Mitmetasemeline loogikafunktsioonide minimeerimine

1. Kahetasemeline minimeerimine ja mitmetasemeline realisatsioon

abc	xy
000	-0
001	11
010	00
011	00
100	10
101	11
110	11
111	0-

mintermid

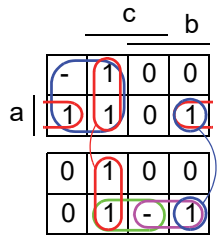
gr.	abc	e
0 o	000	10
1	001	10
	001	01
	100	10
2	101	10
	101	01
	110	10
	110	01
3 o	111	01

1. etapp

gr.	abc	e
0	00-	10*
	-00	10*
1	001	11*
	-01	10*
	-01	01*
	10-	10*
	1-0	10 A
2	101	11*
	110	11 B
	1-1	01 C
	11-	01 D

2. etapp

gr.	abc	e
0	-0-	10 E
1	-01	11 F



lihtimplikandid

abc	e	A	B	C	D	E	F
~~001~~	~~10~~					+	+
* 001	01						*
100	10	+			+		
~~101~~	~~10~~					+	+
~~101~~	~~01~~			+			+
110	10	+	+				
110	01		+		+		

3 varianti
1: F, A, B
2: F, A, D
3: F, B, E

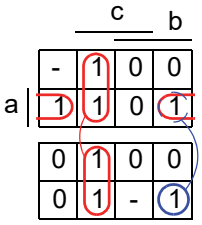
abc	e	A	B	C	D	E
100	10	+				+
110	10	+	+			
110	01		+		+	

Skeemi suurus ei sõltu ainult implikantide arvust (mõjutab loogikaelementide arvu), vaid ka loogikaelementide suurusest (sisentie arvust).

3 varianti
1: F, A, B
2: F, A, D
3: F, B, E

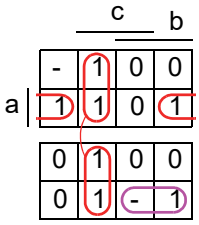
Lisaks üksikult minimeeritud
4: A, E, D, F

abc	xy
-01	11
1-0	10
110	01



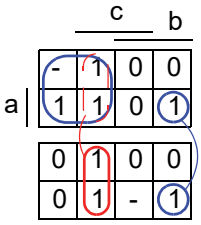
NOT - 2
2-NAND - 4
3-NAND - 1
26 transistori
[13 literaali]

abc	xy
-01	11
1-0	10
11-	01



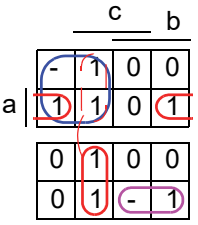
NOT - 2
2-NAND - 5
3-NAND - 0
24 transistori
[12 literaali]

abc	xy
-01	01
110	11
-0-	10



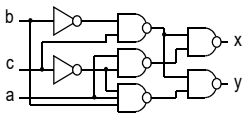
NOT - 2
2-NAND - 3
3-NAND - 1
22 transistori
[11 literaali]

abc	xy
1-0	10
-0-	10
11-	01
-01	01



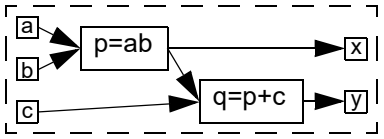
NOT - 2
2-NAND - 5
3-NAND - 0
24 transistori
[12 literaali]

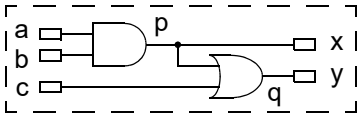
bi=b̄; ci=c̄; t1=bi·c; t2=a·ci; t3=a·b·ci; x=t1+t2; y=t1+t3;

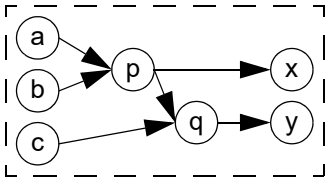


2. Mitmetasemeline loogikafunktsioonide minimeerimine

- Loogikaelementide teegid – loogikalülid / makrod
- Esitusviis – loogikavõrkgraaf (logic network) – omavahel ühendatud loogikafunktsioonid
- Seotud võrkgraaf (bound/mapped network) – omavahel ühendatud loogikalülid (struktuurne mudel)





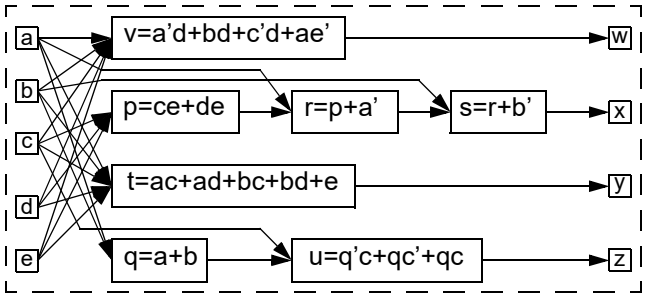


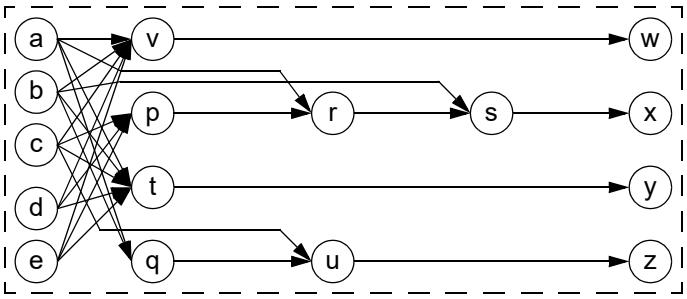
- Pindala (võimsustarbe) ennustuse minimeerimine – arvestada tuleb viite piiranguid
- Suurima viite minimeerimine – arvestada tuleb pindala (võimsustarbe) piiranguid
- Testitavuse parendamine / võimsustarbe vähendamine
- Ennustus (estimation) – literaalide arv, loogikalülide mudelid, olulised signaaliteed, ...
- Mitmetasemeline minimeerimine on *raske* !
- Heuristilised optimeerimis-strateegiad – samm-sammuline parendamine (teisendused võrkgraafil), erinevad meetodid – teisenduste variandid & teisenduste rakendamise järjekorrad.

Näitevõrkgraaf:

```
    p=ce+de
    q=a+b
    r=p+a'
    s=r+b'
    t=ac+ad+bc+bd+e
    u=q'c+qc'+qc
    v=a'd+bd+c'd+ae'
    w=v
    x=s
    y=t
    z=u

    w= a'd+bd+c'd+ae'
f:  x= a'+b'+ce+de
    y= ac+ad+bc+bd+e
    z= a+b+c
```



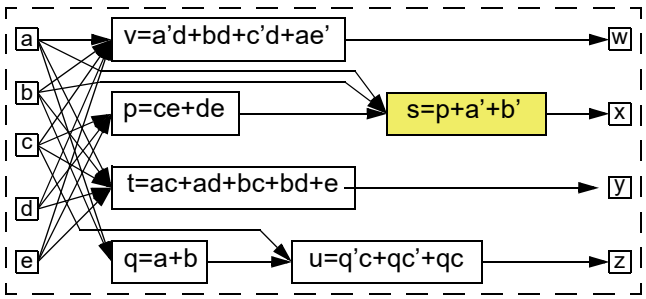


Eemaldamine (Elimination)
Eemaldatakse üks funktsioon, asendatakse vastavad muutujad.
Näide – eemaldatakse *r*

```
s=r+b';   r=p+a';
```

asendus *s*-s

```
s=p+a'+b';
```

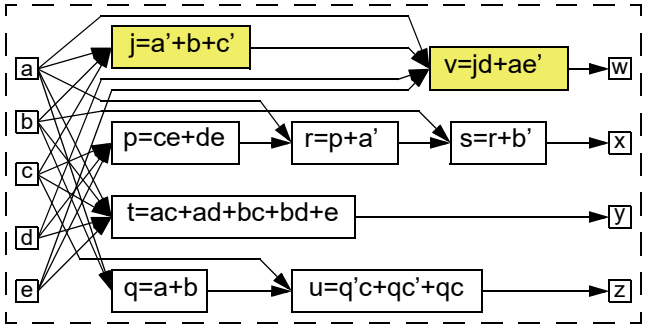


Dekompositsioon (Decomposition)
Üks funktsioon *jagatakse väik-semateks* funktsioonideks, luuakse uus sõlm (uued sõlmed).
Näide – *v* jagatakse kaheks

```
v=a'd+bd+c'd+ae';
```

luuakse *j*

```
j=a'+b+c';   v=jd+ae';
```



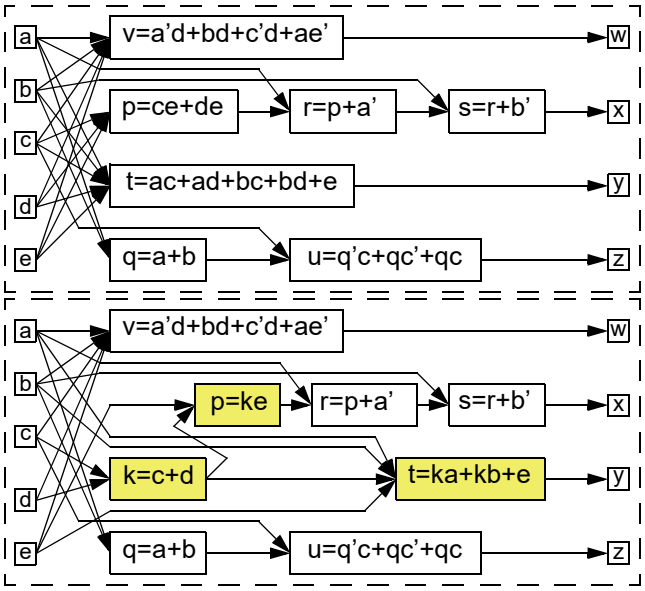
Eraldamine (Extraction)
Leitakse *ühine alam-avaldis* (common sub-expression) kahele (või enamale) funktsioonile. Alam-avaldis eraldatakse kui uus funktsioon. Luuakse uus sõlm.

Näide – *p* ja *t* jagavad alam-avaldist

```
p=ce+de;   t=ac+ad+bc+bd+e;
p=(c+d)e;  t=(c+d)(a+b)+e;
```

luuakse *k*

```
k=c+d;   p=ke;   t=ka+kb+e;
```



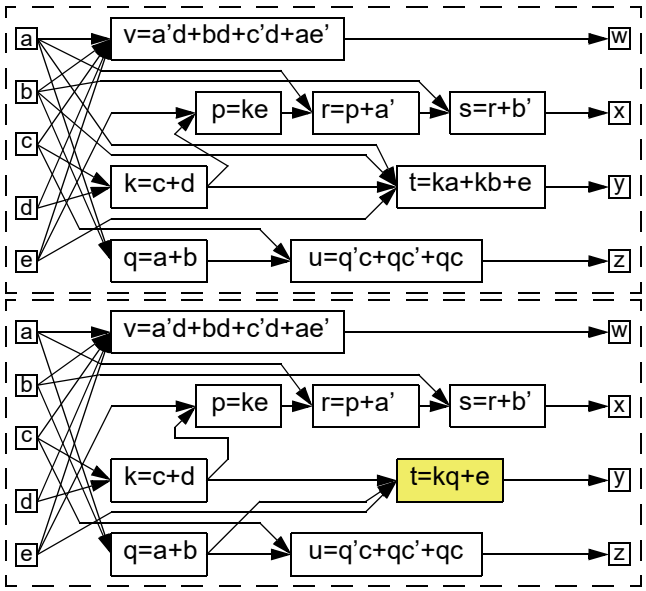
Asendamine (Substitution)
Lihtsustatakse osa-funktsiooni kasutades *lisa-sisendit*/*-muutujat*, mida selles varem funktsioonis ei esinenud.

Näide – *t* sisaldab *q* kui alam-avaldist

```
t=ka+kb+e;
q=a+b;
```

uus *t*

```
t=kq+e;
```



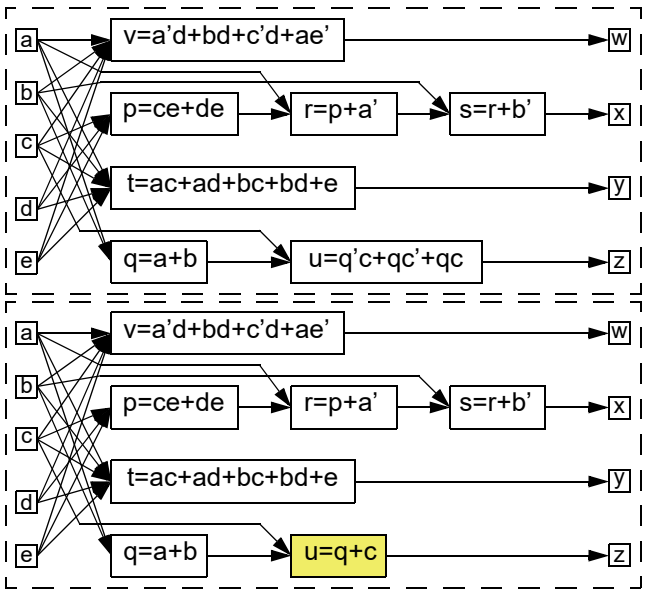
Lihtsustamine (Simplification)
Lihtsustatakse osa-funktsioon.

Näide – lihtsustatakse *u*

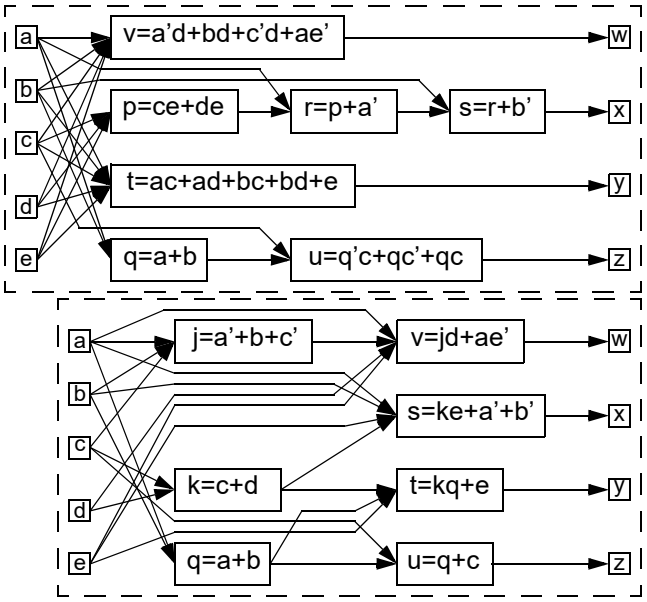
```
u=q'c+qc'+qc
```

uus *u*

```
u=q+c
```



Teisenduste jada – lõplik tulemus



Lõplik funktsioon

```
j=a'+b+c'
k=c+d
q=a+b
s=ke+a'+b'
t=q+c
u=q+c
v=jd+ae'
```

Funktsioone/loogikalülisid – 7 ja 7
Literaale – 33 ja 20
Viide – 3 ja 2 (sõlmi)
Viide – 9 ja 7 (sõlmi+literaale)

Kõik sammud

Lähteülesanne

```
p=ce+de;  q=a+b;  r=p+a';  s=r+b';  t=ac+ad+bc+bd+e;
u=q'c+qc'+qc;  v=a'd+bd+c'd+ae';  w=v;  x=s;  y=t;  z=u
```

1. Eemaldamine – s, r → s

```
p=ce+de;  q=a+b;  s=p+a'+b';  t=ac+ad+bc+bd+e;
u=q'c+qc'+qc;  v=a'd+bd+c'd+ae';  w=v;  x=s;  y=t;  z=u
```

2. Dekompositsioon – v → j, v

```
p=ce+de;  q=a+b;  s=p+a'+b';  t=ac+ad+bc+bd+e;
u=q'c+qc'+qc;  j=a'+b+c';  v=jd+ae';  w=v;  x=s;  y=t;  z=u
```

3. Eraldamine – p, t → k, p, t

```
j=a'+b+c';  k=c+d;  p=ke;  q=a+b;  s=p+a'+b';  t=ka+kb+e;
u=q'c+qc'+qc;  v=jd+ae';  w=v;  x=s;  y=t;  z=u
```

4. Asendamine – q, t → q, t

```
j=a'+b+c';  k=c+d;  p=ke;  q=a+b;  s=p+a'+b';  t=kq+e;
u=q'c+qc'+qc;  v=jd+ae';  w=v;  x=s;  y=t;  z=u
```

5. Lihtsustamine – u → u

```
j=a'+b+c';  k=c+d;  p=ke;  q=a+b;  s=p+a'+b';  t=kq+e;
u=q+c;  v=jd+ae';  w=v;  x=s;  y=t;  z=u
```

6. Eemaldamine – s, p → s

```
j=a'+b+c';  k=c+d;  q=a+b;  s=ke+a'+b';  t=kq+e;
u=q+c;  v=jd+ae';  w=v;  x=s;  y=t;  z=u
```

3. Optimeerimisviisid

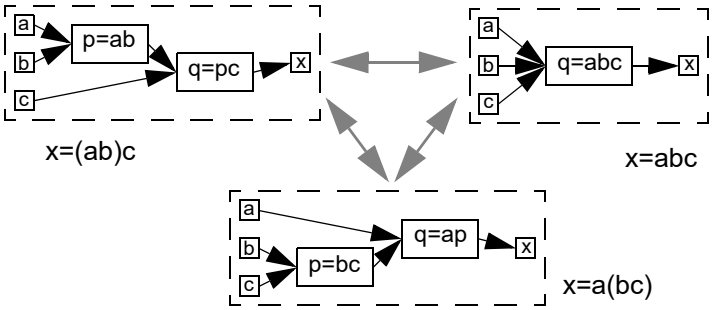
Algoritmiline – iga teisenduse tüübi jaoks tuleb määrata algoritm (*operaator*), kasutusel heuristilised meetodid (nõrk lokaalne optimeerimine), võimalik on ajutine (pinna/viite) ennustuse halvenemin. Operaatorite järjekord on skriptipõhine (puhtkogemuslik). Teisendused on nii algebralised kui kakahendmeetodid.

Reeglitel põhinev – andmebaas sisaldab mustripaaride hulka, mustrite asendused on defineeritud reeglite hulgaga.

Tehnoloogiast sõltuv optimeerimine (technology mapping) – elementide teegid, kus element realiseerib lihtsamat mõne sisendiga funktsiooni. Programmeeritav loogika – mõne sisendiga suvalised funktsioonid (nt. CLB).

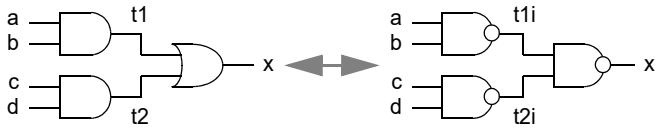
Reeglitel põhinev optimeerimine

Mustri (pattern) otsimine ja asendamine teisega, vajalik on kanoonilise esitusviis.
Primitiivsete operatsioonide võrkgraaf – mustrite keerukus pole piiratud.
Seos kasutatava tehnoloogiaga tihtipeale läbi abstraktsete tehnoloogiate.

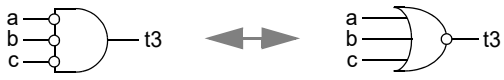


<u>Teisenduste näited:</u>

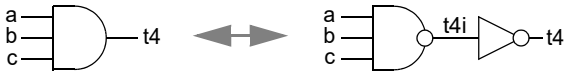
```
t1=a b; t2=c d; x=t1+t2;
t1'=(a b)'; t2'=(c d)';
   x=( t1' t2')';
t1i=(a b)'; t2i=(c d)';
   x=(t1i t2i)';
```




```
t3 = a' b' c';
t3 = ( a + b + c )';
```




```
t4 = a b c ;
t4i = ( a b c )';  t4 = t4i';
```

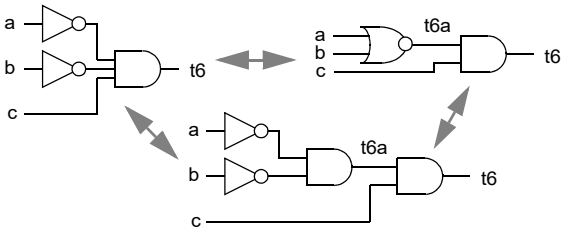


topelteitus

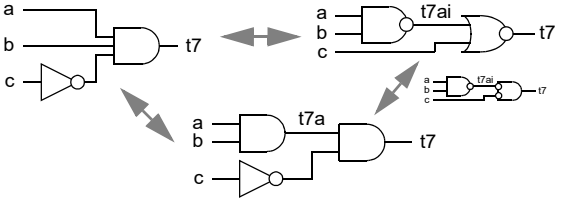
```
t5i = ( a b )';   t5 = t5';
t5 = a b;
```

De Morgani & topelteituse seadused

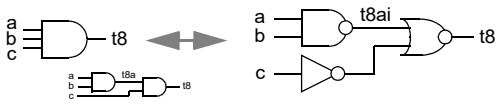
```
t6 = a' b' c;
t6a = a' b';   t6 = t6a c;
t6a = ( a + b )';   t6 = t6a c;
```




```
t7 = a b c';
t7a = a b;   t7 = t7a c';
[ t7ai= ( a b )';  t7 = t7ai' c'; ]
t7ai= ( a b )';  t7 = ( t7ai + c )';
```

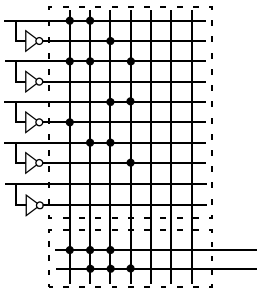



```
t8 = a b c;
[ t8a = a b;   t8 = t8a c; ]
t8ai= ( a b )';  t8 = (t8ai + c')';
```

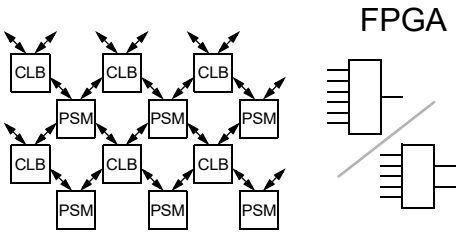


Erijuhud – programmeeritav loogika / PLD – Programmable Logic Device

PLA – Programmable Logic Array
- Programmeeritavad loogikamaatriksid (ja ühendused)
- Loogikamaatriks – mitme sisendiga ja väljundiga loogikafunktsioonide süsteem, implikantide arv piiratud
- Tükeldamine vastava suurusega funktsioonideks



FPGA – Field Programmable Gate Array
- Väliprogrammeeritav loogika (ka korduv-programmeeritav loogika)
- Programmeeritavad loogikaplokid (CLB)
  ... ja programmeeritavad ühendused (PSM)
- Xilinx – Spartan, Artix, Kintex & Virtex seeriad
  CLB – neli 6->1 / 5->2 funktsiooni
- Tükeldamine 5- või 6-sisendiga funktsioonideks



4. Funktsioonide teisendused – kahendmeetodid

Kasutatakse loogikafunktsioonide omadusi, on võimalik kasutada osaliselt määratust (don't-care). Aegajalt on siiski (liigagi) keeruline.

Kahendasendus (boolean substitution)

lähtefunktsioonid: `h=a+bcd+e; q=a+cd;` , tulemus: `h=a+bq+e` , sest

`a+bq+e=a+b(a+cd)+e=a+bcd+e  [a+b(a+cd)+e=a+ab+bcd+e=a+bcd+e]`

või hoopis?!

```
h=a+bcd+e; q=cd+e   ->   h=a+bq+e
a+bq+e=a+b(cd+e)+e=a+bcd+e   [a+b(cd+e)+e=a+bcd+be+e=a+bcd+e]
```

5. Funktsioonide teisendused – algebralise meetodid

Funktsioone vaadeldakse kui *polünoome*, kasutatakse polünoomide algebra omadusi. Lihtsam, kiirem, kui nõrgem optimeerimisvõime.

Algebraline asendus (algebraic substitution)

lähtefunktsioon: `t=ka+kb+e` , tulemus: `t=kq+e` , sest: `q=a+b` .

Algebralised meetodid – mõisted:
- *Polünoom* (polynomial) – korrutiste summa
- *Monoom* (monomial), üksliige – üksik korrutis e. kuup (e. implikant)

Kasutusel on ainult distributiivsuse seadus – a(b+c)=ab+ac, kuid a+bc≠(a+b)(a+c) .

Täiendid pole defineeritud, muutuja täiendit vaadeldakse kui lisamuutujat.

Määramatused pole defineeritud – operatsioonid ainult avaldistega, mille muutujate hulgad ei kattu; liiaste kuupide eemaldamine pole võimalik.

(a+b) ja (c+d) → (a+b)(c+d)=ac+ad+bc+bd OK!
(a+b) ja (a+c) → (a+b)(a+c)=aa+ac+ba+bc≠a+bc ei
(a+b) ja ($\bar{a}$+c) → (a+b)($\bar{a}$+c)=a$\bar{a}$+ac+b$\bar{a}$+bc≠ac+b$\bar{a}$ ei

Algebraline jagatis

On antud kaks lgebralist avaldist:

- *jagatav* (dividend), *jagaja* (divisor), *jagatis* (quotient), *jääk* (remainder)
- $f_{jagatis} = f_{jagatav} / f_{jagaja}$, kui
- $f_{jagatav} = f_{jagaja} \cdot f_{jagatis} + f_{jääk}$
- $f_{jagaja} \cdot f_{jagatis} \neq \varnothing$
- ning f_{jagaja} ja $f_{jagatis}$ [NB! Muutujate hulgad ei kattu ($sup(f_{jagaja}) \cap sup(f_{jagatis}) = \varnothing$)]

Näide: $f_{jagatav} = ac + ad + bc + bd + e$ & $f_{jagaja} = a + b$; $f_{jagatis} = c + d$ & $f_{jääk} = e$, sest $(a+b)(c+d)+e=f_{jagatav}$ & $\{a,b\} \cap \{c,d\} = \varnothing$.

Vrdl. mitte-algebraline jagatis – $f_i = a + bc$ & $f_j = a + b$; $(a+b)(a+c)=f_i$ kuid $\{a,b\} \cap \{a,c\} \neq \varnothing$.

Jagamisalgoritm

Jagatav: $A = \{ C_j^A, j=1,2,...,l\}$ – jagatava kuupide hulk.

Jagaja: $B = \{ C_i^B, i=1,2,...,n\}$ – jagaja kuupide hulk.

Jagatis Q ja jääk R on kuupide summad.

```
ALGEBRAIC_DIVISION (A,B) {
  for (i=1 to n) {
    D={Cj^A such that Cj^A⊇Ci^B};
    if (D==∅) return(∅,A);
    Di=D with var. in sup(Ci^B) dropped;
    if (i==1)   Q=Di;   else Q=Q∩Di;
  }
  R=A-Q×B;
  return(Q,R);
}
```

Näide #1:

$f_{jagatav} = ac + ad + bc + bd + e$ & $f_{jagaja} = a + b$

$A = \{ac, ad, bc, bd, e\}$ & $B = \{a, b\}$

- *i = 1*

C_1^B=a, $D=\{ac, ad\}$ & $D_1=\{c,d\}$, $Q = \{c,d\}$

- *i = 2*

C_2^B=b, $D=\{bc,bd\}$ & $D_2=\{c,d\}$, $Q = \{c,d\} \cap \{c,d\} = \{c,d\}$ -- *kuup vastab elemendile!*

Tulemus – $Q = \{c,d\}$ & $R = \{e\}$ – $f_{jagatis}=c+d$ & $f_{jääk}=e$

Näide #2:

$f_{jagatav} = axc + axd + bc + bxd + e$ & $f_{jagaja} = ax + b$

$A = \{axc, axd, bc, bxd, e\}$ & $B = \{ax, b\}$

- *i = 1*

C_1^B=ax, $D=\{axc, axd\}$ & $D_1=\{c,d\}$, $Q = \{c,d\}$

- *i = 2*

C_2^B=b, $D=\{bc,bxd\}$ & $D_2=\{c,xd\}$, $Q = \{c,d\} \cap \{c,xd\} = \{c\}$ -- *kuup vastab elemendile!*

Tulemus – $Q = \{c\}$ & $R = \{axd,bxd,e\}$ – $f_{jagatis}=c$ & $f_{jääk}=axd+bxd+e$

Jagamine – mis siis ikkagi toimub?

#1: A = ac + ad + bc + bd + e & B = a + b

(1) a (c+d) + bc + bd + e

(2) a (c+d) + b (c+d) + e → (c+d) "∩" (c+d) "=" (c+d)

(R) [ac+ad+bc+bd+e] "-" [(a+b) (c+d)] "=" [ac+ad+bc+bd+e] "-" [ac+ad+bc+bd] "=" [e]

ac+ad+bc+bd+e = a(c+d)+b(c+d)+e = (c+d)(a+b)+e

#2: A = axc + axd + bc + bxd + e & B = ax + b

(1) ax (c+d) + bc + bxd + e

(2) ax (c+d) + b (c+xd) + e → (c+d) "∩" (c+xd) "=" (c)

(R) [axc+axd+bc+bxd+e] "-" [(ax+b) (c)] "="
"=" [axc+axd+bc+bxd+e] "-" [axc+bc] "=" [axd+bxd+e]

axc+axd+bc+bxd+e= ax(c+d)+b(c+xd)+e = ax(c)+b(c)+axd+bxd+e = c(ax+b)+axd+bxd+e

Jagatise eksisteerimine?

Teoreem: Antud kaks algebralist avaldist f_i ja f_j ;

f_i / f_j on tühi, kui üks järgnevaist tingimustest on täidetud:

- f_j sisaldab muutujat, mida pole f_i-s;
  ab + cd / a + e –> a(b) + e(?) + cd
- f_j sisaldab kuupi, mille tugimuutujad ei sisaldu üheski f_i kuubi tugimuutujate hulgas
  ($\exists\ sup(C^j) \not\subset sup(C^i), \forall C^i \in f_i$);
  abc + def / ab + ad –> ab(c) + ad(?) + def
- f_j sisaldab rohkem liikemid kui f_i;
  ab + cd / a + b + c –> a(b) + b(?) + c(d)
- suvalist muutujat on f_j-s rohkem kui f_i-s
  abc + ade + bcd / ab + ad + ac –> ab(c) + ad(e) + ac(?)

Kasutusel kiireks kontrolliks – jagatist ei pruugi ikkagi eksisteerida – ac + be / a + b

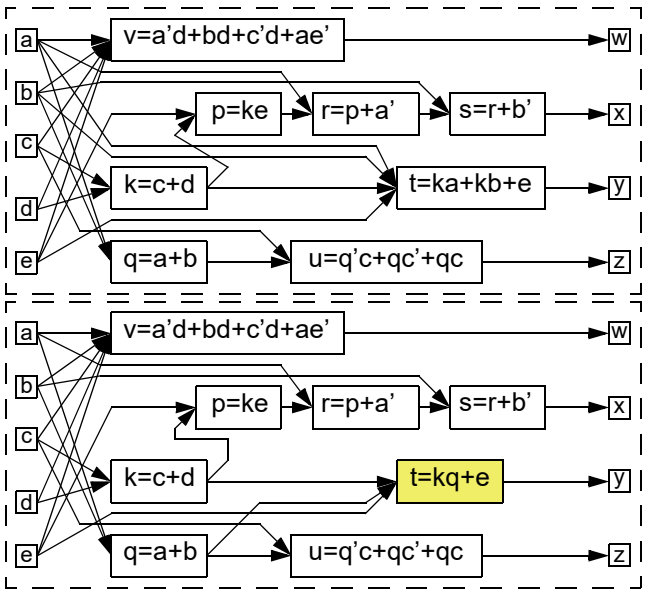
Kas on võimalik muutujate hulkade osaline kattumine?

ab + ac + bc / a + b = ? –> ab + ac + bc = ab + (a+b) c

Kasutamine – asendamine

- Vaadeldakse avaldiste paare
- Jagamine suvalises järjekorras
- Kui jagatis pole tühi, siis:
  ennustatakse pindala/viite muutust
  $f_{jagatav}$ asemele tuleb
  $f_{jagaja} \cdot f_{jagatis} + f_{jääk}$

f_t=ka+kb+e ($f_{jagatav}$)
f_q=a+b (f_{jagaja})
$f_{jagatis}$=k & $f_{jääk}$=e
f_t=kq+e



Kasutamine – <u>eraldamine</u>

- Ühiste alam-avaldiste otsimine – üksik kuup (monoom) / mitmik-kuup (*tuum* e. kernel)
- Sobivate *jagajate* leidmine
- *Kuubivaba* (cube-free) avaldis – pole võimalik faktoriseerida kuupi kasutades (st mitte ühtegi muutujat ei saa sulgude ette tuua)
- Avaldise *tuum* – avaldise kuubivaba jagatis, kui jagaja on kuup (*kaas-tuum* e. co-kernel))
- Avaldise *tuumade hulk* K(f) – kahe (või enama) avaldise ühine mitmik-kuup jagaja – tuumade hulkade ühisosa
- Tuumade leidmine
  naiivne – üritatakse leida jagatised muutujate kombinatsioonidele
  rekurssivne – tuumade tuumad on samuti tuumad

<u>Tuumade näide</u>

f_x = ace + bce + de + g

f_x / a → ce → ei ole kuubivaba
f_x / b → ce → ei ole kuubivaba
f_x / c → ae + be → ei ole kuubivaba
f_x / ce → a + b → kuubivaba → *tuum*
f_x / d → e → ei ole kuubivaba
f_x / e → ac+bc+d → kuubivaba → *tuum*
f_x / g → 1 → ei ole kuubivaba
f_x / 1 → ace+bce+de+g → kuubivaba → *tuum*

Tuumade hulk: K(fx) = { (a+b), (ac+bc+d), (ace+bce+de+g) }

<u>Eraldamise näide</u>

f_x = ace + bce + de + g
f_y = ad + bd + cde + ge
f_z = abc

K(f_x) = { (a+b), (ac+bc+d), (ace+bce+de+g) }
K(f_y) = { (a+b+ce), (cd+g), (ad+bd+cde+ge) }
K(f_z) = {}

f_w = a+b
f_x = wce + de + g
f_y = wd + cde + ge
f_z = abc

Kasutamine - <u>dekompositsioon</u>

Näide – väiksemateks funktsioonideks jagamine kahe sammuga:

1) f_x = ace + bce + de + g;
2) f_t = ac + bc + d; f_x = te + g;
3) f_s = a + b; f_t = sc + d; f_x = te + g;

Tuumadel põhinev dekompositsioon – avaldist jagatakse rekursiivselt:

1) $f_x = ace + bce + de + g$
   $K(f_x) = \{ (a+b), \underline{(ac+bc+d)}, (ace+bce+de+g) \}$
2) $f_t = ac + bc + d;$ $f_x = te + g;$
   $K(f_t) = \{ \underline{(a+b)}, (ac+bc+d) \}$
3) $f_s = a + b;$ $f_t = sc + d;$ $f_x = te + g;$