

## Binäriset päätösdiagrammit

- Binäärinen päätösdiagrammi (binary decision diagram; BDD): Boolean funktion esitys graafina [Le, 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.
- OBD-tekniikalla pystytään käsittelemään suuria sovellutuksissa esiintyviä Boolean funktioita (digitaaliset piirit, tilakoneet, ...)

## Redusoitu järjestetty binäärinen päätösdiagrammi (ROBDD)

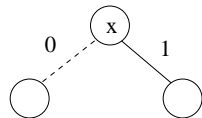
Valitaan muuttujille täydellinen järjestys  $<$ . Jokaisen solmun  $u$  ja  $u$ :n ei-terminaali lapsen  $v$  tulee toteuttaa vaatimus  $\text{var}(u) < \text{var}(v)$ .

### Reduktiosäännöt:

- Poistetaan terminaalien kopiot:  
Hävitetään terminaalisolmun 0/1 kopiot yhtä lukuunottamatta ja siirretään poistettuihin kopioihin tulleet kaaret jäljelle jäävään.
- Poistetaan ei-terminaalien kopiot:  
Jos ei-terminaalisolmuille  $v$  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-terminaalisolmulle  $v$  pätee  $\text{hi}(v) = \text{lo}(v)$ , hävitetään  $v$  ja siirretään siihen tulevat kaaret solmuun  $\text{lo}(v)$ .

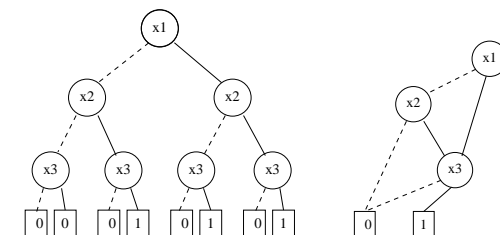
## Binäärinen päätösdiagrammi

- 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.



### 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

Reduosit OBDD

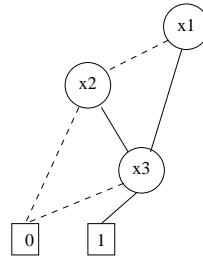
## BDD:t ohjelminä

- 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
```



© 2004 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

## Loogiset päättelyongelmat

- *Looginen ekvivalenssi*: lauseiden ROBDD:t isomorfiset.
- *Toteutus*: lauseen ROBDD ei ole isomorfinen funktion **0** ROBDD:n (terminaalisolmusta 0 koostuva yksisolmuinen BDD) kanssa.
- *Pätevyys*: lauseen ROBDD on isomorfinen funktion **1** ROBDD:n (terminaalisolmusta 1 koostuva yksisolmuinen BDD) kanssa.
- *Looginen seuraavuus*:  $\{P\} \models Q \iff$  lauseen  $\neg P \vee Q$  ROBDD on isomorfinen funktion **1** graafin kanssa.

**Huomio.** Muistat seuraavat yhteydet:

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

© 2004 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

## Kanoninen esitystapa

- ROBDD (*maksimaalisesti redusoitu järjestetty* BDD) antaa kanonisen (yksikäsitteisen) esitystavan Boolean funktiolle.
- Kaikki annettua Boolean funktiota  $f$  esittävät ROBDD:t ovat *isomorfisia* seuraavalla tavalla.



**Määritelmä.** 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

- (i) sekä  $v$  että  $v'$  ovat terminaalisolmuja, joilla on sama arvo tai
- (ii) sekä  $v$  että  $v'$  ovat ei-terminaalisolmuja siten, että  $\text{var}(v) = \text{var}(v')$ ,  $f(\text{lo}(v)) = \text{lo}(v')$  ja  $f(\text{hi}(v)) = \text{hi}(v')$ .

- BD-isomorfia on suoraviivainen tarkistaa.

© 2004 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

## OBDD:n rakentaminen

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

- (b) muuttujia esittävistä graafeista:



- Tyypillisiä operaattoreita:
  - $\text{Apply}(op, f, g): f op g$
  - $\text{Restrict}(f, x_i, k): f_{x_i \leftarrow k}(x_1, \dots, x_n) = f(x_1, \dots, x_{i-1}, k, x_{i+1}, \dots, x_n)$

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

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

© 2004 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

**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:

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

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

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.  $f \wedge \mathbf{0} = \mathbf{0}$ .

**Toteutustekniikkaa**

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

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

► Restrict( $f, x, k$ )

Lasketaan muuttamalla funktion  $f$  OBD-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.

**Laskennallinen vaativuus**

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

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

missä  $m_f$  ja  $m_g$  ovat funktioiden  $f$  ja  $g$  OBD-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.** Tarkastellaan lausetta  $(a_1 \wedge b_1) \vee \dots \vee (a_n \wedge b_n)$ .

1. Järjestyksellä  $a_1 < b_1 < a_2 < \dots < a_n < b_n$  saadaan  $2n$  ei-terminaalisolmua.
2. Järjestyksellä  $a_1 < \dots < a_n < b_1 < \dots < b_n$  saadaan  $2(2^n - 1)$  ei-terminaalisolmua.

## Toteutusten ominaispiirteitä

- OBD-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.

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.

## 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 "tulkaamalla piiriä symbolisesti" (operaatio Apply( $\cdot$ )):
   
Jos komponentin input-signaaleille  $s_1$  ja  $s_2$  on 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 s.e.  $|A| = N$ .  
Tarvitaan  $n = \lceil \log_2 N \rceil$  binääriarvoa.  
Joukon  $A$  koodausfunktio  $\sigma : A \rightarrow \{0, 1\}^n$   
 $\sigma_i(a)$  on  $a$ :n koodauksen  $i$ :s jäsen.  
Usein on olemassa luonnollinen koodaus, esim. kokonaisluvut.

## Tilakoneiden analyysi

(Hajautetut järjestelmät, protokollat, sekvenssipiirit ...)

- ▶ 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).

## Joukko-operaatioiden esittäminen

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

$$\begin{aligned} \chi_{\emptyset} &= \mathbf{0} \\ \chi_{S \cup T} &= \chi_S \vee \chi_T \\ \chi_{S \cap T} &= \chi_S \wedge \chi_T \\ \chi_{S - T} &= \chi_S \wedge \neg \chi_T \end{aligned}$$

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

## 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.
- ▶  $\chi_{S^*}$  voidaan laskea iteratiivisesti käyttäen identiteettirelaatiota ( $I$ ) 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

$$S_0 = I$$

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

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

- Laskenta voidaan toteuttaa OBD-tekniikalla käyttämällä relaatioille karakteristisia funktioita.
- Toteutuksen kannalta on keskeistä, että OBD-tekniikka tarjoaa tehokkaan *ekvivalenssitestin*: sarja suppenee, kun  $\chi_{S_i}$  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

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