Olekute kodeerimine

- **Leida olekutele sellised koodid, et siirde- ja väljundfunktsioonid oleksid minimaalsed**
- **Heuristilised reeglid**
  - **D-trigerid – mida rohkem siirdeid mingisse olekusse, seda vähem peaks selle kood sisaldama 1**
  - **SR-, JK- & T-trigerid – jooksva ja järgnev oleku koodid peaksid võimalikult vähe erinema**
- **Mitmevalentse loogika kasutamine**
  - **Kodeerimata automaadi siirdefunktsiooni vaadeldakse kui mitmevalentse loogika funktsioon**
  - **Eesmärgiks ridade (implikantide) arvu vähendamine**
  - **Tulemuseks minimaalne MV funktsioon, mis seab nõuded olekute kodeerimisele**
  - **Sümboolne minimeerimine ja kodeerimine**
- **Mitmetasemeline realisatsioon**
  - **Mitmetasemeline loogika funktsioonide minimeerimine**
  - **Erinevate trigeritüüpide kasutamine, olekute ümberkodeerimine**
- **Tükelduste kasutamine**
  - **eraldab olekute grupid, mis teistest ei sõltu (asendusomadustega tükeldused)**

Automaatide algebralise struktuuriteooria alused

- Olekute kodeerimine
  - siirde- ja väljundfunktsiooni lihtsustamine
- Automaadi minimeerimine
  - olekute arvu vähendamine
- Automaatide dekompositsioon ja kompositsioon
  - automaadi tükeldamine kaheks või enamaks automaadiks
  - kahe või enama automaadi ühendamine
- Tükeldused
  - binaarsuhte jagamine ekvivalentsiklassideks
  - olekute hulk (ka sisendite ja väljundite hulgad kasutusel)
  - Vastavuse φ erijuhtu, lähte- ja sihthulk langevad kokku - $D(\varphi) = R(\varphi) = A$
  - Tähistus – $R \subseteq A \times A$
  - Ekvivalentsisuhe R on refleksiivne, sümmeetriline ja transitiivne
  - Elemendi $a \in A$ ekvivalentsiklass ekvivalentsisuhtes R – $K(a) = \{ b \mid \langle a,b \rangle \in R \}$

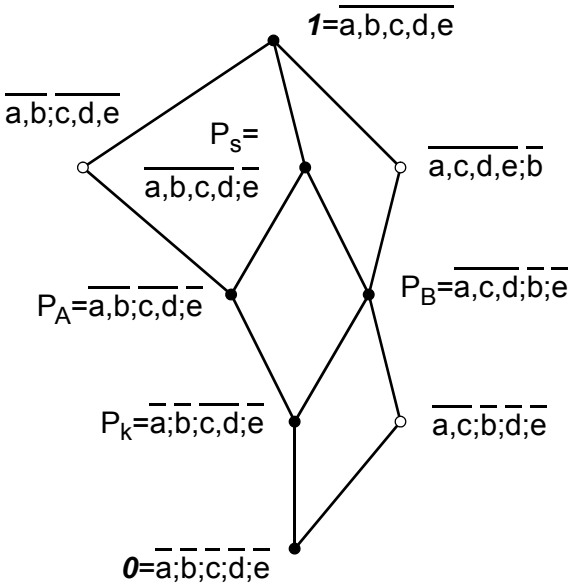
Tükeldus

- Ekvivalentsisuhe genereerib tükelduse (partition) P hulgal A
  Tükeldus P koosneb ekvivalentsiklassidest K_i, i=1,...,n
  $P = \{ K_1, K_2, ..., K_n \}$, kus $K_i \neq \varnothing$, i=1,...,n;
  $K_i \cap K_j = \varnothing$, i,j=1,...,n, $i \neq j$;
  $\cup K_i = A$
  - 0-tükeldus (nulltükeldus) koosneb 1-elemendilistest ekvivalentsi klassidest
  - 1-tükelduses (ühiktükelduses) on ainult üks ekvivalentsiklass

- $B_P(a)$ – tükelduse P ekvivalentsiklass (plokk) K_i, millesse kuulub a
  - $K_i=B_P(a) \Rightarrow a \in K_i$
- $a_1 \equiv a_2$ (P) – elemendid a_1 ja a_2 kuuluvad samasse tükelduse P plokki K_i siis ja ainult siis, kui $B_P(a_1)=B_P(a_2)$

Tükeldus – operatsioonid

- Hulk: A={a,b,c,d,e}, tükeldused: P_A={ $\overline{a,b}$; $\overline{c,d}$; $\overline{e}$ } ja P_B={ $\overline{a,c,d}$; $\overline{b}$; $\overline{e}$ }
  - P_A={ {a,b}, {c,d}, {e} } ja P_B={ {a,c,d}, {b}, {e} }
- Korrutis: $P_1 \bullet P_2$: $(a_1 \equiv a_2\ (P_1 \bullet P_2)) \Leftrightarrow (a_1 \equiv a_2\ (P_1)\ \&\ a_1 \equiv a_2\ (P_2))$
  - $P_k = P_A \bullet P_B$ = { $\overline{a,b}$; $\overline{c,d}$; $\overline{e}$ }•{ $\overline{a,c,d}$; $\overline{b}$; $\overline{e}$ } = { $\overline{a}$; $\overline{b}$; $\overline{c,d}$; $\overline{e}$ }
- Summa: $P_1 + P_2$: $(a_1 \equiv a_2\ (P_1 + P_2)) \Leftrightarrow (a_1 \equiv a_2\ (P_1)\ V\ a_1 \equiv a_2\ (P_2))$
  - $P_s = P_A + P_B$ = { $\overline{a,b}$; $\overline{c,d}$; $\overline{e}$ }+{ $\overline{a,c,d}$; $\overline{b}$; $\overline{e}$ } = { $\overline{a,b,c,d}$; $\overline{e}$ }
- Võrdlus:
  $P_1 \leq P_2 \Leftrightarrow P_1 \bullet P_2 = P_1 \Leftrightarrow [\ \forall\ K_i \in P_1\ (\ \exists\ K_j \in P_2\ [K_i \subseteq K_j]\ )\ ]$
  - $P_A \leq P_s$ – { $\overline{a,b}$; $\overline{c,d}$; $\overline{e}$ }•{ $\overline{a,b,c,d}$; $\overline{e}$ } = { $\overline{a,b}$; $\overline{c,d}$; $\overline{e}$ }
  - P_A ja P_B ei ole võrreldavad
- Tükelduste võre
  - Tehted ~~ asukoht võrel
  - Võrreldavus ~~ asukoht võrel
- Null-tükeldus (*0*) – iga element on omaette plokis
- Ühik-tükeldus (*1*) – kõik elemendid on samas plokis



Tükelduste kasutamine kodeerimisel

- Tükeldustel baseeruv lähenemine
  - automaadi olekute hulga tükeldamine selliselt, et tükelduste plokid omaksid võimalikult väikest “vastastikust sõltuvust”
- Leitakse automaadi asendusomadusega (S.P.) tükeldused
  - väljundeid ei vaadelda
  - iga tükeldus vastab sõltumatule osale
  - teostatakse paralleelne dekompositsioon
- Näide
  - S.P. tükeldused - $P_1 = \overline{1,2,3}\ \overline{4,5,6}$, $P_2 = \overline{1,6}\ \overline{2,5}\ \overline{3,4}$
  - Dekompositsiooniks vajalik, et $P_1 \cdot P_2 = P_0$
    - vajadusel geneeritakse täiendavaid tükeldusi
  - Iga tükeldus (kui automaat) kodeeritakse eraldi:
    - P_1-> 1-{1,2,3} 0-{4,5,6} & P_2-> 10-{1,6} 01-{2,5} 00-{3,4}
    - Olekute koodid: 1->110, 2->101, 3->100, 4->000, 5->001, 6->010

x	s_t	s_{t+1}	y
0	1	4	0
1		3	
0	2	6	0
1		3	
0	3	5	0
1		2	
0	4	2	1
1		5	
0	5	1	0
1		4	
0	6	3	0
1		4	

Kodeerimine – võrdlus

Automaat

x	s	s+	y
0	1	4	0
1	1	3	0
0	2	6	0
1	2	3	0
0	3	5	0
1	3	2	0
0	4	2	1
1	4	5	1
0	5	1	0
1	5	4	0
0	6	3	0
1	6	4	0

Kodeering #1

“loendur”

```
1:000   2:001
3:010   4:011
5:100   6:101
```

```
faas 1011
0001 1100
000- 0010
01-0 0100
11-- 0010
1-10 0110
0-10 1100
1-11 1101
0-11 0111
```

Kodeering #2

D-trigeri heuristika

```
1: 1   2: 2   1:110   2:010
3: 3   4: 3   3:000   4:001
5: 2   6: 1   5:100   6:101
```

```
ülesanne
0110 0010
1110 0000
0010 1010
1010 0000
0000 1000
1000 0100
0001 0101
1001 1001
0100 1100
1100 0010
0101 0000
1101 0010
-011 ----
-111 ----
```

```
faas 1111
00-0 1000
0-1- 0010
110- 0010
10-1 1001
00-1 0101
1000 0100
0100 1100
```

Kodeering #3

$\overline{1,2,3}\,\overline{4,5,6}$

$\overline{1,6}\,\overline{2,5}\,\overline{3,4}$

```
1:110   2:101
3:100   4:000
5:001   6:010
```

```
faas 1111
-000 0001
00-- 1000
11-- 1000
0--1 0100
--00 0010
```

$y = \overline{q}_1\,\overline{q}_2\,\overline{q}_3$

$d_1 = \overline{x}\,\overline{q}_1 + x\,q_1$

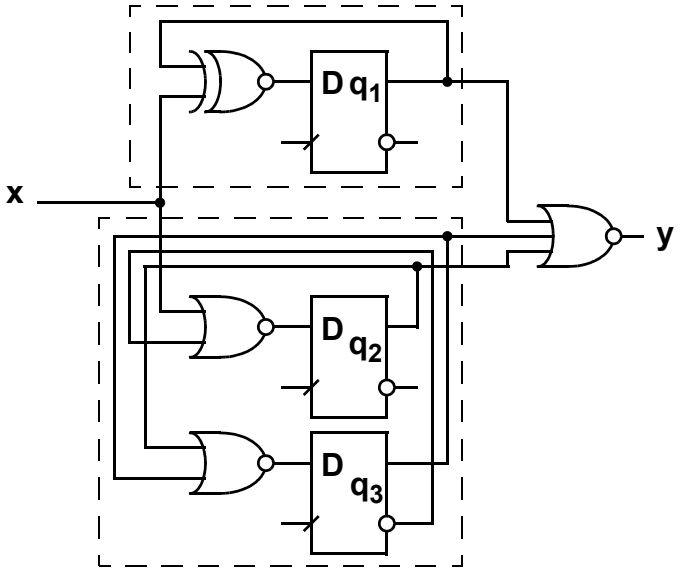
$d_2 = \overline{x}\,q_3$

$d_3 = \overline{q}_2\,\overline{q}_3$

Tükeldused, kodeering #3 ja realisatsioon

- Olekute koodid: 1->110, 2->101, 3->100, 4->000, 5->001, 6->010
- Tükeldused: $P_1=\{\overline{1,2,3};\ \overline{4,5,6}\}$, $P_2=\{\overline{1,6};\ \overline{2,3,4,5}\}$, $P_3=\{\overline{2,5};\ \overline{1,3,4,6}\}$
  - kaheplokilised tükeldused: 1 - parempoolne plokk, 0 - vasakpoolne plokk
- $P_1 \bullet P_2 \bullet P_3=\{\overline{1};\overline{2};\overline{3};\overline{4};\overline{5};\overline{6}\}=0$ & $P_1+P_2+P_3=\{\overline{1,2,3,4,5,6}\}=1$

- **Automaatide võrk - kaks paralleelselt töötavat automaati**
  - $d_1=q_1\bar{x}+\bar{q}_1\bar{x} = \bar{x}\,\mathrm{nor}(q_1,x)$
  - $d_2=q_3\bar{x} = \mathrm{nor}(\bar{q}_3,x)$
  - $d_3=\bar{q}_2\bar{q}_3 = \mathrm{nor}(q_2,q_3)$
  - $y=q_1q_2q_3 = \mathrm{nor}(\bar{q}_1,\bar{q}_2,\bar{q}_3)$
  - Tükeldused:
    - 1. automaat: $P_A=\{\overline{1,2,3};\ \overline{4,5,6}\}$ $(=P_1)$
    - 2. automaat: $P_B=\{\overline{1,6};\ \overline{2,5};\ \overline{3,4}\}$ $(=P_2 \bullet P_3)$



Asendusomadus (S.P.)

- **Automaadi M=(S,I,O,δ,λ) olekute hulga tükeldusel P on *asendusomadus* (substitution property, S.P.) siis ja ainult siis, kui**
  s $\equiv$ t(P) järeldub, et ka δ(s,i) $\equiv$ δ(t,i)(P) $\forall$ i$\in$I
  - **kui lähteolekud kuuluvad ühte tükelduse plokki, siis ka järgmised olekud peavad kuuluma ühte plokki (ei pruugi kokku langeda lähteolekute plokiga) kõikide sisendkombinatsioonide korral**
  - **tükeldus P - kasutusel ka tähistus π**

- **Asendusomadustega tükeldused:**
  - **P_A = {$\overline{1,2,3}$; $\overline{4,5,6}$}**
  - **P_B = {$\overline{1,6}$; $\overline{2,5}$; $\overline{3,4}$}**

- **0-tükeldus ja 1-tükeldus on S.P. tükeldused**

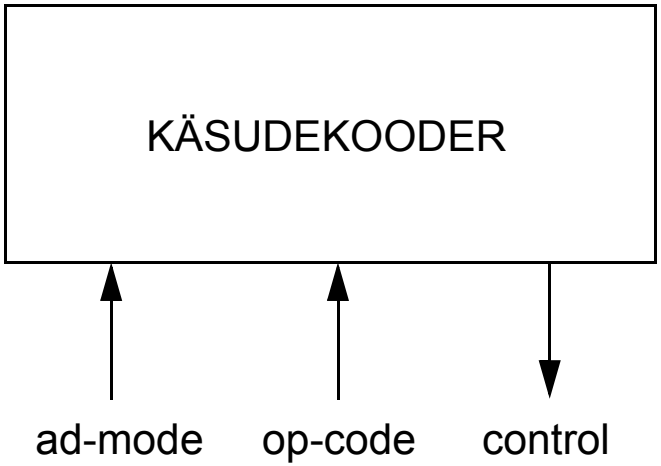
x	s_t	s_{t+1}	y
0	1	4	0
1		3	
0	2	6	0
1		3	
0	3	5	0
1		2	
0	4	2	1
1		5	
0	5	1	0
1		4	
0	6	3	0
1		4	

Sümboolne minimeerimine ja kodeerimine

- Sümbolite tabelite minimeerimine binaarsete tabelite asemel
- Kahend- ja MV-loogika funktsioonide minimeerimise edasiarendus
  - Rakendused – kodeerimine
    - operatsioonikoodid, automaadi olekud, jne.
  - Lahendatavad probleemid
    - sisendite kodeerimine
    - väljundite kodeerimine
    - kombineeritud kodeerimine
- Definitsioone
  - Sümbolkate (symbolic cover)
    - sümbolimplikantide loetelu (tabeliridade loetelu)
  - Sümbolimplikant (symbolic implicant)
    - sümbol-literaalide konjunktsioon
  - Sümbol-literaal
    - lihtne – üks sümbol; liit – mõne sümboli disjunktsioon

Näide

ad-mode	op-code	control
INDEX	AND	CNTA
INDEX	OR	CNTA
INDEX	JMP	CNTA
INDEX	ADD	CNTA
DIR	AND	CNTB
DIR	OR	CNTB
DIR	JMP	CNTC
DIR	ADD	CNTC
IND	AND	CNTB
IND	OR	CNTD
IND	JMP	CNTD
IND	ADD	CNTC



Sisendite kodeerimine

- Suurem vabadus koodide valikul
- Eesmärk – esitusviisi suuruse vähendamine
- Lähenemised
  - kodeerimine ridade arvu (koodikombinatsioonide) vähendamiseks
  - kodeerimine bittide arvu (koodipikkuse) vähendamiseks
- Lahendus
  - lähtekodeering – “1-hot” e. bit iga sümboli kohta
  - MV kodeering minimeerijatele
  - tabel esitada funktsioonide süsteemina
    - minimeerida MV abil
  - tulemuse interpreteerimine
    - liit-MV-literaalid
    - sümbolite grupid

Näide

ad-mode	op-code	control
INDEX	AND	CNTA
INDEX	OR	CNTA
INDEX	JMP	CNTA
INDEX	ADD	CNTA
DIR	AND	CNTB
DIR	OR	CNTB
DIR	JMP	CNTC
DIR	ADD	CNTC
IND	AND	CNTB
IND	OR	CNTD
IND	JMP	CNTD
IND	ADD	CNTC

1-hot / MV	
100 1000	1000
100 0100	1000
100 0010	1000
100 0001	1000
010 1000	0100
010 0100	0100
010 0010	0010
010 0001	0010
001 1000	0100
001 0100	0001
001 0010	0001
001 0001	0010

1-hot / MV	
100 1111	1000
010 1100	0100
001 1000	0100
010 0011	0010
001 0001	0010
001 0110	0001

ad-md.	op-code	ctrl
INDEX	AND,OR,JMP,ADD	CNTA
DIR	AND,OR	CNTB
IND	AND	CNTB
DIR	JMP,ADD	CNTC
IND	ADD	CNTC
IND	OR,JMP	CNTD

Sisendite kodeerimine – lahendamine

- Minimaalse sümbolkatte teisendamine minimaalseks binaarkatteks
- Sümbolimplikantide vastavus kahendimplikantidega
- Liitimplikandid
  - vastavad sümbolid tuleb kodeerida selliselt, et vastav superkuup ei sisalda teisi sümboleid
- Sümbolid asendada kattes koodidega

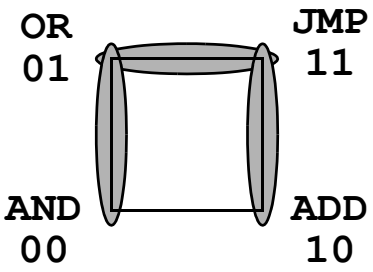
Liitliteraalid

AND,OR,JMP,ADD

AND,OR

JMP,ADD

OR,JMP



AND	00
OR	01
JMP	11
ADD	10

INDEX	00
DIR	01
IND	11

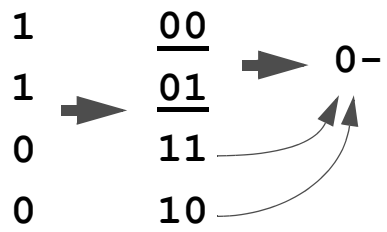
Kodeeritud kate

ad.	op.	ctrl
00	--	1000
01	0-	0100
11	00	0100
01	1-	0010
11	10	0010
11	-1	0001

Kodeerimisalgoritmid

- **Piirangute maatriks *A***
  - $a_{ij} = 1$, kui sümbol *j* kuulub literaali *i*
  - veerud – sümbolid, read - liitliteraalid
- **Koodide maatriks *E***
  - *E* rahuldab *A* seatud piiranguid, kui iga rea a^T (*A*-st) puhul 1-dele vastavate *E* ridade superkuup ei oma ühisosa ühegi 0-dele vastava *E* reaga
  - read – sümbolid, veerud – koodi bitid

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad E = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$



Kodeerimisalgoritmid (järg)

- Leida selline minimaalse veergude arvuga koodide maatriks E, mis rahuldab maatriksiga A ette antud piiranguid
  - täpsed algoritmid katte leidmise või värvimise teel, baseeruvad dihhotoomial
- *Dihhotoomia* (dichotomy)
  - kaks hulka (L;R)
  - sümbolite hulga alamhulga bikromaatiline tükeldus
  - kodeerimine – E veergude hulk – sümbolite alamhulga bikromaatiliste tükelduste hulk
- Rea a^T-ga seotud dihhotoomia – paar (L;R)
  - L – sümbolid, millele vastab 1 a^T-s
  - R – sümbolid, millele vastab 0 a^T-s
- Rea a^T-ga seotud *algdihhotoomia* (seed dichotomy) – paar (L;R)
  - L – sümbolid, millele vastab 1 a^T-s
  - R – *üks* sümbol, millele vastab 0 a^T-s

Kodeerimisalgoritmid (järg)

- Näide – $a^T = 1100$
  - dihhotoomia – *({AND,OR};{JMP,ADD})*
  - algdihhotoomiad – *({AND,OR};{JMP}) ja ({AND,OR};{ADD})*
- Sobivus (compatibility) – $(L_1;R_1)$ ja $(L_2;R_2)$ on sobivad, kui
  - $L_1 \cap R_2 = \varnothing$ ja $R_1 \cap L_2 = \varnothing$ või
  - $L_1 \cap L_2 = \varnothing$ ja $R_1 \cap R_2 = \varnothing$
- Kaetus (covering) – $(L_1;R_1)$ katab $(L_2;R_2)$, kui
  - $L_1 \supseteq L_2$ ja $R_1 \supseteq R_2$ või
  - $L_1 \supseteq R_2$ ja $R_1 \supseteq L_2$
- Lihtdihhotoomia (prime dichotomy)
  - dihhotoomia, mis pole kaetud mitte ühegi teise sobiva antud hulga dihhotoomiaga

Täpne sisendite kodeerimine

- Leia kõik lihtdihhotoomiad
- Moodusta liht-/algdihhotoomiate tabel
- Leia minimaalne algdihhotoomiate kate lithdihhotoomiate poolt

Algdihhotoomiad

s_1 ({AND,OR}; {JMP})
s_2 ({AND,OR}; {ADD})
s_3 ({JMP,ADD}; {AND})
s_4 ({JMP,ADD}; {OR})
s_5 ({OR,JMP}; {AND})
s_6 ({OR,JMP}; {ADD})

	s_1	s_2	s_3	s_4	s_5	s_6
p_1	1	1	1	1	0	0
p_2	0	0	0	0	1	1
p_3	0	0	1	0	1	0
p_4	0	1	0	0	0	1

Lihtdihhotoomiad

p_1 ({AND,OR}; {JMP,ADD})
p_2 ({OR,JMP}; {AND,ADD})
p_3 ({OR,JMP,ADD}; {AND})
p_4 ({AND,OR,JMP}; {ADD})

p_1 & p_2

$$E = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Heuristiline sisendite kodeerimine

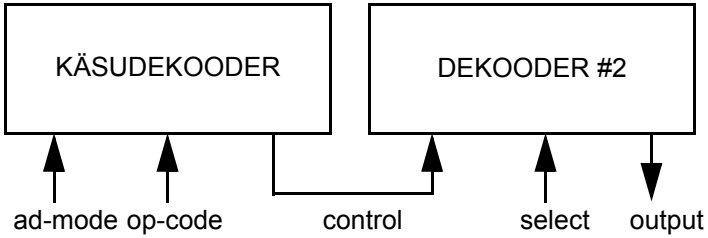
- Leia A ridadele vastavad dihhotoomiad
- Konstrueeri E veerg-veeru haaval
- Itereeri
  - leia maksimaalne sobiv hulk
  - leia sobiv kodeering
  - kasuta seda E veeruna

Näide

- Dihhotoomiad
  - d_1 ({AND,OR}); {(JMP,ADD)}
  - d_2 ({JMP,ADD}; {AND,OR})
  - d_3 ({OR,JMP}; {AND,ADD})
- Kaks esimest on omavahel sobivad – veerg $[1100]^T$ rahuldab d_1, d_2
- Vaja rahuldada piirang d_3
- Teine kodeeriv veerg on $[0110]^T$

Väljundite kodeerimine ja kombineeritud kodeerimine

- **Laiendus MV minimeerimisele**
- **Arvestab**
  - **kattuvuse suhteid**
  - **disjunktiivseid suhteid**



ad-md.	op-code	ctrl
INDEX	AND,OR,JMP,ADD	CNTA
DIR	AND,OR	CNTB
IND	AND	CNTB
DIR	JMP,ADD	CNTC
IND	ADD	CNTC
IND	OR,JMP	CNTD

CNTD kood katab _CNTB_ & _CNTC_ koodid

100 1111	CNTA
011 1100	CNTB
011 0011	CNTC
001 0110	CNTD

Kodderitud kate

ad.	op.	ctrl
00	--	00
-1	0-	01
-1	1-	10
11	-1	11

Kahendotsustus diagrammid
(Binary Decision Diagrams)
BDD

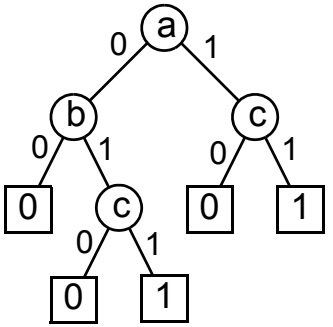
- **Efektiivne loogikafunktsioonide esitus**
  - **kasutusse – Lee & Akers**
  - **kanooniline vorm – Bryant**
- **Manipuleerimine loogikafunktsioonidega**
- **Kasutatavus ka teistes valdkondades**
  - **hulgad ja suhted**
  - **simuleerimine, testimine, jne.**

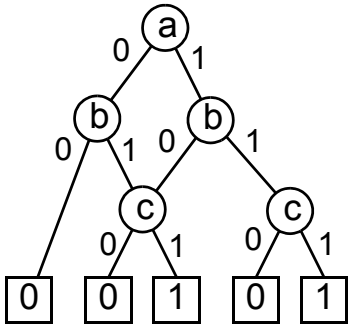
Definitsioone

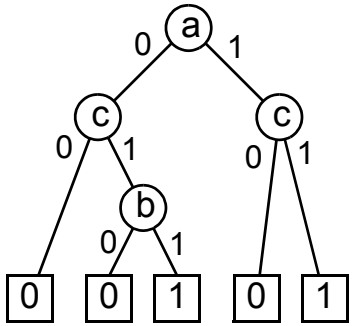
- **BDD**
  - puu või juurega suunatud atsükliline graaf (DAG)
  - iga sõlm sisaldab otsustust
- **Järjestatud BDD (OBDD, Ordered BDD)**
  - igale otsustusele vastab kahendmuutuja
  - puul (või DAG-l) võib luua tasandid selliselt, et iga tasand vastaks ühele muutujale
  - juurega suunatud atsükliline graaf
  - iga mitte-leht-sõlmega (v) on seotud
    - viit muutujale – *index(v)*
    - kaks järglast (children) – *low(v)* & *high(v)*
  - iga leht-sõlmega on seotud väärtus (1 või 0)
  - järjestus
    - index(v) < index(low(v))
    - index(v) < index(high(v))

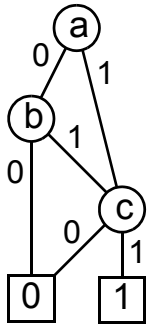
f=(a+b)c

f=a'(b(c))+a(c)





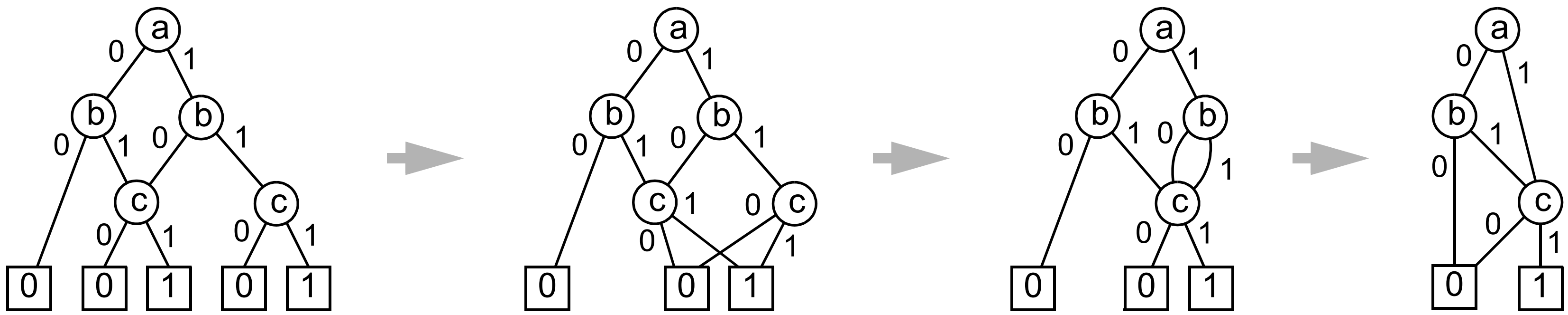




Teisendused

f=(a+b)c

f=a’(b(c))+a(c)



OBDD omadusi

- Iga OBDD, mille juur on *v*, defineerib funktsiooni f^v :
  - f^v =1, kui *v* on leht väärtusega *value(v)*=1
  - f^v =0, kui *v* on leht väärtusega *value(v)*=0
  - $f^v = x'_i \cdot f^{low(v)} + x_i \cdot f^{high(v)}$, kui *v* ei ole leht ja *index(v)*=i
- Funktsiooni võib kirjeldada mitme erineva OBDD-ga
- OBDD suurus sõltub muutujate järjekorrast

- Shannoni arendus - $f = x'_i \cdot f_{xi=0} + x_i \cdot f_{xi=1}$

Näide

f = a'b+ab'+bc'd+a'd'

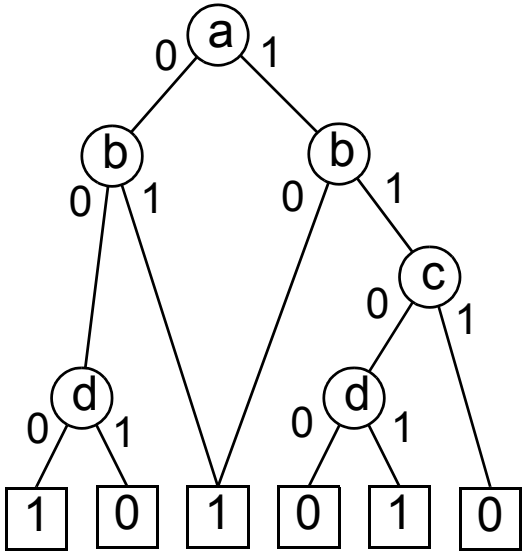
....	$\bar{a}$	a
abcd	*bcd	*bcd
01--	*1--	~~*1--~~
10--	~~*0--~~	*0--
-101	~~*101~~	*101
0--0	*--0	~~*--0~~

f = a'(b+d')+a(b'+bc'd)

$\overline{ab}$	$\bar{a}\bar{b}$	$a\bar{b}$	ab
**cd	**cd	**cd	**cd
~~**--~~	**--	**--	~~**--~~
**-0	~~**--0~~	~~**01~~	**01

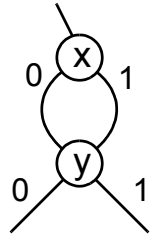
f = a'(b'(d')+b(1))+a(b'(1)+b(c'd))

f = a'(b'(d'(1)+d(0))+b(1))+
a(b'(1)+b(c'(d'(0)+d(1))+c(0)))



ROBDD - Reduced Ordered Binary Decision Diagrams

- **Liiasuste vaba**
  - $\overline{\exists}$ **sellist sõlme, et** ***low(v)=high(v)***
  - $\overline{\exists}$ **sellist sõlmede paari** $\{u,v\}$**, et** $f^u = f^v$**,**
    s.t. et vastavad alamgraafid (alam-BDD-d) oleksid isomorfsed

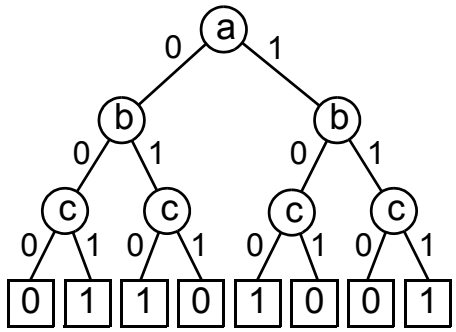


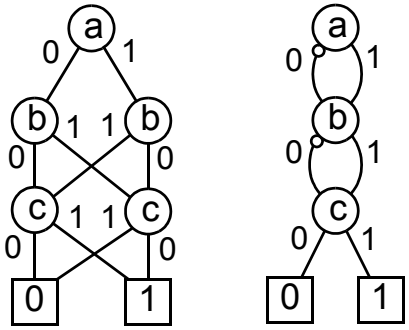
- **Redutseerimine on saavutatav polünomiaalse ajaga**
- **ROBDD – kanooniline esitusviis**
- **Kanooniline esitusviis lubab:**
  - **kontrollida loogilist ekvivalentsust konstantse ajaga (ehitusaeg v.a.)**
  - **tautoloogia ja loogika-operatsioonide teostamine ajaga, mis on proportsionaalne graafi suurusega (sõlmede arvuga)**
- **Puudus – suurus sõltub muutujate järjekorrast!**

ROBDD omadused ja laiendusi

- **ROBDD suurused**
  - korrutid – ekponentsiaalne
  - liitjad – eksponentsiaalne kuni lineaarne
  - suvaloogika – heuristiline lähenemine aitab hoida BDD-d väikesed
- **ROBDD laiendusi**
  - Inverteeritud servad (complemented edges)
    - vähendab ROBDD suurust
    - täiendi leidmine konstantse ajaga
    - kanoonilisuse säilitamiseks eksisteerivad piirangud, kuhu võib luua inverteeritud servi
  - Eba-oluline (don't-care) leht esitamaks mittetäielikult määratud funktsioone
- **ROBDD eeliseid**
  - Mitmed ROBDD manipuleerivad algoritmid töötavad polünomiaalse ajaga
  - Tihti on ROBDD väikesed
  - Suur valik tarkvarapakette saadaval

x=a⊕b⊕c





Muutujate järjestus

- Leida selline muutujate järjestus, et funktsiooni esitav ROBDD oleks minimaalse suurusega
- Täpne lahend eksponentsiaalse ajaga (intractable problem) - $O(n^2 \cdot 3^n)$
- Heuristiline staatiline järjestus
  - Muutujad on järjestatud ette-antud skeemi struktuuri alusel
  - Põhjendus – muutujad, mis mõjutavad väljundi(te)le lähedal olevaid loogikalülisid, peaksid olema diagrammi põhjas, sest nad mõjutavad ainult osa funktsioonist
  - Meetod – muutuja järjekord sõltub tema kaugusest väljundini

Dünaamiline muutujate ümberjärjestamine

- Loogikafunktsioonidega manipuleerides ei pruugi esialgne muutujate järjestus olla enam piisavalt hea
- Tarkvarapaketid
  - iteratiivne naabermuutujate vahetamine
  - piirang – tabeleid tuleks muuta nii vähe kui võimalik

- Kihid vahetatavatest muutujates ülal- ja allpool ei muutu
- Luuakse kaks uut sõlme
  - võivad juba tabelis eksisteerida
- *Nihutamine*
  - üks muutuja korraga nihutatakse uuele positsioonile
  - korratakse kõigi muutujatega

Dünaamiline muutujate ümberjärjestamine (järg)

$$F = (x_i,F_1,F_0) = x_i \cdot F_1 + x'_i \cdot F_0$$

$$F = (x_i,F_1,F_0) = (x_i , x_{i+1} \cdot F_{11} + x'_{i+1} \cdot F_{10} , x_{i+1} \cdot F_{01} + x'_{i+1} \cdot F_{00})$$

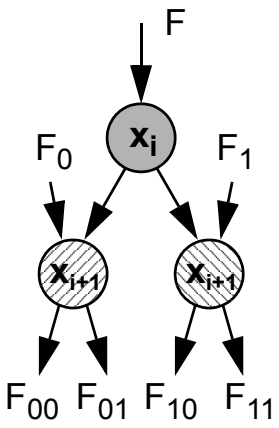
$$F = (x_i,F_1,F_0) = x_i \cdot x_{i+1} \cdot F_{11} + x_i \cdot x'_{i+1} \cdot F_{10} + x'_i \cdot x_{i+1} \cdot F_{01} + x'_i \cdot x'_{i+1} \cdot F_{00}$$

$$F = (x_i,F_1,F_0) = (x_{i+1},(x_i,F_{11},F_{01}),(x_i,F_{10},F_{00}))$$

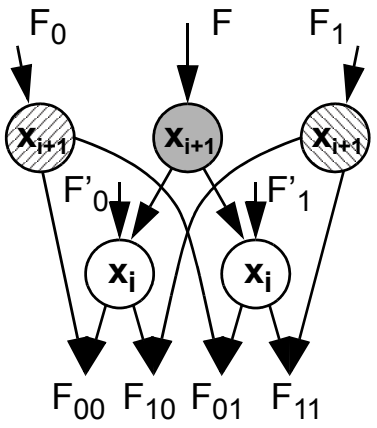
$F \quad x_i \quad F_0 \quad F_1$

$F_0 \quad x_{i+1} \quad F_{00} \quad F_{01}$

$F_1 \quad x_{i+1} \quad F_{10} \quad F_{11}$



vana



uus

$* \; F \quad x_{i+1} \quad F'_0 \quad F'_1$

$+ \; F'_0 \quad x_i \quad F_{00} \quad F_{10}$

$+ \; F'_1 \quad x_i \quad F_{01} \quad F_{11}$

$F_0 \quad x_{i+1} \quad F_{00} \quad F_{01}$

$F_1 \quad x_{i+1} \quad F_{10} \quad F_{11}$

Dünaamiline muutujate ümberjärjestamine (järg)

$$F = (x_i,F_1,F_0) = x_i \cdot F_1 + x'_i \cdot F_0$$

$$F_1 = (x_{i+1},F_{11},F_{10}) = x_{i+1} \cdot F_{11} + x'_{i+1} \cdot F_{10}$$

$$F_0 = (x_{i+1},F_{01},F_{00}) = x_{i+1} \cdot F_{01} + x'_{i+1} \cdot F_{00}$$

$$F = (x_i,F_1,F_0) = x_i \cdot (x_{i+1} \cdot F_{11} + x'_{i+1} \cdot F_{10}) + x'_i \cdot (x_{i+1} \cdot F_{01} + x'_{i+1} \cdot F_{00})$$

$$F = (x_i,F_1,F_0) = x_i \cdot x_{i+1} \cdot F_{11} + x_i \cdot x'_{i+1} \cdot F_{10} + x'_i \cdot x_{i+1} \cdot F_{01} + x'_i \cdot x'_{i+1} \cdot F_{00}$$

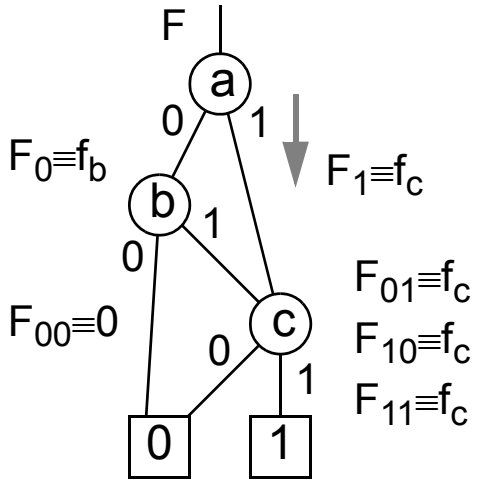
$$F = (x_i,F_1,F_0) = x_{i+1} \cdot x_i \cdot F_{11} + x_{i+1} \cdot x'_i \cdot F_{01} + x'_{i+1} \cdot x_i \cdot F_{10} + x'_{i+1} \cdot x'_i \cdot F_{00}$$

$$F = (x_i,F_1,F_0) = x_{i+1} \cdot (x_i \cdot F_{11} + x'_i \cdot F_{01}) + x'_{i+1} \cdot (x_i \cdot F_{10} + x'_i \cdot F_{00})$$

$$F = (x_i,F_1,F_0) = (x_{i+1},F'_1,F'_0)$$

$$F = (x_i,F_1,F_0) = (x_{i+1},(x_i,F_{11},F_{01}),(x_i,F_{10},F_{00}))$$

Dünaamiline muutujate ümberjärjestamine (näide)



$$F = (x_i, F_1, F_0) = (x_{i+1}, (x_i, F_{11}, F_{01}), (x_i, F_{10}, F_{00}))$$

F	x_i	F_0	F_1
F_0	x_{i+1}	F_{00}	F_{01}
F_1	x_{i+1}	F_{10}	F_{11}

→

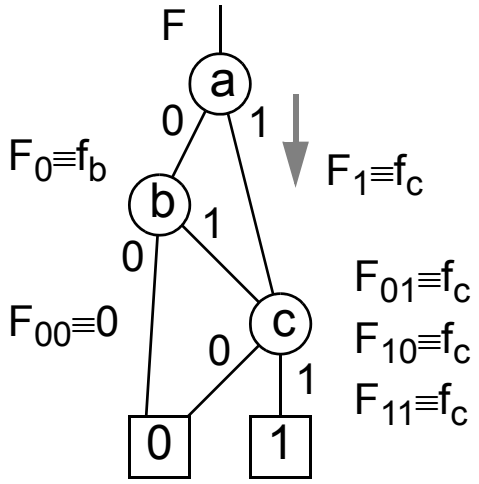
* F	x_{i+1}	F'_0	F'_1
+ F'_0	x_i	F_{00}	F_{10}
+ F'_1	x_i	F_{01}	F_{11}
F_0	x_{i+1}	F_{00}	F_{01}
F_1	x_{i+1}	F_{10}	F_{11}

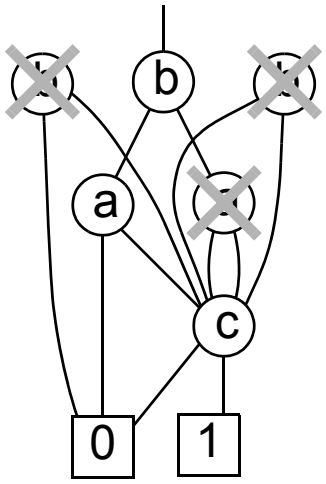
$F = a'bc + ac$

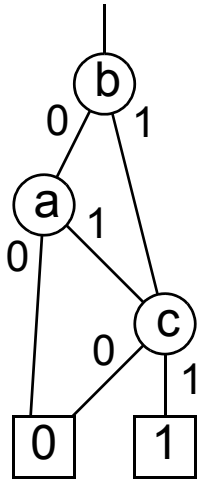
$[\ F = c\ (\ a + b\ )\ ]$

$$(a, f_c, f_b) = (\ b,\ (a, f_c, f_c),\ (a, f_c, 0)\ )$$

Dünaamiline muutujate ümberjärjestamine (näide)







$(a,f_c,f_b)=(b,(a,f_c,f_c),(a,f_c,0))$

$(a,f_c,f_b)=(b,f_c,(a,f_c,0))$

F = a'bc + ac

F = b'ac + bc

F = c (a + b)