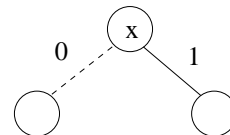


Binääriset päätösdiagrammit

- Binäärinen päätösdiagrammi (binary decision diagram; BDD): Boolean funktion esitys graafina [Lee, 1950; Akers, 1978].
- Ottamalla käyttöön järjestys Boolean funktion muuttujille saadaan *järjestetty binäärinen päätösdiagrammi* (ordered binary decision diagram OBDD) [Bryant, 1986], joka tarjoaa Boolean funktioille kanonisen esitystavan.
- OBDD-tekniikalla pystytään käsittelemään suuria sovellutuksissa esiintyviä Boolean funktioita (digitaaliset piirit, tilakoneet, ...)

Binäärinen päätösdiagrammi

- Esittää Boolean funktioita asyklisina suunnattuna graafina.
- Jokaisella ei-terminaalisolmulla v :
nimetty muuttuja $\text{var}(v)$
 $\text{lo}(v)$: lapsisolmu, kun $\text{var}(v) = 0$
 $\text{hi}(v)$: lapsisolmu, kun $\text{var}(v) = 1$
- Jokainen terminaalisolmu: nimetty joko 0 tai 1
- Jokaisella muuttujien arvojen yhdistelmällä funktion arvo saadaan vastaavasta lehtisolmusta.



Redusoitu järjestetty binäärinen päätösdiagrammi (ROBDD)

Valitaan muuttujille täydellinen järjestys $<$. Jokaisen solmun u ja u :n ei-terminaali lapsen v tulee toteuttaa vaatimus $\text{var}(u) < \text{var}(v)$.

Reduktiosäännöt:

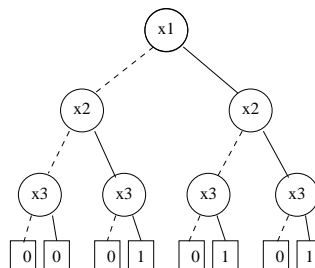
- Poistetaan terminaalien kopiot:
Hävitetään kaikki paitsi yksi kunkin niminen (0/1) terminaali ja siirretään kaaret poistetuista terminaaleista jäljelle jääneisiin.
- Poistetaan ei-terminaalien kopiot:
Jos ei-terminaalisolmuilla v ja u pätee $\text{var}(v) = \text{var}(u)$, $\text{hi}(v) = \text{hi}(u)$ ja $\text{lo}(v) = \text{lo}(u)$, hävitetään toinen solmuista ja siirretään siihen tulevat kaaret toiseen solmuun.
- Poistetaan tarpeettomat testit:
Jos ei-terminaalisolmulla v pätee $\text{hi}(v) = \text{lo}(v)$, hävitetään v ja siirretään siihen tulevat kaaret solmuun $\text{lo}(v)$.

© 2001 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

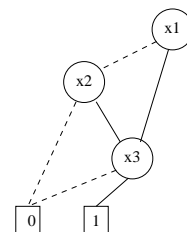
Esimerkki.

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Funktio f



BDD



Redusoitu OBDD

© 2001 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

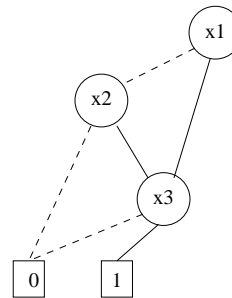
BDD:t ohjelmina

- BDD:t vastaavat yksinkertaisia *haarautuvia ohjelmia* (branching programs):

```
if x then ... /* x=1 -haara */
else ... /* x=0 -haara */
```

- Esimerkiksi edellinen redusoitu OBDD:

```
if x1 then
    return x3
else if x2 then
    return x3
else return 0
endif
```



© 2001 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Kanoninen esitystapa

- ROBDD (maksimaalisesti redusoitu graafi) kanoninen esitystapa Boolean funktiolle:

Kaikki annettua Boolean funktiota f esittävät ROBDD:t ovat *isomorfisia* seuraavalla tavalla.

- BDD:t B_1 ja B_2 ovat isomorfisia, joss on olemassa bijektiivinen funktio f graafin B_1 solmuilta graafin B_2 solmuille siten, että kaikille graafin B_1 solmuille v pätee: jos $f(v) = v'$, joko
 - sekä v että v' ovat terminaalisolmuja, joilla on sama arvo tai
 - sekä v että v' ovat ei-terminaalisolmuja siten, että $\text{var}(v) = \text{var}(v')$, $f(\text{lo}(v)) = \text{lo}(v')$ ja $f(\text{hi}(v)) = \text{hi}(v')$.
- BDD-isomorfismi suoraviivainen tarkistaa.

© 2001 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Kanoninen esitystapa hyödyllinen:

- Loogisen ekvivalenssin tarkastus: lauseiden ROBDD:t isomorfiset.
- Toteutuvuuden tarkastus: lauseen ROBDD ei ole isomorfinen funktion **0** graafin (yksi solmuinen graafi, jossa ainoa solmu on 0 niminen terminaali) kanssa.
- Pätevyyden tarkastus: lauseen ROBDD on isomorfinen funktion **1** graafin kanssa.
- Loogisen seuraavuuden tarkastus $\{P\} \models Q$: lauseen $\neg P \vee Q$ ROBDD on isomorfinen funktion **1** graafin kanssa.

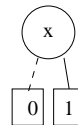
Huomio. Muistanet seuraavat yhteydet:

$$\{P\} \models Q \iff \models P \rightarrow Q \iff \models \neg P \vee Q.$$

OBDD:n rakentaminen

- Kootaan vaiheittain operaattoreilla alkaen
 - (a) vakiofunktioiden **0** ja **1** graafeista 0 ja 1 sekä

- (b) muuttujia esittävistä graafeista:



- Tyypillisiä operaattoreita:

—Apply(op , f, g): $f \text{ op } g$

Esimerkki. $f \wedge g$, $f \oplus g$, missä \oplus on poissulkeva disjunktio.

—Restrict(f, x_i, k):

$$f_{x_i \leftarrow k}(x_1, \dots, x_n) = f(x_1, \dots, x_{i-1}, k, x_{i+1}, \dots, x_n)$$

Esimerkki. $(x \wedge y)_{x \leftarrow 0} = \mathbf{0}$, $(x \wedge y)_{x \leftarrow 1} = y$

Muita operaattoreita

► if-then-else: $\text{Ite}(f, g, h) = (f \wedge g) \vee (\neg f \wedge h)$

► Komplementti: $\neg f = f \oplus \mathbf{1}$

► Kompositio: $f_{x_i \leftarrow g}(x_1, \dots, x_n) =$
 $f(x_1, \dots, x_{i-1}, g(x_1, \dots, x_n), x_{i+1}, \dots, x_n)$

$$f_{x \leftarrow g} = (\neg g \wedge f_{x \leftarrow 0}) \vee (g \wedge f_{x \leftarrow 1})$$

► Kvantifiointi:

$$\exists x f = f_{x \leftarrow 0} \vee f_{x \leftarrow 1}$$

$$\forall x f = f_{x \leftarrow 0} \wedge f_{x \leftarrow 1}$$

Toteutustekniikkaa

► Operaatiot Boolean funktioille voidaan toteuttaa graafialgoritmeilla funktioiden OBDD-esityksille siten, että muuttujien järjestys säilyy ja tuloksena on täysin redusoitu OBDD tulosfunktiolle.

► Riittäisi toteuttaa Apply() ja Restrict() mutta tehokkuussyistä toteutetaan suoraan usein myös esim. kompositio ja if-then-else.

► Restrict(f, x, k)

Lasketaan muuttamalla funktion f OBDD-esitystä siten, että siirretään kuhunkin solmuun v , jolle $\text{var}(v) = x$, tulevat kaaret suoraan solmuun $\text{lo}(v)$, jos $k = 0$ ja solmuun $\text{hi}(v)$, jos $k = 1$. Samalla redusoidaan tulosgraafi suoraan käyttäen annettuja redusointisääntöjä.

Apply(op , f, g):

- Toteutetaan syvyysuuntaisesti funktioiden f ja g graafeille.
- Perustuu operaattorin op seuraavaan ominaisuuteen

$$f \text{ op } g = (\neg x \wedge (f_{x \leftarrow 0} \text{ op } g_{x \leftarrow 0})) \vee (x \wedge (f_{x \leftarrow 1} \text{ op } g_{x \leftarrow 1}))$$

- ja OBDD-esityksen ominaisuuteen: jos r_f on funktion f OBDD:n juurisolmu ja $x \leq \text{var}(r_f)$,

$$f_{x \leftarrow b} = \begin{cases} r_f, & x < \text{var}(r_f) \\ \text{lo}(r_f), & x = \text{var}(r_f) \text{ and } b = 0 \\ \text{hi}(r_f), & x = \text{var}(r_f) \text{ and } b = 1 \end{cases}$$

- Redusoitu OBDD voidaan laskea sovellettaessa operaattoria op .
- Optimoidaan huomioimalla dominoivat arvot. Esim. $f \wedge \mathbf{0} = \mathbf{0}$.

Laskennallinen vaativuus

- OBDD-operaatiot voidaan suorittaa polynomisessa ajassa argumentteina olevien OBDD:den solmujen määrän suhteen.

Apply(op , f, g)	$\mathcal{O}(m_f m_g)$
Restrict(f, x, k)	$\mathcal{O}(m_f)$
Kompositio $f_{x \leftarrow g}$	$\mathcal{O}(m_f^2 m_g)$
Kvantifointi	$\mathcal{O}(m_f^2)$

missä m_f ja m_g ovat funktioiden f ja g OBDD-esityksen solmujen määrät.

- Tuloksena syntyvän OBDD:n koko on polynominen argumenttien kokojen suhteen.

Muuttujien järjestyksen vaikutus

- ROBDD:n koko ja muoto riippuvat muuttujien järjestyksestä.
- On olemassa funktioita, joilla ROBDD:n koko vaihtelee polynomisesta eksponentiaaliseen (muuttujien määrän suhteen) riippuen muuttujien järjestyksestä.

Esimerkki. $(a_1 \wedge b_1) \vee \dots \vee (a_n \wedge b_n)$.

Järjestyksellä $a_1 < b_1 < a_2 < \dots < a_n < b_n$ saadaan $2n$ ei-terminaalisolmua.

Järjestyksellä $a_1 < \dots < a_n < b_1 < \dots < b_n$ saadaan $2(2^n - 1)$ ei-terminaalisolmua.

OBDD:n koko erällä funktioluokilla:

Funktio	Koko	
	paras	huonoin
Symmetriset funktiot	lineaarinen	neliöllinen
Kokonaislukuyhteenlasku (mikä tahansa bitti)	lineaarinen	eksponentiaalinen
Kokonaislukukertolasku (keskibitit)	eksponentiaalinen	eksponentiaalinen

- Esim. 15-bittisen kertolaskupiirin OBDD:
yli $12 \cdot 10^6$ solmua (0.26 GB muistia).
Kukin lisäbitti nostaa solmujen määrän n. 2.7-kertaiseksi.
- Heuristiikkoja hyvän muuttujajärjestyksen löytämiseksi.
- Muuttujajärjestystä voidaan muuttaa dynaamisesti.

Toteutusten ominaispiirteitä

- OBDD-toteutukset perustuvat dynaamiseen (ei-paikalliseen) muistin käyttöön.
 - ☞ Tarvitaan paljon fyysistä muistia ja hyvät muistihallintamenetelmät.
- Hajautustaulut keskeinen osa tehokasta toteutustekniikkaa.
- Vaikea rinnakkaistaa.
- Tyypillisesti toteutettu (C-)kirjastona.

Sovellutuksia

- Loogiset tehtävät (toteutuvuus, pätevyys, looginen seuraavuus, kaikki mallit, mallien lukumäärä, ...).
- Digitaalisten piirien suunnittelu
 - Verifiointi:** Kahden piirin (esim. piirin määrittely ja toteutus) ekvivalenssin testaus.
 - Piirin OBDD muodostetaan lähtemällä input-signaaleista ja "tulkkamalla piiriä symbolisesti" (Apply()):
 - Olkoon komponentin input-signaaleille s_1 ja s_2 laskettu funktiot f_1 ja f_2 ja komponentti suorittaa operaation $s_1 \text{ op } s_2$, tällöin komponentin output-signaalille saadaan funktio $\text{Apply}(\text{op}, f_1, f_2)$.

Joukkojen käsittely

- ▶ Äärellinen joukko voidaan koodata vektoriksi binääriarvoja.
Olkoon A äärellinen joukko $|A| = N$. Tarvitaan $n = \lceil \log_2 N \rceil$ binääriarvoa.
Joukon A koodausfunktio $\sigma : A \mapsto \{0, 1\}^n$
 $\sigma_i(a)$ on a :n koodauksen i :s jäsen.
Usein on olemassa luonnollinen koodaus, esim. kokonaisluvut.

Joukko-operaatioiden esittäminen

- ▶ Olkoon A joukko, jolla on binäärikoodausfunktio σ .
 A :n alijoukkoja voidaan esittää karakteristisillä funktioilla.
Joukolle $S \subseteq A$ funktio $\chi_S : \{0, 1\}^n \mapsto \{0, 1\}$ missä

$$\chi_S(\vec{x}) = \bigvee_{a \in S} \bigwedge_{1 \leq i \leq n} \neg(x_i \oplus \sigma_i(a))$$

- ▶ Näin joukko-operaatiolle saadaan seuraavat esitykset:

$$\chi_\emptyset = \mathbf{0}$$

$$\chi_{S \cup T} = \chi_S \vee \chi_T$$

$$\chi_{S \cap T} = \chi_S \wedge \chi_T$$

$$\chi_{S - T} = \chi_S \wedge \neg \chi_T$$

- ▶ $S \subseteq T$ joss $S - T = \emptyset$ joss $\chi_S \wedge \neg \chi_T = \mathbf{0}$.

Tilakoneiden analyysi

(Hajautetut järjestelmät, protokollat, sekvenssiipiirit ...)

- Tilat ja syöteakkosto koodataan binäärisesti.
- Siirtymärelaatio koodataan binäärisesti karakteristisella funktiolla $\delta(\vec{x}, \vec{o}, \vec{n})$, joka saa arvon 1 joss syötteellä \vec{x} tapahtuu siirtymä tilasta \vec{o} tilaan \vec{n} .
- Saavutettavuusanalyysi voidaan tehdä symbolisesti.
- Joskus voidaan analysoida järjestelmiä, joiden tilavaruus on valtava (yli 10^{20} tilaa).

Symbolinen saavutettavuusanalyysi

- Relaatio S : siirtymä tilasta \vec{o} tilaan \vec{n} jollakin syötteellä.
Karakteristinen funktio: $\chi_S(\vec{o}, \vec{n}) = \exists \vec{x} \delta(\vec{x}, \vec{o}, \vec{n})$
- $R(\vec{s})$ (saavutettavat tilat tilasta \vec{q}): $\chi_R(\vec{s}) = \chi_{S^*}(\vec{q}, \vec{s})$
missä S^* on relaation S transitiivinen sulkeuma.
- χ_{S^*} voidaan laskea iteratiivisesti käyttäen identiteetterelaatiota (I) ja relaatioiden kompositiota (\circ).
- Relaatioiden kompositio: $\chi_{R \circ S} = \exists z (\chi_R(x, z) \wedge \chi_S(z, y))$

- Transitiivinen sulkeuma saadaan, kun sarja

$$S_0 = I$$

$$S_{i+1} = I \cup (S \circ S_i)$$

suppenee, t.s. löytyy i , jolle $S_i = S_{i-1}$.

- Laskenta voidaan toteuttaa OBDD-tekniikalla käyttämällä relaatioille karakteristisia funktioita.
- Toteutuksen kannalta on keskeistä, että OBDD-tekniikka tarjoaa tehokkaan ekvivalenssitestin:
sarja suppenee, kun χ_{S_i} ja $\chi_{S_{i-1}}$ ekvivalentteja.
- Iteraatioiden määrää voidaan vähentää käyttämällä sarjaa

$$S_{(0)} = I \cup S$$

$$S_{(i+1)} = S_{(i)} \circ S_{(i)}$$

Arviointia

- OBDD-tekniikka lyönyt läpi esim. digitaalipiirien ja tilakoneiden verifiointissa.
- Vaatii paljon muistia.
- Hyvän muuttujajärjestyksen löytyminen keskeinen kysymys.
- BDD sisältää "kaikki ratkaisut."
Näin ollen BDD-tekniikka ei ole parhaimmillaan, kun halutaan löytää yksi ratkaisu monen joukosta (esim. toteutuvuus, kombinatoriset ongelmien ratkaiseminen).