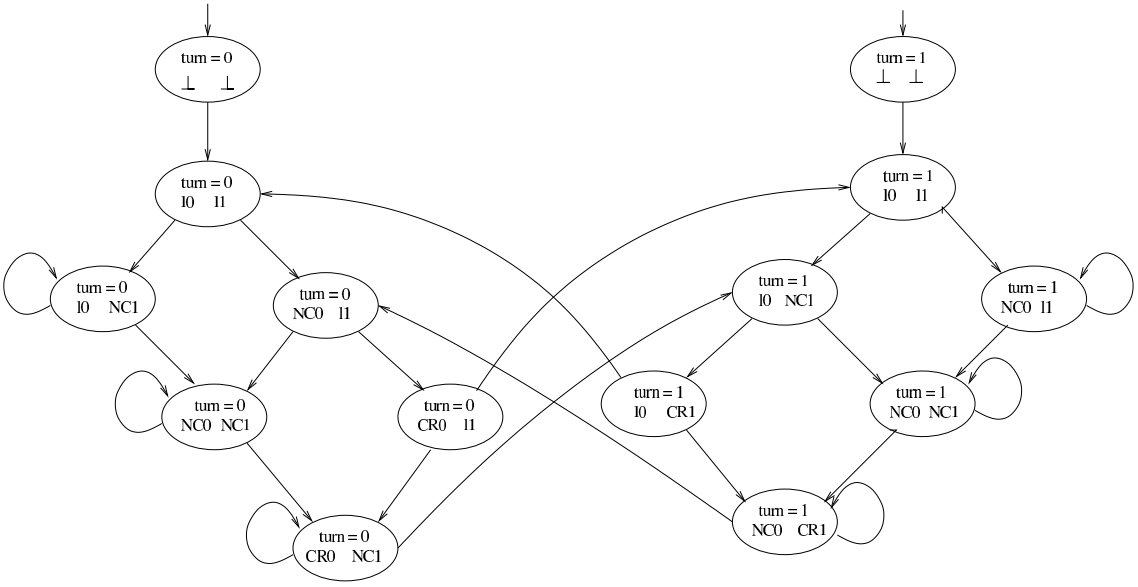
Jokaiselle prosessille P_i, $\mathcal{C}(l_i, P_i, l'_i)$ on disjunktio:

- $pc_i = l_i \ \wedge \ pc'_i = NC_i \ \wedge \ True \ \wedge \ \mathtt{same}(turn)$
- $pc_i = NC_i \ \wedge \ pc'_i = CR_i \ \wedge \ turn = i \ \wedge \ \mathtt{same}(turn)$
- $pc_i = CR_i \ \wedge \ pc'_i = l_i \ \wedge \ turn' = (i+1) \mod 2$
- $pc = NC_i \ \wedge \ pc'_i = NC_i \ \wedge \ turn \neq i \ \wedge \ \mathtt{same}(turn)$
- $pc_i = l_i \ \wedge \ pc'_i = l'_i \ \wedge \ False \ \wedge \ \mathtt{same}(turn)$

Kaavoista $\mathcal{S}_0$ ja $\mathcal{R}$ johdettu Kripke struktuuri on esitetty kuvassa 1. Kripke struktuurin tiloja tutkimalla nähdään etteivät prosessit ole kriittisellä alueella yhtä aikaa.

3 Symbolinen verifiointi

Aluksi esitetään Kripke struktuurin esittäminen BDD:llä. Tämän jälkeen esitellään saavutettavien tilojen laskemisalgoritmi, jossa oletetaan mallin olevan BDD:nä. Lopuksi esitellään symbolinen verifiointi algoritmi, edellisten tietojen perusteella.



Kuva 1: Kripke struktuurin saavutettavat tilat

3.1 Kripke struktuuri BDD:llä

[CGP99] BDD:llä *binary decision diagram* [BRY86] voidaan kuvata äärellisiin joukkoihin määriteltyjä relaatioita. Jos Q on n-arvoinen *n-ary* relaatio joukolle $\{0,1\}$, niin Q voidaan esittää BDD:llä. Tämän BDD:n karakteristinen funktio *characteristic function* $f_Q(x_1,\cdots,x_n) = 1$ joss $Q(x_1,\cdots,x_n)$. Olkoon Q n-arvoinen relaatio äärelliselle joukolle D. Oletetaan, että D:ssä on 2^m alkiota ($m > 1$). Jotta Q voitaisiin esittää BDD:nä, käytetään bijektiota $\phi : \{0,1\}^m \to D$, joka kuvaa m-alkioisen booleanvektorin jokaiselle D:n alkiolle. Käyttäen ϕ:tä muodostetaan $m \times n$-arvoinen booleanrelaatio $\hat{Q}$ käyttäen sääntöä $\hat{Q}(\bar{x}_1,\cdots\bar{x}_n) = Q(\phi(\bar{x}_1),\cdots\phi(\bar{x}_n))$, jossa $\bar{x}_i$ on m:n boolean muuttujan vektori, joka kuvaa muuttujaa x_i. Q voidaan nyt esittää $\hat{Q}$:n karakteristisen funktion $f_{\hat{Q}}$ määräämänä BDD:nä. Tätä menetelmää voidaan soveltaa joukkojen esittämiseen BDD:nä.

Olkoon $M = (S, R, L)$ Kripke struktuuri. Tilajoukko S, oletetaan yksinkertaisuuden vuoksi että joukossa on tasan 2^m alkiota, esitetään BDD:nä käyttämällä jälleen kuvausta $\phi : \{0,1\}^m \to S$. Funktio ϕ kuvaa booleanvektoreita tiloiksi. Koska jokai-

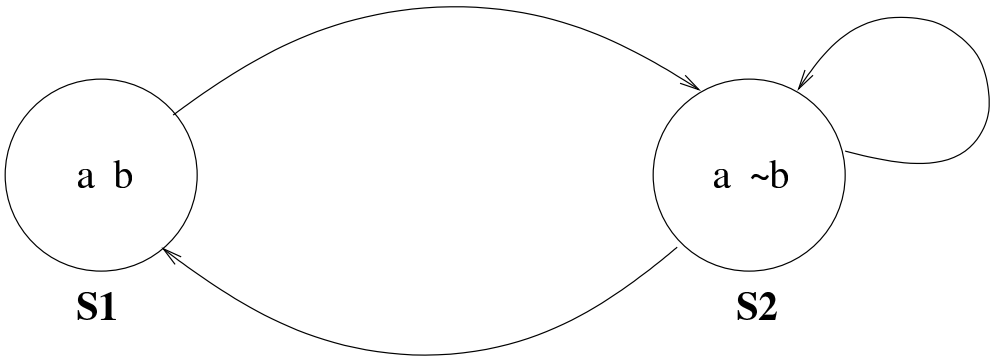
nen booleanvektori on tila joukossa S, S:n esittävä karakteristinen funktio on BDD, jonka arvo on 1. Transitiorelaation R transitiot käsitellään samalla tavalla kuin tilat. Kuten edellä transitio koostuu kahdesta tilasta: alku- ja lopputilasta. Tarvitaan kaksi joukkoa booleanmuuttujia, ensimmäinen alkutilan esittämiseen ja toinen lopputilan esittämiseen. Jos transitiorelaatio R on esitetty booleanrelaationa $\hat{R}(\bar{x}, \bar{x}')$, niin R:ää vastaa karakteristinen funktio $f_{\hat{R}}$. Lopuksi funktio L, joka on määritelty kuvaamaan tiloja joukolle atomisia propositioita. L voidaan myös määritellä kuvaamaan atomisia propositioita joukolle tiloja eli atominen propositio p kuvataan joukolle tiloja, joilla pätee $\{s \mid p \in L(s)\}$. Tämä joukko voidaan nyt muuttaa käyttäen funktiota ϕ, kuten edellä on esitetty. Jokainen atominen propositio muunnetaan erikseen.

3.1.1 Esimerkki

Ajatellaan kuvan 2 kaksi tilaisen Kripke struktuurin esittämistä BDD:nä. Tässä tapauksessa on kaksi muuttujaa a ja b. Merkitään a' ja b' muuttujien arvoja transition lopputilassa. Eli siirtymä tilasta s_1 tilaan s_2 esitetään kaavalla $(a \wedge b \wedge a' \wedge \neg b')$. Transitiorelaatio on kaava $(a \wedge b \wedge a' \wedge \neg b') \vee (a \wedge \neg b \wedge a' \wedge \neg b') \vee (a \wedge \neg b \wedge a' \wedge b')$. Kaavassa on kolme disjunktiota, koska Kripke struktuurissa on kolme transitiota. Tämä kaava voidaan nyt esittää BDD:nä, jolloin sitä voidaan käyttää esitettävässä verifiointi algoritmissa.

3.2 Saavutettavat tilat

[BCL94] Olkoon S_0 joukko tiloja, esitettynä BDD:nä $S_0(V)$. Halutaan laskea BDD $S(V)$, joka esittää tiloja jotka ovat saavutettavissa S_0:sta transitiorelaation R transitioilla. Esityisesti halutaan laskea tilat S_1, jotka saavutetaan S_0 yhdellä askeleella. Tämä joukko voidaan esittää $S_1 = S_0 \cup \{s' \mid \exists s[s \in S_0 \wedge (s, s') \in R]\}$. Transitiorelaatio on myös annettu BDD:nä $R(V, V')$. Nyt voidaan laskea S_1:ä esittävä BDD kaavalla $S_1(V') = S_0(V') \vee \underset{v \in V}{\exists}[S_0(V) \wedge R(V, V')]$, jossa operaatiot $\vee$ ja $\wedge$ on toteutettu BDD-kirjastoissa. Samoin voidaan laskea kahdella transitiolla saavutettavat



Kuva 2: Kaksi tilainen Kripke struktuuri

tila $S_2(V') = S_0(V') \vee \underset{v \in V}{\exists}[S_1(V) \wedge R(V, V')]$. Yleisesti $k+1$ askeleella saavutettavat tilat saadaan kaavalla $S_{k+1}(V') = S_0(V') \vee \underset{v \in V}{\exists}[S_k(V) \wedge R(V, V')]$. Koska tiloja on äärellinen määrä, jossain vaiheessa $S_{k+1} = S_k$ on tosi.

Laskennallisesti vaativin osa kaavaa on tulo *relational product* $\underset{v \in V}{\exists}[S_k(V) \wedge R(V, V')]$. Kuvassa 3 on esitetty algoritmi RelProd tämän laskemiseksi. Algoritmi käy kerran läpi BDD:t $S(V)$ ja $(R(V, V')$, mutta se ei muodosta BDD:tä $S(V) \wedge (R(V, V')$, joka on yleensä hyvin suuri. Algoritmissa ideana on suorittaa normaali konjunktio, paitsi aina kun luotaisiin BDD:hen solmu, joka kuvaisi V:n alkiota, suoritetaankin disjunktio. BDD $S'(V')$ lasketaan kutsulla `RelProd(`$S(V)$`,`$R(V, V')$`,`V`)`.

3.3 Algoritmi

Symbolinen verifiointi algoritmi:

```
Tilajoukot S,S';
S := AlkutilatS_0;
S' := S ∪ reach(S);
while S ∩ S̄' ≠ ∅ do
                    S := S'
                    S' := S' ∪ reach(S')
end while.
```

```
function RelProd(f, g: BDD, E: set of variables): BDD
if f = false ∨ g = false
    return false
else if f = true ∧ g = true
    return true
else if (f, g, E, h) is in the result cache
    return h
else
    let x be the top variable of f
    let y be the top variable of g
    let z be the topmost of x and y
    h0 := RelProd(f|z=0, g|z=0, E)
    h1 := RelProd(f|z=1, g|z=1, E)
    if z ∈ E
        h := Or(h0, h1)
            /* BDD for h0 ∨ h1 */
    else
        h := IfThenElse(z, h1, h0)
            /* BDD for (z ∧ h1) ∨ (¬z ∧ h0) */
    endif
    insert (f, g, E, h) in the result cache
    return h
endif
```

Kuva 3: Relational product algoritmi

Edellä on esitetty funktion `reach` toiminta, se siis laskee tiloista S' yhdellä askeleella saavutettavat tilat. Kaikki muut operaatiot voidaan helposti muuttaa BDD:n operaatioiksi. Algoritmiin voisi vielä lisätä komennon, jolla se oikeasti tutkisi tilajoukon S' tiloja. Esimerkiksi voitaisiin määritellä tilajoukko S_{BAD}, joka sisältäisi virhetiloja, jolloin tutkimalla $S' \cap S_{BAD} \neq \emptyset$ saataisiin selville mahdollisia virhetilanteita.

4 Yhteenveto

Tässä esiteltiin rinnakkaisjärjestelmän mallinnusta BDD:llä ja symbolista verifiointia. Kuitenkin muun muassa CTL ja muut temporaalilogiikat on yksinkertaisuuden

vuoksi jätetty huomiotta.

Viitteet

BRY86 Bryant R.E., Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8), (1986), 677-691.

BCM92 Burch J.R., Clarke E.M., McMillan K.L., Dill D.L., Hwang L.J., Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation*, 98(2), (1992), 142-170.

BCL94 Burch J.R., Clarke E.M., Long D.E., McMillan K.L., Dill D.L., Symbolic Model Checking for Sequential Circuit Verification, *IEEE Transactions on Computers* Aided design of integrated circuits and systems, 13(4), (1994), 401-424.

CGP99 Clarke E.M. Jr., Grumberg O., Peled D.A., Model Checking, *MIT press*, (1999).

BBF01 Bérard B., Bidoit M., Finkel A., Laroussinie F., Petit A., Pertucci L., Schoedelen Ph., McKenzie P., Systems and Software Verification, *Springer-Verlag*, (2001).