Espresso Explained

- How *espresso* works and what is behind this
  - *espresso* – two-level Boolean function minimizer
  - Center for Electronic Systems Design, UC Berkeley
    - https://ptolemy.berkeley.edu/projects/embedded/pubs/downloads/espresso/
    - Previously – UC Berkeley Design Technology Warehouse
    - Local web-page (slightly modified) – http://mini.pld.ttu.ee/~lrv/espresso/
- Exact and heuristic minimization of Boolean functions
- Multi-Valued Logic
- Data encoding and main operations
  - Giovanni De Micheli, Synthesis and Optimization of Digital Circuits. McGraw-Hill.

Set of Boolean Functions

- Black box model of a combinational circuit
- Defined based on Boolean algebra – (B,+,*,~), B={0,1}
- Logic functions (Boolean functions) can be
  - with multiple outputs (set of functions) – $f: B^n \rightarrow B$ and $f: B^n \rightarrow B^m$
  - partially defined (incomplete) – $f: B^n \rightarrow \{0,1,-\}^m$ (also $f{:}B^n\rightarrow\{0,1,*\}^m$)
    - depends how functions are used, e.g., impossible input combinations

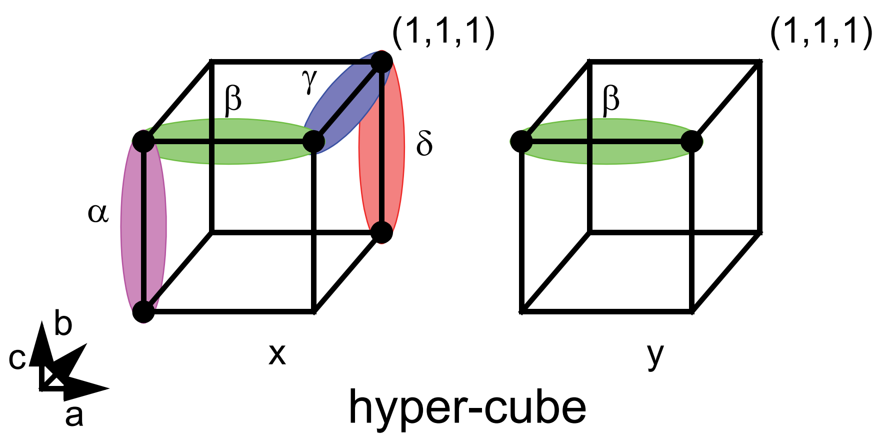
  - ON-set – F_f – the domain of a function where *f* is true
  - OFF-set – R_f – the domain of a function where *f* is false
  - DC-set – D_f – the domain of a function where *f* is undefined (don't-care)
  - Defined for every component in a set of functions

Definitions and Representations

- ***variable***
- ***literal*** **– variable and its inversion**
- ***product / cube / conjunction*** **– multiplication of literals**
- ***implicant / interval*** **– conjunction defining the value of a function (usually 1)**
  - ***hypercube***
- ***minterm*** **– implicant containing all input variables**
  - **a vertice (node) in hyper-cube**
- ***truth table***
  - **list of all minterms**
- ***implicant table / interval table / cover***
  - **set of implicants sufficient to define a function**

abc	xy
000	10
001	11
101	11
110	10
111	10

abc	xy
00-	10
-01	11
1-1	10
11-	10



hyper-cube

TTÜ1918

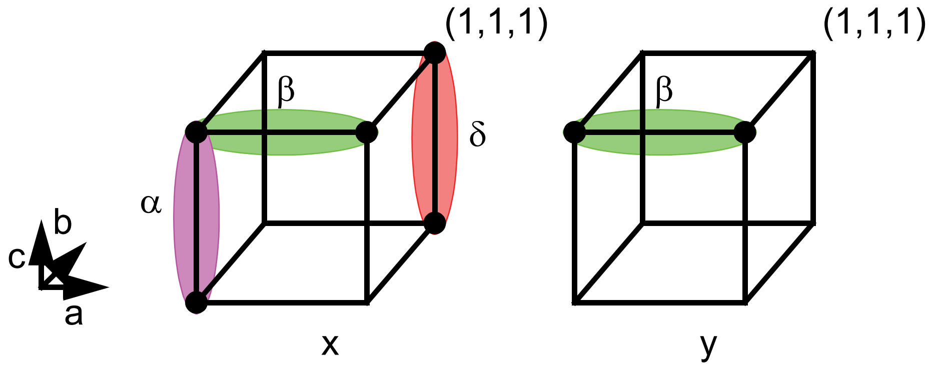
Definitions – Cover

- ***Minimum cover***
  - **a cover with the smallest number of implicants**
  - **global optimum**
- ***Minimal cover / Irredundant cover***
  - **a cover not contained in any other cover**
  - **it is not possible to remove any of the implicants**
  - **local optimum**
- ***Prime implicant* – not contained in any other implicant**
- ***Prime cover* – a cover of prime implicants**
- ***Essential prime implicant* – there is a minterm cover by only this implicant**

essential implicants

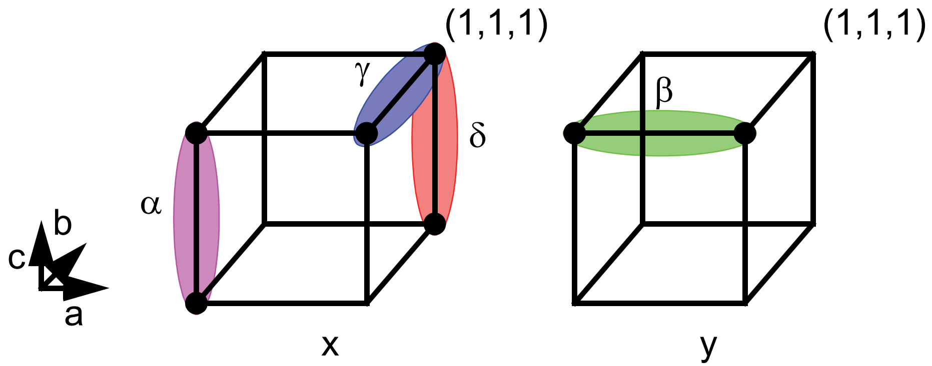
x – α & δ

y – β



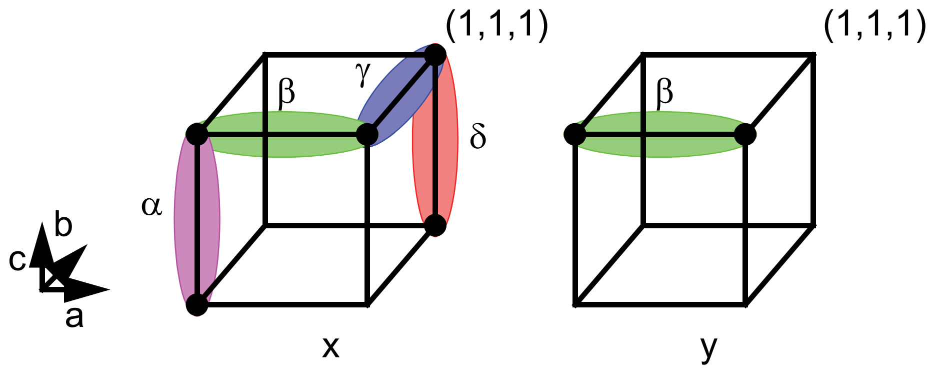
prime cover

minimum cover



prime cover

minimal cover



prime cover

Logic Function Minimization

- Exact methods (e.g. Quine-McCluskey method)
  - will find minimum cover(s)
  - often impossible for large functions
- Heuristic methods (MINI, PRESTO, ESPRESSO, ...)
  - will find minimal covers (can find minimum cover)
- Quine's theorem – minimum cover is prime cover
  - $\rightarrow$ prime implicants are enough to find minimum cover
- Quine-McCluskey method
  - main steps – [1] find all prime implicants, [2] find minimum cover
- Prime implicant table (chart)
  - exponential size! – up to 2^n minterms (can be grouped)
  - up to $3^n/n$ prime implicants (some functions have less)

Finding Prime Implicants – Quine method

$$f = \bar{a}\,\bar{d} + \bar{a}\,b + a\,\bar{b} + a\,\bar{c}\,d$$

$$f(a,b,c,d) = \Sigma(0,2,4,5,6,7,8,9,10,11,13)$$

- minterms are initial prime implicants
- implicants are combined pair wise
- covered (and duplicated) implicants are removed
- combining implicants and removing covered ones, this is continued until no more new implicants is generated

start

abcd
0000
0010
0100
0101
0110
0111
1000
1001
1010
1011
1101

1st phase

abcd	abcd
~~0000~~	-000
~~0010~~	0-10
~~0100~~	-010
~~0101~~	010-
~~0110~~	01-0
~~0111~~	100-
~~1000~~	10-0
~~1001~~	01-1
~~1010~~	-101
~~1011~~	011-
~~1101~~	10-1
00-0	1-01
0-00	101-

2nd phase

abcd	abcd
~~00-0~~	~~10-1~~
~~0-00~~	1-01
~~-000~~	~~101-~~
~~0-10~~	0--0
~~-010~~	~~0--0~~
~~010-~~	-0-0
~~01-0~~	~~-0-0~~
~~100-~~	01--
~~10-0~~	~~01--~~
~~01-1~~	10--
-101	~~10--~~
~~011-~~	

result

abcd
-101
1-01
0--0
-0-0
01--
10--

Finding Prime Implicants – Quine-McCluskey method

- Constraints when finding prime implicants
  - minterms are grouped by the number of 1-s
  - only implicants from neighboring groups can be combined
    - compare the number of 1-s!
  - implicants covering don’t-cares only have special notation (e.g. *)
    - the notation spreads when combining two implicants with the same notation
    - a hint – an implicant covering only don’t-cares is not needed in the implicant cover
  - irrelevant inputs can be taken into account when combining implicants
    - only implicants depending on the same variables can be combined

- Finding the minimal/minimum cover of prime implicants
  - simplifying the task by identifying the essential primes and removing from the table
  - identifying duplicated minterms (covered by the same primes) and removing them too

Finding Prime Implicants

$f = \bar{a}\,\bar{d} + \bar{a}\,b + a\,\bar{b} + a\,\bar{c}\,d$

$f(a,b,c,d) = \Sigma(0,2,4,5,6,7,8,9,10,11,13)$

minterms

gr.	abcd	
0	0000	*
1	0010	*
	0100	*
	1000	*
2	0101	*
	0110	*
	1001	*
	1010	*
3	0111	*
	1011	*
	1101	*

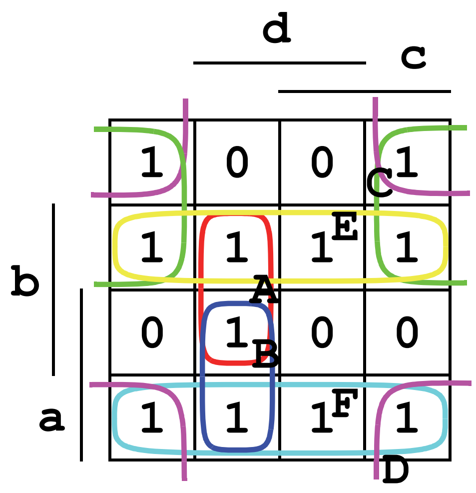
1st stage

gr.	abcd	
0	00-0	*
	0-00	*
	-000	*
1	0-10	*
	-010	*
	010-	*
	01-0	*
	100-	*
	10-0	*
2	01-1	*
	-101	A
	011-	*
	10-1	*
	1-01	B
	101-	*

2nd stage

gr.	abcd	
0	0--0	C
	-0-0	D
1	01--	E
	10--	F

* - is covered



Finding Prime Implicant Cover

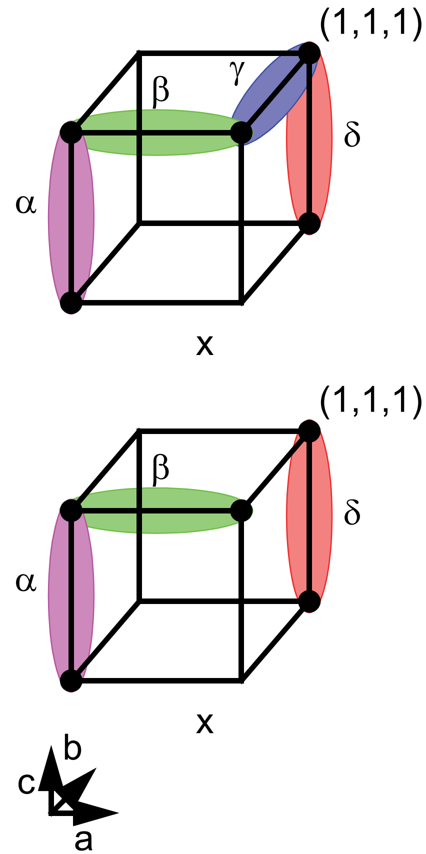
- $x = \bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,\bar{b}\,c + a\,\bar{b}\,c + a\,b\,\bar{c} + a\,b\,c$

- **Earlier methods to find cover**
  - **Petrick's method**
    - **implicants as product-of-sum (pos)**
    - **convert to sum-of-products (sop)**
    - **select the smallest product**
  - **pos -** $(\alpha)\,(\alpha+\beta)\,(\beta+\gamma)\,(\delta)\,(\gamma+\delta) = 1$
  - **sop -** $\alpha\beta\delta + \alpha\gamma\delta = 1$
  - **Solutions -** $\{\alpha,\beta,\delta\}$ **or** $\{\alpha,\gamma,\delta\}$

  - **Similar to SAT (satisfiability) task**

	abc x
α	00- 1
β	-01 1
γ	1-1 1
δ	11- 1

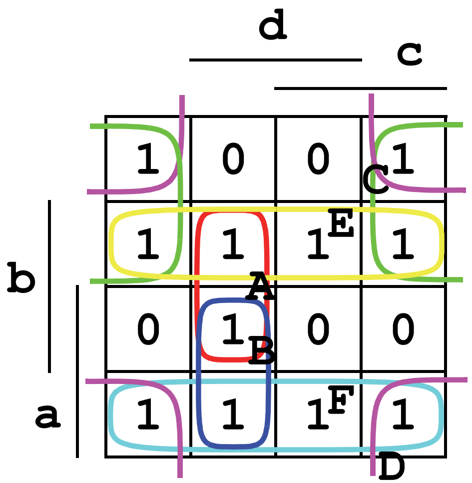
abc	α β γ δ
000	1 0 0 0
001	1 1 0 0
101	0 1 1 0
110	0 0 0 1
111	0 0 1 1



Example Task – Petrick's Method

prime implicants

abcd	A	B	C	D	E	F
0000	0	0	1	1	0	0
0010	0	0	1	1	0	0
0100	0	0	1	0	1	0
1000	0	0	0	1	0	1
0101	1	0	0	0	1	0
0110	0	0	1	0	1	0
1001	0	1	0	0	0	1
1010	0	0	0	1	0	1
0111	0	0	0	0	1	0
1011	0	0	0	0	0	1
1101	1	1	0	0	0	0



ACEF:

$$f = b\bar{c}d + \bar{a}\bar{d} + \bar{a}b + a\bar{b}$$

ADEF:

$$f = b\bar{c}d + \bar{b}\bar{d} + \bar{a}b + a\bar{b}$$

BCEF:

$$f = a\bar{c}d + \bar{a}\bar{d} + \bar{a}b + a\bar{b}$$

BDEF:

$$f = a\bar{c}d + \bar{b}\bar{d} + \bar{a}b + a\bar{b}$$

(C+D)(C+D)(C+E)(D+F)(A+E)(C+E)(B+F)(D+F)(E)(F)(A+B)=1

(C+D)(C+E)(D+F)(A+E)(B+F)(E)(F)(A+B)=1

(CC+CE+DC+DE)(DA+DE+FA+FE)(BE+FE)(FA+FB)=1

(C+DE)(AD+AF+DE+EF)(BE+EF)(AF+BF)=1

(CAD+CAF+CDE+CEF+DEAD+DEAF+DEDE+DEEF)(BEAF+BEBF+EFAF+EFBF)=1

(ACD+ACF+CEF+DE)(BEF+AEF)=1

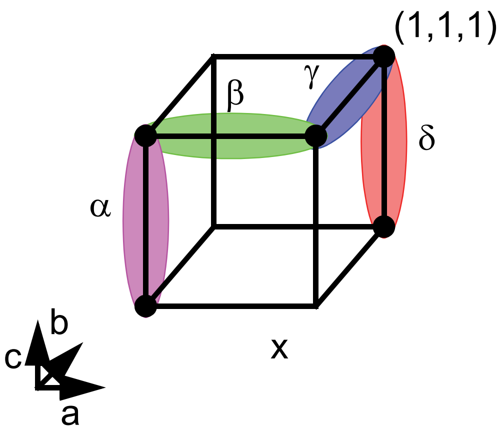
(ACDBEF+ACDAEF+ACFBEF+ACFAEF+CEFBEF+CEFAEF+DEBEF+DEAEF)=1

ACEF+ADEF+BCEF+BDEF=1

Example Task – Integer Linear Programming (ILP)

- **Optimization using matrices**
  - **Implicant table is binary matrix *A***
  - **Selected implicants are in binary vector *x***
  - **The task – find *x* that satisfies**
    - **$A\,x \geq 1$; select a sufficient number of columns to cover all lines**
  - **Minimize power of *x***

abc	α	β	γ	δ
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
110	0	0	0	1
111	0	0	1	1



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Example Task – Solving with Table

Implicant Table / Implicant Chart

prime implicants

	abcd	A	B	C	D	E	F
	0000			+	+		
	0010			+	+		
	~~0100~~			~~+~~		~~+~~	
	~~1000~~				~~+~~		~~+~~
	~~0101~~	~~+~~				~~+~~	
	~~0110~~			~~+~~		~~+~~	
	~~1001~~		~~+~~				~~+~~
	~~1010~~				~~+~~		~~+~~
*	0111					*	
*	1011						*
	1101	+	+				

E, F

* - essentials

abcd	A	B	C	D
0000			*	+
0010			*	+
1101	*	+		

A, C

		1	0	0	1
b		1	1	1	1
b	a	0	1	0	0
	a	1	1	1	1

$$f = b\,\bar{c}\,d + \bar{a}\,\bar{d} + \bar{a}\,b + a\,\bar{b}$$

A C E F

Heuristic Minimization

- Exact minimization is expensive
  - Finding all prime implicants takes memory and time
- Heuristic minimization
  - Avoids lacks of exact minimization(s)
  - Irredundant cover(s) with “reasonable” size(s)
  - Usable in many areas
  - Local (minimal) cover
    - initial cover is given
    - converting into prime cover
    - removing redundancy
  - Iterative improving
    - modifying implicants to improve size
    - extending/reducing is decided based on neighboring implicants

Heuristic Minimization – Main Operators

- **Expand**
  - **converting implicants into prime implicants**
  - **removing covered implicants**
- **Reduce**
  - **making implicants smaller while keeping cover correct**
- **Reshape**
  - **modifying a pair of implicants by increasing one and reducing another**
- **Irredundant**
  - **removing redundancy from the cover**

Example

task

0000	1
0010	1
0100	1
0101	1
0110	1
0111	1
1000	1
1001	1
1010	1
1011	1
1101	1

all prime implicants

0--0	a
-0-0	b
01--	c
10--	d
1-01	e
-101	f

1	0	0	1
1	1	1	1
0	1	0	0
1	1	1	1

possible solutions

{a,c,d,e}

{b,c,d,e}

Example – Expand

0000	*exp*
0010	x
0100	x
0101	
0110	x
0111	
1000	
1001	
1010	
1011	
1101	

0000
↓
0--0

0--0
covers
0010
0100
0110

0--0	a
0101	*exp*
0111	x
1000	
1001	
1010	
1011	
1101	

0101
↓
01--

01--
covers
[0100]
[0110]
0111

0--0	a
01--	c
1000	*exp*
1001	
1010	x
1011	
1101	

1	0	0	1
1	1	1	1
0	1	0	0
1	1	1	1

Example – Expand – All Steps

0000	exp
0010	x
0100	x
0101	
0110	x
0111	
1000	
1001	
1010	
1011	
1101	

0--0	a
0101	exp
0111	x
1000	
1001	
1010	
1011	
1101	

0--0	a
01--	c
1000	exp
1001	
1010	x
1011	
1101	

0--0	a
01--	c
-0-0	b
1001	exp
1011	x
1101	

0--0	a
01--	c
-0-0	b
10--	d
1101	exp

0--0	a
01--	c
-0-0	b
10--	d
1-01	e

{a,b,c,d,e}

1	0	0	1
1	1	1	1
0	1	0	0
1	1	1	1

Example – Reduce

0--0	xxxx
01--	c
-0-0	b
10--	d
1-01	e

0--0

↓

00-0

↓

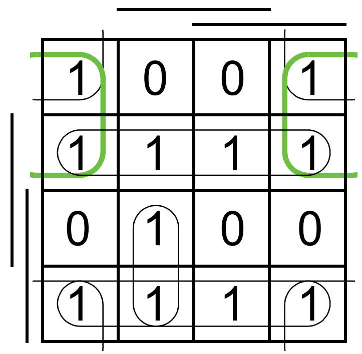
0000

-0-0

covers

~~0000~~

01--	c
-0-0	00-0
10--	d
1-01	e



Coverage analysis:
-0-0 & 0000 = 0000 - covers

For comparison:
-0-0 & 10-- = 10-0 - does not cover

Example – Reduce – All Steps

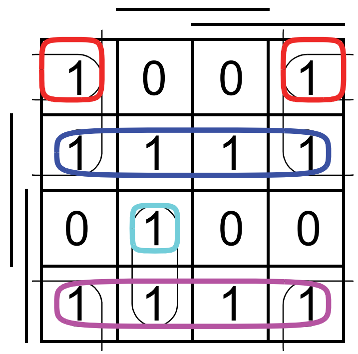
0--0	xxxx
01--	c
-0-0	b
10--	d
1-01	e

01--	c
-0-0	00-0
10--	d
1-01	e

01--	c
00-0	b′
10--	d
1-01	1101

01--	c
00-0	b′
10--	d
1101	e′

{ b’,c,d,e’ }

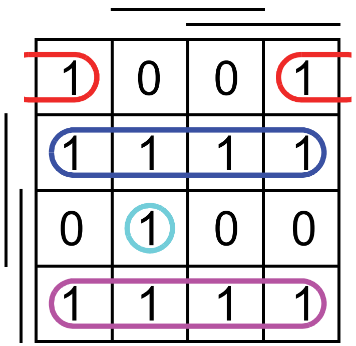


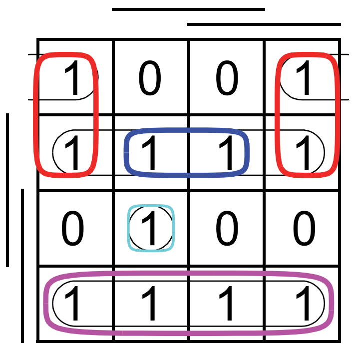
Example – Reshape

01--	01-1
00-0	0--0
10--	d
1101	e′

01-1	c′
0--0	a
10--	d
1101	e′

{ b’,c,d,e’ }





{ a,c’,d,e’ }

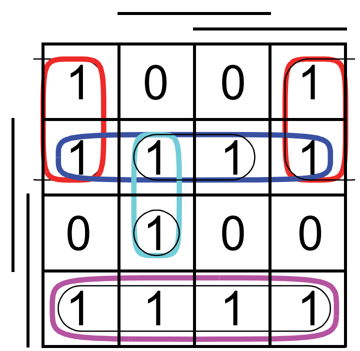
Example – Expand #2

01-1	*exp*
0--0	a
10--	d
1101	e′

01--	c
0--0	a
10--	d
1101	*exp*

01--	c
0--0	a
10--	d
-101	f

{ a,c,d,f }



Conclusion

- **MINI**
  - **Expand: cover – {a,b,c,d,e} – prime cover, redundant (no implicants inside another implicants)**
  - **Reduce: a is removed; b [-0-0] → b' [00-0] ; e [1-01] → e' [1101] ; cover – (b',c,d,e')**
  - **Reshape: {b',c} [00-0][01--] → {a,c'} [0--0][01-1]**
  - **Expand #2: kate – {a,c,d,f} ; prime cover, no redundancy**

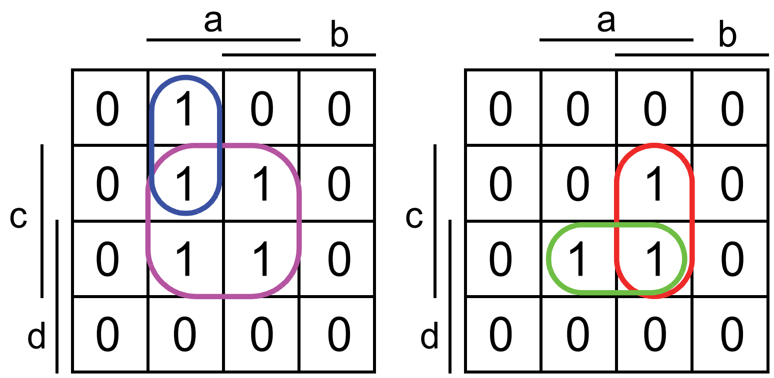
- **Intuitive expansion**
  - **in every implicant, replace '0' one '1' with '-' if possible**
  - **remove all covered implicants**
  - **problems – correctness check & order of implicants**

- **Correctness check**
  - **Espresso, MINI – *intersection* of expanded implicant against all 0-implicants (F_R) – must be empty, inversion/complement needed**
  - **Presto – checking whether the expanded implicant is still in the union of 1- & *-implicants ($F_F \cup F_D$), equal to recursive tautology check**

- **Expand – heuristic hints**
  - **Expanding first those intervals that are unlikely to be covered by the other implicants**
  - **Weighted intervals – larger weight hints to a smaller possibility to be covered (“sparsely populated neighborhood”)**
- **Reduce – heuristic hints**
  - **Weighted intervals – smaller weight hints to a larger possibility to be covered (“densely populated neighborhood”)**
- **Redundancy removal**
  - **Identifying essential intervals**
  - **Solving the cover problem heuristically**
- **Espresso**
  - **Find inversion**
  - **Identifying essential intervals/minterms (after expanding and redundancy removal)**
  - **Iteration – expand, irredundant, reduce**
  - **Weight functions – power of the cover & weighted sum of interval and literal count**

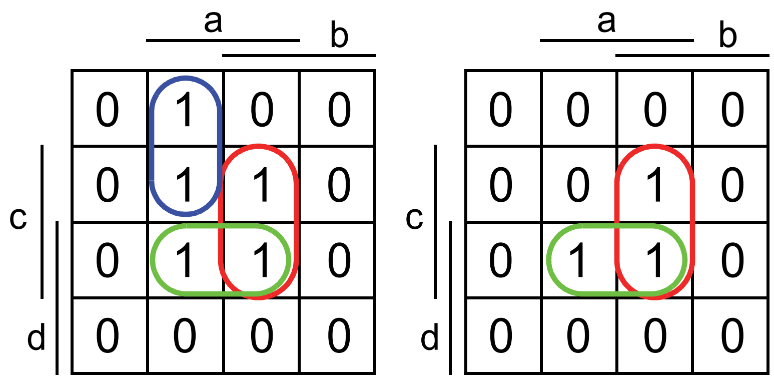
Minimizing Sets of Functions

abcd	xy
10-0	10
1-1-	10
1-11	01
111-	01



functions separately

abcd	xy
10-0	10
1-11	11
111-	11



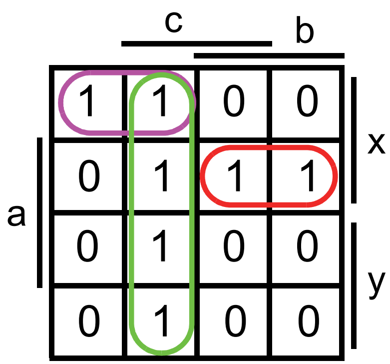
functions all together

Minimizing Sets of Functions

- Outputs are considered as an additional multi-valued input
- Same operations are used to find implicants

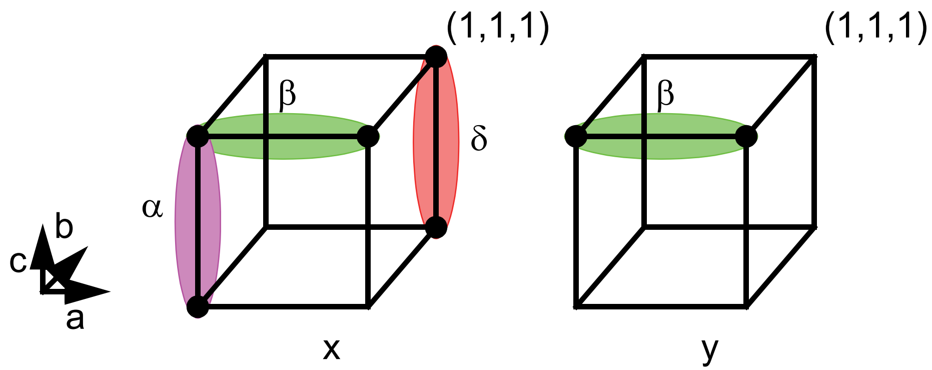
abc	xy
000	10
001	11
101	11
110	10
111	10

abcO	
0001	1
001-	1
101-	1
1101	1
1111	1



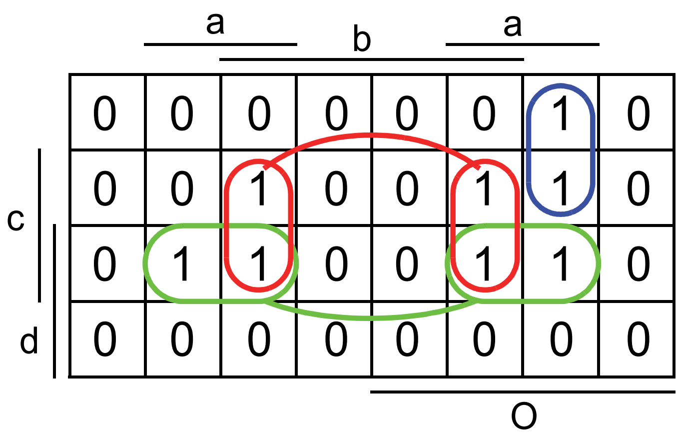
abcO	
00-1	1
-01-	1
11-1	1

abc	xy
00-	10
-01	11
11-	10



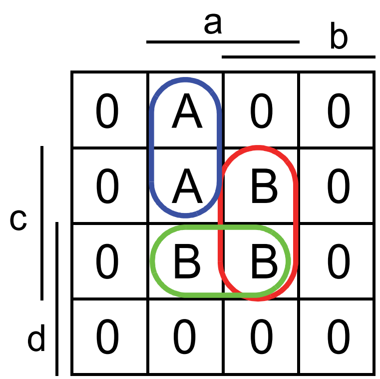
Minimizing Sets of Functions

abcdO	z
10-01	1
1-1-1	1
1-110	1
111-0	1



abcdO	z
10-01	1
1-11-	1
111--	1

with multi-valued logic



abcd	xy
10-0	10
1-11	11
111-	11

with symbolic encoding

Multiple-Valued Logic - MVL

- **Post algebra**
  - Generalization of Boolean algebra
  - Used as a mathematical base to design MVL logic gates
  - Emil Leon Post (1897-1954), 1921 – the first multi-valued logic
- **Binary logic – (B,*,+,~), B={0,1}**
  - fully determined functions – $f: B^n \rightarrow B$ and $f: B^n \rightarrow B^m$
  - incompletely determined functions – $f: B^n \rightarrow \{0,1,-\}^m$ (also $f{:}B^n{\rightarrow}\{0,1,*\}^m$)
  - AND, OR, NOT – full set of logic functions
- **MV-logic – ($\{P_i\}$,MIN,MAX,literal), $P_i=\{0,1,\ldots,m_i-1\}$**
  - incompletely determined functions – $f: P_1xP_2x\ldots xP_n \rightarrow P_m$
    or also $f: P_1xP_2x\ldots xP_n \rightarrow \{0,1,-\}^m$
  - MIN, MAX, literal - full set of MVL-functions

Multi-Valued Logic – Operations

- MIN(x, y) – minimal value of x and y [·]
  - cmp. AND in binary logic
- MAX(x, y) – maximal value of x and y [+]
  - cmp. OR in binary logic
- literal – unary operation – $x_i^{\{ci\}} = m_i\text{-}1$, when x_i=ci, else 0
  - notation – $x_1^{\{2\}} \equiv x_1^2$ and $x_1^{\{2\}} \equiv \overset{2}{x_1}$
- set literal – $x_i^{\{S\}} = m_i\text{-}1$, when $x_i \in S$, else 0
  - notation $x_3^{\{0,2\}} \equiv \overset{\{0,2\}}{x_3} \equiv \overset{0,2}{x_3}$
  - cmp. against binary logic – $x_i^{\{0\}} == \overline{x}_i$, $x_i^{\{1\}} == x_i$, $x_i^{\{0,1\}} == -$ (don't-care)
- Shannon's expansion (Boole's expansion)
  - $f()=\overline{x}f_{\overline{x}}()+xf_x()$ / $f()=x^0f_{x^0}()+x^1f_{x^1}()+\ldots+x^{m-1}f_{x^{m-1}}()$

Presentation Forms

Truth table

x_1	x_2	f	
0	0	0	$0 \cdot x_1^{\{0\}} \cdot x_2^{\{0\}}$
0	1	0	$0 \cdot x_1^{\{0\}} \cdot x_2^{\{1\}}$
0	2	2	$2 \cdot x_1^{\{0\}} \cdot x_2^{\{2\}}$
1	0	1	$1 \cdot x_1^{\{1\}} \cdot x_2^{\{0\}}$
1	1	1	$1 \cdot x_1^{\{1\}} \cdot x_2^{\{1\}}$
1	2	0	$0 \cdot x_1^{\{1\}} \cdot x_2^{\{2\}}$
2	0	0	$0 \cdot x_1^{\{2\}} \cdot x_2^{\{0\}}$
2	1	0	$0 \cdot x_1^{\{2\}} \cdot x_2^{\{1\}}$
2	2	2	$2 \cdot x_1^{\{2\}} \cdot x_2^{\{2\}}$

" · " – MIN "+" – MAX $x^{\{i\}}$ – x'i literal

Equation

$$f(x_1,x_2) = 1x_1^{\{1\}}x_2^{\{0\}} + 1x_1^{\{1\}}x_2^{\{1\}} + \\ + 2x_1^{\{0\}}x_2^{\{2\}} + 2x_1^{\{2\}}x_2^{\{2\}}$$

$$f(x_1,x_2) = 1x_1^{\{1\}}x_2^{\{0,1\}} + 2x_1^{\{0,2\}}x_2^{\{2\}}$$

Karnaugh map

x_2 \ x_1	0	1	2
0	0	1	0
1	0	1	0
2	2	0	2

x_2 \ x_1	0	1	2
0	0	1	0
1	0	1	0
2	2	0	2

Minimizing MV-Functions

- Nothing new!
- Given – cover of ones (f) and don't-cares (d) (and zeros (r)) of function *F*
- Find the minimal sum-of-products form of function *F*

- Generate *f+d* prime implicants
- Create table of implicants
- Solve cover problem
  - Algorithms differ in details only

Functions with Multiple Outputs

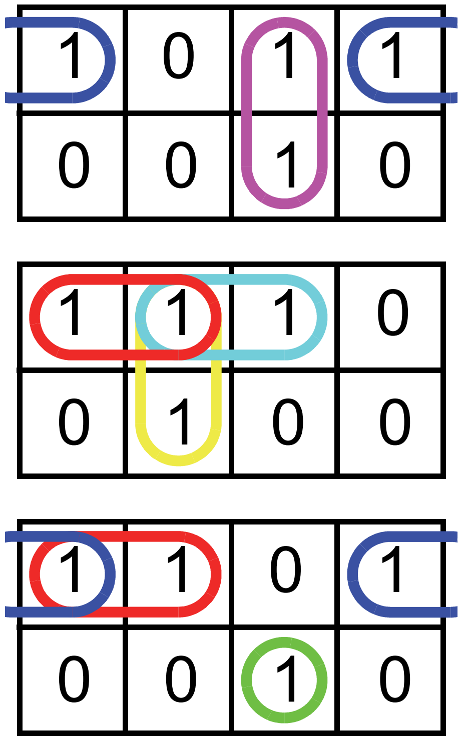
- Binary functions with *n* variables and *k* outputs is converted into a function with *n+1* inputs and 1 output; one of the input variables is MV:
  $\{0,1\}^n \rightarrow \{0,1\}^k \equiv \{0,1\}^n \times \{0, 1, ..., m\text{-}1\} \rightarrow \{0,1\}$
- Hong's theorem
  - every n-variable implicant plus corresponding outputs form an implicant in n+1-space
  - the number of outputs defines the number of values of the additional input
  - outputs defined by the implicant, form a set-literal in the additional input

Finding Prime Implicants

- Similar to finding implicants for a single function
  - difference in an exactly one input
    - compare against distributivity [(a+b)(a+c)==a(b+c)]
  - in practice, it makes sense to differ binary and MV inputs
- Difference in one binary input
  - exactly one binary input differs – 0 in one and 1 in another (MV-parts are identical)
  - combining: $a^0b^0c^0e^0 + a^0b^1c^0e^0 = a^0c^0e^0\ (b^0+b^1) = a^0b^{\{0,1\}}c^0e^0$
- Difference in MV input (~~outputs differ)
  - all binary inputs are identical
  - combining: $a^0b^1c^1e^0 + a^0b^1c^1e^1 = a^0b^1c^1\ (e^0+e^1) = a^0b^1c^1e^{\{0,1\}}$
- Bit vector representation – binary ‘0’, ‘1’ and ‘-’; positional encoding in MV part
  - $a^0b^0c^0e^0 + a^0b^1c^0e^0$: 000 100 + 010 100 => 0-0 100
  - $a^0b^1c^1e^0 + a^0b^1c^1e^1$: 011 100 + 011 010 => 011 110

Example

abc	xyz
000	111
001	011
010	101
011	110
100	000
101	010
110	000
111	101



abc	xyz
0-0	101
-11	100
00-	011
0-1	010
-01	010
111	001

Example (cont.)

abc	xyz
000	111
001	011
010	101
011	110
100	000
101	010
110	000
111	101

$$x(a,b,c)=\bar{a}\bar{b}\bar{c}+\bar{a}b\bar{c}+\bar{a}bc+abc$$

$$y(a,b,c)=\bar{a}\bar{b}\bar{c}+\bar{a}\bar{b}c+\bar{a}bc+a\bar{b}c$$

$$z(a,b,c)=\bar{a}\bar{b}\bar{c}+\bar{a}\bar{b}c+\bar{a}b\bar{c}+abc$$

$$f: \{0,1\}x\{0,1\}x\{0,1\}x\{0,1,2\} \text{ -> } \{0,1\}$$

$$o(a,b,c,e)=a^0b^0c^0e^0+a^0b^1c^0e^0+a^0b^1c^1e^0+ +a^1b^1c^1e^0+a^0b^0c^0e^1+a^0b^0c^1e^1+ +a^0b^1c^1e^1+a^1b^0c^1e^1+a^0b^0c^0e^2+ +a^0b^0c^1e^2+a^0b^1c^0e^2+a^1b^1c^1e^2$$

abc	e	o
000	100	1
010	100	1
011	100	1
111	100	1
000	010	1
001	010	1
011	010	1
101	010	1
000	001	1
001	001	1
010	001	1
111	001	1

Example (cont.)

abc	e	o	
000	100	1	*1*
010	100	1	*2*
011	100	1	*3*
111	100	1	*4*
000	010	1	*5*
001	010	1	*6*
011	010	1	*7*
101	010	1	*8*
000	001	1	*9*
001	001	1	*10*
010	001	1	*11*
111	001	1	*12*

$$\underset{3.}{a^0b^1c^1e^0} + \underset{7.}{a^0b^1c^1e^1} = a^0b^1c^1e^{\{0,1\}}$$

$$\underset{1.}{a^0b^0c^0e^0} + \underset{2.}{a^0b^1c^0e^0} + \underset{9.}{a^0b^0c^0e^2} + \underset{11.}{a^0b^1c^0e^2} = \ldots$$

$$\ldots = a^0b^{\{0,1\}}c^0e^0 + a^0b^{\{0,1\}}c^0e^2 = a^0b^{\{0,1\}}c^0e^{\{0,2\}} = a^0c^0e^{\{0,2\}}$$

$$\underset{5.}{a^0b^0c^0e^1} + \underset{9.}{a^0b^0c^0e^2} + \underset{6.}{a^0b^0c^1e^1} + \underset{10.}{a^0b^0c^1e^2} = \ldots$$

$$\ldots = a^0b^0c^0e^{\{1,2\}} + a^0b^0c^1e^{\{1,2\}} = a^0b^0c^{\{0,1\}}e^{\{1,2\}} = a^0b^0e^{\{1,2\}}$$

TTÜ1918

Example – Minimizing

abc	e	o	
000	100	1	*1*
010	100	1	*2*
011	100	1	*3*
111	100	1	*4*
000	010	1	*5*
001	010	1	*6*
011	010	1	*7*
101	010	1	*8*
000	001	1	*9*
001	001	1	*10*
010	001	1	*11*
111	001	1	*12*

1. *2.* *9.* *11.*

$$a^0b^0c^0e^0 + a^0b^1c^0e^0 + a^0b^0c^0e^2 + a^0b^1c^0e^2 = \ldots$$

$$\ldots = a^0b^{\{0,1\}}c^0e^0 + a^0b^{\{0,1\}}c^0e^2 = a^0b^{\{0,1\}}c^0e^{\{0,2\}} = \underline{a^0c^0e^{\{0,2\}}}$$

minterms

gr.	abc	e
0	000	100
	000	010
	000	001
1	010	100
	001	010
	001	001
	010	001
2	011	100
	011	010
	101	010
3	111	100
	111	001

1st phase

gr.	abc	e
0	000	110
	000	101
	000	011
	0-0	100
	00-	010
	00-	001
	0-0	001
1	010	101
	001	011
	. . .	

2nd phase

gr.	abc	e
0	000	111
	<u>0-0</u>	<u>101</u>
	. . .	

		c	c, b	b
	1	0	1	1
a	0	0	1	0

1	1	1	0
0	1	0	0

1	1	0	1
0	0	1	0

Example – Minimizing

abc	e	o	
000	100	1	*1*
010	100	1	*2*
011	100	1	*3*
111	100	1	*4*
000	010	1	*5*
001	010	1	*6*
011	010	1	*7*
101	010	1	*8*
000	001	1	*9*
001	001	1	*10*
010	001	1	*11*
111	001	1	*12*

minterms

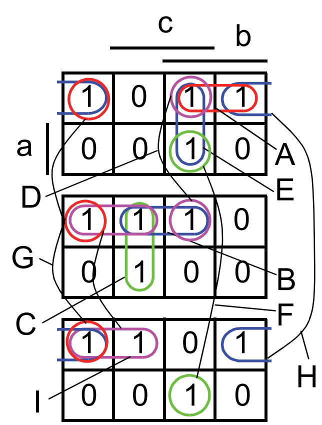
gr.	abc	e	
0	000	100	*
	000	010	*
	000	001	*
1	010	100	*
	001	010	*
	001	001	*
	010	001	*
2	011	100	*
	011	010	*
	101	010	*
3	111	100	*
	111	001	*

1st phase

gr.	abc	e	
0	000	110	*
	000	101	*
	000	011	*
	0-0	100	*
	00-	010	*
	00-	001	*
	0-0	001	*
1	010	101	*
	001	011	*
	01-	100	A
	0-1	010	B
	-01	010	C
2	011	110	D
	-11	100	E
3	111	101	F

2nd phase

gr.	abc	e	
0	000	111	G
	0-0	101	H
	00-	011	I



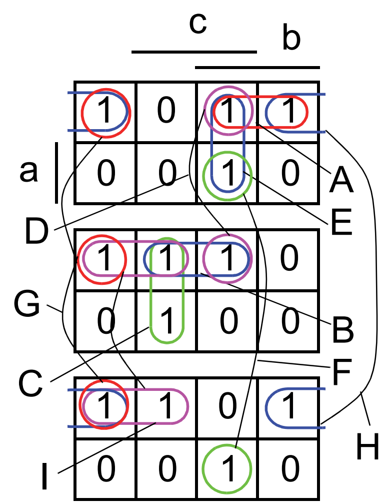
Example – Minimizing

prime implicant table

	abc	e	A	B	C	D	E	F	G	H	I
	~~000~~	~~100~~							+	+	
	~~000~~	~~010~~							+		+
	~~000~~	~~001~~							+	+	+
	~~010~~	~~100~~	+							+	
	~~001~~	~~010~~		+	+						+
*	001	001									*
*	010	001								*	
	011	100	+			+	+				
	011	010		+		+					
*	101	010			*						
	~~111~~	~~100~~					+	+			
*	111	001						*			

abc	e	A	B	D	E	G
011	100	+		*	+	
011	010		+	*		

abc	e	o	
011	110	1	D
111	101	1	F
0-0	101	1	H
00-	011	1	I
-01	010	1	C



TTÜ1918

Example (after minimization)

$$o(a,b,c,e)=a^0b^0c^0e^0+a^0b^1c^0e^0+a^0b^1c^1e^0+$$
$$+a^1b^1c^1e^0+a^0b^0c^0e^1+a^0b^0c^1e^1+$$
$$+a^0b^1c^1e^1+a^1b^0c^1e^1+a^0b^0c^0e^2+$$
$$+a^0b^0c^1e^2+a^0b^1c^0e^2+a^1b^1c^1e^2$$

Prime implicants & irredundant

$$o(a,b,c,e)=a^0b^1c^1e^{\{0,1\}}+a^1b^1c^1e^{\{0,2\}}+$$
$$+a^0b^{\{0,1\}}c^0e^{\{0,2\}}+a^0b^0c^{\{0,1\}}e^{\{1,2\}}+$$
$$+a^{\{0,1\}}b^0c^1e^1$$

$$o(a,b,c,e)=a^0b^1c^1e^{\{0,1\}}+a^1b^1c^1e^{\{0,2\}}+$$
$$+a^0c^0e^{\{0,2\}}+a^0b^0e^{\{1,2\}}+b^0c^1e^1$$

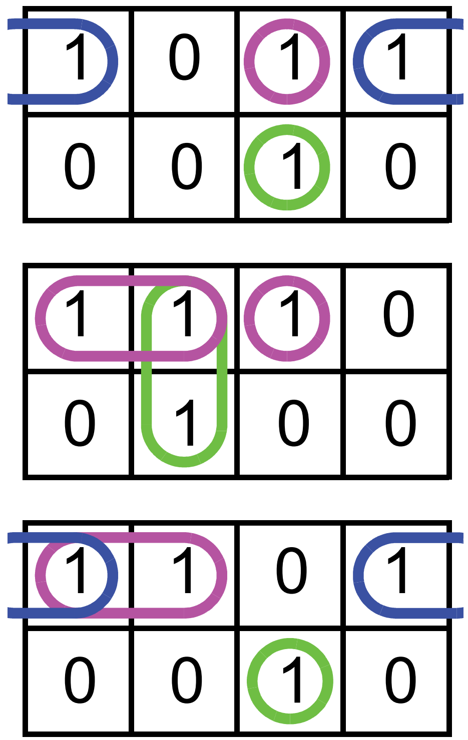
abc	e	o
011	110	1
111	101	1
0-0	101	1
00-	011	1
-01	010	1

Example (minimized binary functions)

$$o(a,b,c,e)=a^0b^1c^1e^{\{0,1\}}+a^1b^1c^1e^{\{0,2\}}+$$
$$+a^0c^0e^{\{0,2\}}+a^0b^0e^{\{1,2\}}+b^0c^1e^1$$

abc	e	o
011	110	1
111	101	1
0-0	101	1
00-	011	1
-01	010	1

abc	xyz
011	110
111	101
0-0	101
00-	011
-01	010



$$x(a,b,c)=\bar{a}bc+abc+\bar{a}\bar{c}$$
$$y(a,b,c)=\bar{a}bc+\bar{a}\bar{b}+\bar{b}c$$
$$z(a,b,c)=abc+\bar{a}\bar{c}+\bar{a}\bar{b}$$

Example #2

f_1: x_2 \ x_1	0	1
0	0	1
1	1	1

f_2: x_2 \ x_1	0	1
0	1	0
1	1	1

f_3: x_2 \ x_1	0	1
0	1	0
1	0	0

f_4: x_2 \ x_1	0	1
0	0	1
1	0	0

2 + 2 + 1 + 1 = 6 implicants

x_3 \ $x_1 x_2$	00	01	11	10	
0	0	1	1	1	f_1
1	1	1	1	0	f_2
2	1	0	0	0	f_3
3	0	0	0	1	f_4

3 implicants

Example #2 (minimized)

x_2x_1	$f_1f_2f_3f_4$
0 0	0 1 1 0
0 1	1 0 0 1
1 0	1 1 0 0
1 1	1 1 0 0

minterms

gr.	21	1234	
0	00	0100	*
	00	0010	*
1	01	1000	*
	01	0001	*
	10	1000	*
	10	0100	*
2	11	1000	*
	11	0100	*

1st phase

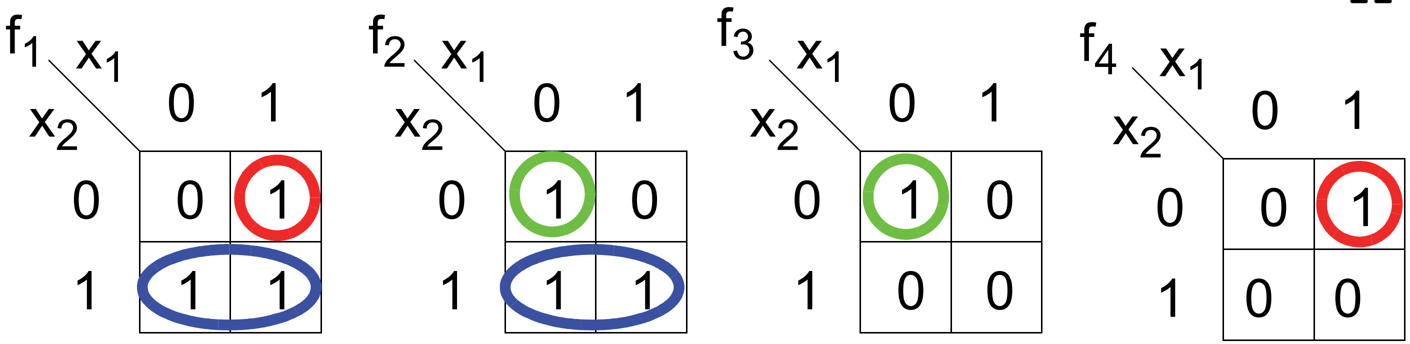
gr.	21	1234	
0	00	0110	A
	-0	0100	B
1	01	1001	C
	10	1100	*
	-1	1000	D
	1-	1000	*
	1-	0100	*
2	11	1100	*

2nd phase

gr.	21	1234	
1	1-	1100	E

prime implicant table

	21	1234	A	B	C	D	E
	~~00~~	~~0100~~	+	+			
*	00	0010	*				
	~~01~~	~~1000~~			+	+	
*	01	0001			*		
*	10	1000					*
	~~10~~	~~0100~~		+			+
	~~11~~	~~1000~~				+	+
*	11	0100					*



3 implicants

Example #3

abc	xy
000	10
001	11
101	11
110	10
111	10

abc e	
000 10	1
001 11	1
101 11	1
110 10	1
111 10	1

minterms

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

1st phase

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

2nd phase

gr.	abc e	
1	-01 11	D

1	1	0	0
0	1	1	1

0	1	0	0
0	1	0	0

prime implicant table

	abc e	A	B	C	D
*	000 10	*			
	~~001 10~~	+			+
*	001 01				*
	~~101 10~~		+		+
*	101 01				*
*	110 10			*	
	~~111 10~~		+	+	

abc	e	o	
00-	10	1	A
11-	10	1	C
-01	11	1	D

abc	xy
00-	10
11-	10
-01	11

Example #4 – Don't-Cares

abc	xyz
000	110
001	0-1
010	-0-
011	010
100	1-0
101	01-
110	101
111	-00

minterms

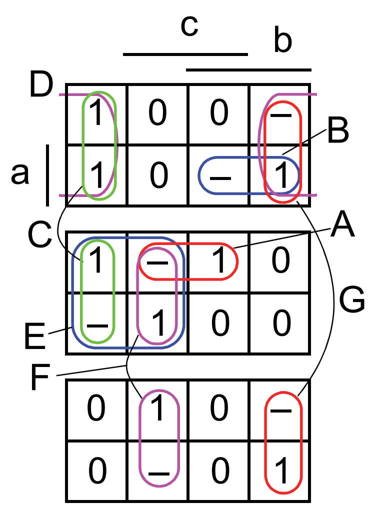
```
gr. abc  e
 0   000 100 *
     000 010 *
----------------
 1  *001 010 *
     001 001 *
    *010 100 *
    *010 001 *
     100 100 *
    *100 010 *
----------------
 2   011 010 *
     101 010 *
    *101 001 *
     110 100 *
     110 001 *
----------------
 3  *111 100 *
```

1st phase

```
gr.  abc  e
 0   000 110 *
     0-0 100 *
     -00 100 *
     00- 010 *
     -00 010 *
----------------
 1   001 011 *
    *010 101 *
     100 110 *
     0-1 010 A
     -01 010 *
     -01 001 *
     -10 100 *
     -10 001 *
     1-0 100 *
     10- 010 *
----------------
 2   101 011 *
     110 101 *
     11- 100 B
```

2nd phase

```
gr.  abc e
 0   -00 110 C
     --0 100 D
     -0- 010 E
----------------
 1   -01 011 F
     -10 101 G
```

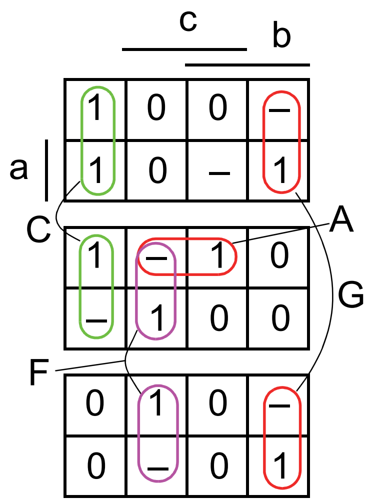


Example #4 – Don't-Cares

prime implicants & table

0-1	010	A
11-	100	B
-00	110	C
--0	100	D
-0-	010	E
-01	011	F
-10	101	G

abc	e	A	B	C	D	E	F	G
000	100			+	+			
000	010			+		+		
001	001						+	
011	010	+						
100	100			+	+			
101	010					+	+	
110	100		+		+			+
110	001							+



	abc	e	A	B	C	D	E	F	G
	000	100			+	+			
	000	010			+		+		
*	001	001						*	
*	011	010	*						
	100	100			+	+			
	~~101~~	~~010~~					~~+~~	~~+~~	
	~~110~~	~~100~~		~~+~~		~~+~~			~~+~~
*	110	001							*

abc	e	B	C	D	E
000	100		+	+	
000	010		+		+
100	100		+	+	

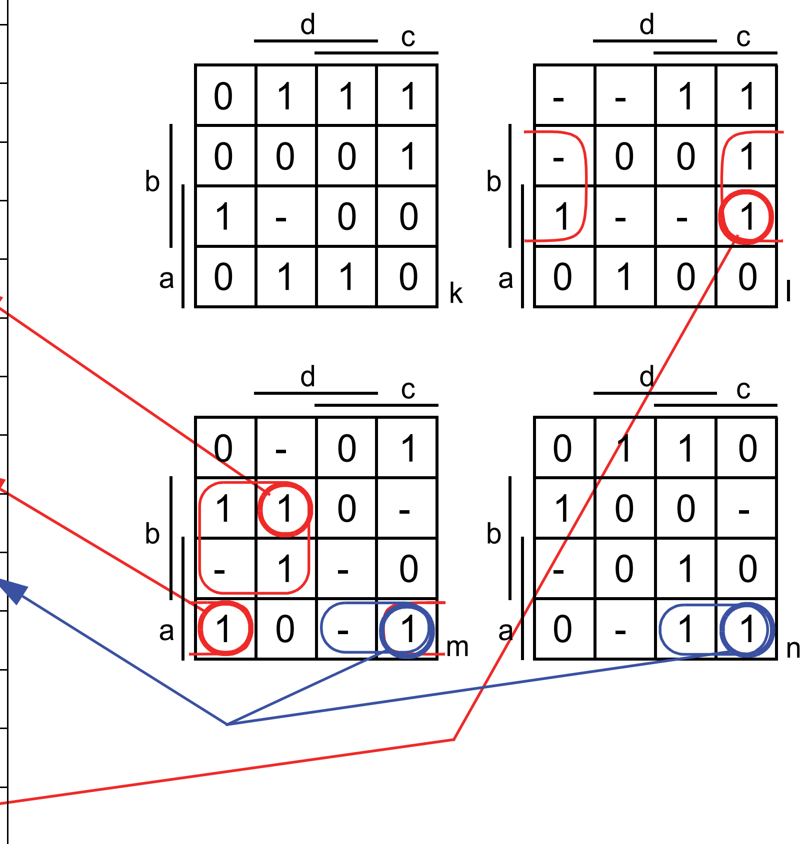
abc	xyz	
0-0	010	A
-00	110	C
-01	011	F
-10	101	G

TTÜ1918

Example #5 – Heuristic Minimization

abcd	klmn
0000	0-00
0001	1--1
0010	1110
0011	1101
0100	0-11
0101	0010
0110	11--
0111	0000
1000	0010
1001	110-
1010	0011
1011	10-1
1100	11--
1101	--10
1110	0100
1111	0--1

initial task...



How to start?

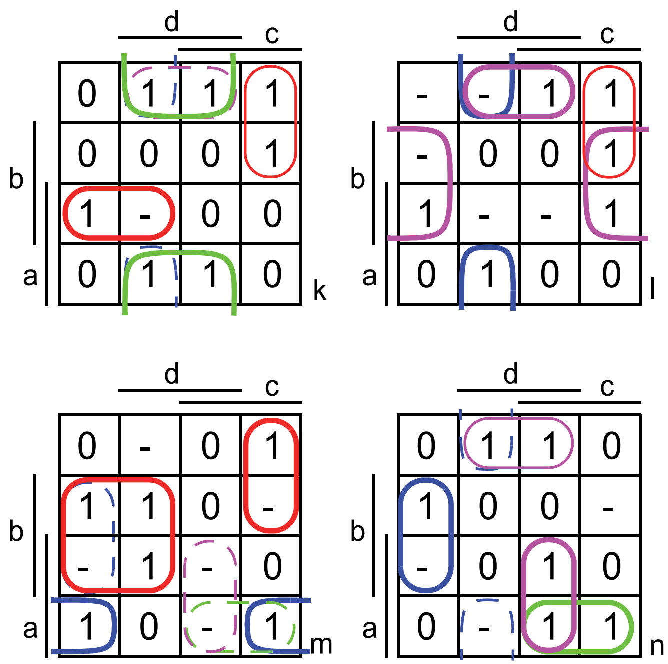
Cmp. essential prime implicants –> must be covered in any case –> “singles”, then “pairs”, ...

1) “singles” –
the largest implicant to cover as many 1-s as possible of the output

2) “pairs” –
the largest implicant to cover both outputs

Example #5 – Heuristic Minimization (cont.)

abcd	klmn
-10-	0010
10-0	0010
-1-0	0100
101-	0001
0-10	1110
-001	0100
00-1	0101
-0-1	1000
110-	1000
-100	0001
1-11	0001



Next the yet uncovered...

Starting again where few ones are left...

Remove those that are already covered (dashed line)

There are more versions...

Data Structures and Algorithms

- What is needed?

- Functions $f(x_1,x_2,\ldots,x_i,\ldots,x_n)$
  - f = ab + bc + ac
- Cofactors $f_{xi} – f(x_1,x_2,\ldots,1,\ldots,x_n)$ & $f_{\overline{xi}} – f(x_1,x_2,\ldots,0,\ldots,x_n)$
  - $f_a = b + c$ & $f_{\overline{a}} = bc$
- Shannon's expansion (Boole's expansion) – $f(x_1,x_2,\ldots,x_i,\ldots,x_n) = x_i \cdot f_{xi} + \overline{x}_i \cdot f_{\overline{xi}}$
  - $f = ab + bc + ac = a\, f_a + \overline{a}\, f_{\overline{a}} = a\,(b+c) + \overline{a}\,(bc)$
- Boole differential – $\partial f/\partial x_i = f_{xi} \oplus f_{\overline{xi}}$
  - shows how function depends on variable x_i – $\partial f/\partial a = f_a \oplus f_{\overline{a}} = (b+c) \oplus bc = \overline{b}c + b\overline{c}$

Matrix or Vector Representation

- Usually one line per implicant
- Different encodings exist
- Binary and multi-valued logic use the same idea – one position per symbol

x	00	illegal
0	10	"0"
1	01	"1"
-	11	don't-care

a b c d

01.11.10.11

$1\text{-}0\text{-} = a\bar{c}$

0.1.1.1

1.1.0.1

$1\text{-}0\text{-} = a\bar{c}$

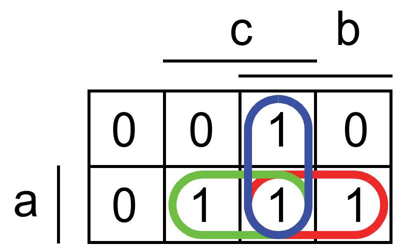
Encoding Possibilities

a b c d		and	(or)	legal?
$\boxed{01.11.10.11}$	$1\text{-}0\text{-} = a\bar{c}$	'&' of words	'|' of words	can not be'00'
$\boxed{10.00.01.00}$	$1\text{-}0\text{-} = a\bar{c}$	'|' of words	'&' of words	can not be '11'
$\boxed{0.1.1.1}$ $\boxed{1.1.0.1}$	$1\text{-}0\text{-} = a\bar{c}$	'&' of words	'|' of words	can not be $\begin{matrix}0\\0\end{matrix}$ ('|' with itself)

(and vice versa...)

- **Function –**
  f = ab + bc + ac

01 01 11 = ab
11 01 01 = bc
01 11 01 = ac



- **AND – intersection**

AND – (ab)(bc) = abc
01 01 01 = abc

($\bar{a}$c)(bc) = [10.11.01]&[11.01.01] =
= [10.01.01] = $\bar{a}$bc

($\bar{a}$c)(ab) = [10.11.01]&[01.01.11] =
= [00.01.01] = xxx

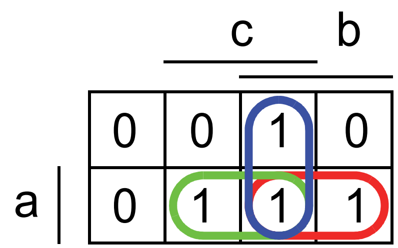
- **OR – union**

OR – 11 01 11 = b

$f = ab+bc+ac$

$f_a = b+c$

$f_{\bar{a}} = bc$



01 01 11 = ab
11 01 01 = bc
01 11 01 = ac

- **Cofactor**
  - **variable or its inversion**
  - **intersection**
  - **replace variable with don't-care**
  - **check coverage**

01 11 11 = a
01 01 11 = ab
01 01 01 = abc
01 11 01 = ac
11 01 11 = b
11 11 01 = c
~~11 01 01 = bc~~

10 11 11 = $\bar{a}$
$\underline{00}$ 01 11 = !!
10 01 01 = $\bar{a}bc$
$\underline{00}$ 11 01 = !!
11 01 01 = bc

Representing MV-Function / Set of Functions

- **Literal – set of legal values**
  - implicant – a line in the table
  - one bit per every value – indicates whether the value is presented or not

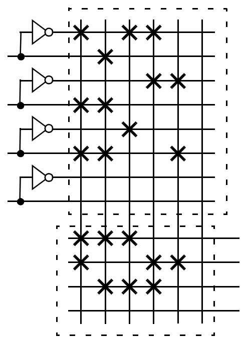
abc	xyz
011	110
111	101
0-0	101
00-	011
-01	010

abc	e	o
011	{1,2}	1
111	{1,3}	1
0-0	{1,3}	1
00-	{2,3}	1
-01	{2}	1

abc	e	o
011	110	1
111	101	1
0-0	101	1
00-	011	1
-01	010	1

```
10 01 01 110
01 01 01 101
10 11 10 101
10 10 11 011
11 10 01 010
```

  - two bits per (binary) input / one bit per output



- **Compare against PLA implementation!**

Exact Minimization – Example

minterms

gr.	abc	e	
0	000	100	*
	000	010	*
	000	001	*
1	010	100	*
	001	010	*
	001	001	*
	010	001	*
2	011	100	*
	011	010	*
	101	010	*
3	111	100	*
	111	001	*

1st phase

gr.	abc	e	
0	000	110	*
	000	101	*
	000	011	*
	0-0	100	*
	00-	010	*
	00-	001	*
	0-0	001	*
1	010	101	*
	001	011	*
	01-	100	A
	0-1	010	B
	-01	010	C
2	011	110	D
	-11	100	E
3	111	101	F

2nd phase

gr.	abc	e	
0	000	111	G
	0-0	101	H
	00-	011	I

gr.	a	b	c	e	
0	10	10	10	100	*
	10	10	10	010	*
	10	10	10	001	*
1	10	01	10	100	*
	10	10	01	010	*
	10	10	01	001	*
	10	01	10	001	*
2	10	01	01	100	*
	10	01	01	010	*
	01	10	01	010	*
3	01	01	01	100	*
	01	01	01	001	*

gr.	a	b	c	e	
0	10	10	10	110	*
	10	10	10	101	*
	10	10	10	011	*
	10	11	10	100	*
	10	10	11	010	*
	10	10	11	001	*
	10	11	10	001	*
1	10	01	10	101	*
	10	10	01	011	*
	10	01	11	100	A
	10	11	01	010	B
	11	10	01	010	C
2	10	01	01	110	D
	11	01	01	100	E
3	01	01	01	101	F

gr.	a	b	c	e	
0	10	10	10	111	G
	10	11	10	101	H
	10	10	11	011	I

Tautology

- Value of the function is always true
- Can be solved recursively
  - expansion by a variable
  - function is tautology when cofactors are tautologies
- Tautology:
  - one line (implicant) contains only ones (all don't-cares)
  - cover depends on one variable only and there is no column of 0-s
  - not tautology – there is a column of 0-s in the cover

$f = ab + ac + a\bar{b}\bar{c} + \bar{a}$

```
01 01 11
01 11 01
01 10 10
10 11 11
```

f_a

```
11 01 11
11 11 01
11 10 10
```

$f_{\bar{a}}$

```
11 11 11
```

↓

f_{ab}

```
11 11 11
~~11 11 01~~
~~11 00 10~~
```

Tautology -->

$f_{a\bar{b}}$

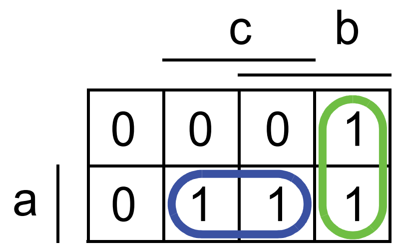
```
~~11 00 11~~
11 11 [01]
11 11 [10]
```

Covering

- **Implicant A is covered by F if and only if F_A is tautology**
  - **that is, cofactor of function F by A is tautology**

```
f = ac + bc̄
01 11 01 = ac
11 01 10 = bc̄

 ab covered?
```

```
f_ab
01 01 11
--------
11 11 01 = c
11 11 10 = c̄
tautology
```



Inversion / Complement

- Inversion – $\bar{f} = x \cdot \bar{f}_x + \bar{x} \cdot \bar{f}_{\bar{x}}$ ($f' = x \cdot f'_x + x' \cdot f'_{x'}$)
  - select variable
  - find cofactors
  - invert cofactors

- Simplifications
  - empty cover – inversion is universal cube (all don't-cares)
  - there is row of 1-s in the cover – inversion is empty (inversion of tautology)
  - cover has only one implicant – use De Morgan's law

```
01 01 11 = ab
11 01 01 = bc
01 11 01 = ac
```

		c	c, b	b
	0	0	1	0
a	0	1	1	1

f

		c	c, b	b
	1	1	0	1
a	1	0	0	0

$\bar{f}$

- Inversion – $\overline{f} = x \cdot \overline{f}_x + \overline{x} \cdot \overline{f}_{\overline{x}}$

- select variable – a
- find cofactors – f_a, $f_{\overline{a}}$
- $\overline{f} = a\,\overline{f}_a + \overline{a}\,\overline{f}_{\overline{a}}$

- $f_a = b + c \qquad f_{\overline{a}} = bc$

- invert cofactors
  - $\overline{f}_a = b\,\overline{f}_{ab} + \overline{b}\,\overline{f}_{a\overline{b}}$
  - $f_{ab} = 1 \quad \Rightarrow \quad \overline{f}_{ab} = 0$
  - $f_{a\overline{b}} = c \quad \Rightarrow \quad \overline{f}_{a\overline{b}} = \overline{c}$ (De Morgan)

- $\overline{f}_a = b\,0 + \overline{b}\,\overline{c} = \overline{b}\,\overline{c}$

```
01 01 11 = ab
11 01 01 = bc
01 11 01 = ac
```

---

f_a	$f_{\overline{a}}$
11 01 11	11 01 01
11 11 01	

---

f_{ab}	$f_{a\overline{b}}$
11 11 11	11 11 01

$\overline{f}_{ab}$	$\overline{f}_{a\overline{b}}$
00 00 00	11 11 10

$b\ \overline{f}_{ab}$	$\overline{b}\ \overline{f}_{a\overline{b}}$
00 00 00	11 10 10

$\overline{f}_a$

11 10 10

- **invert cofactors (cont.)**
  - $f_{\bar{a}} = b\,c$
  - $\overline{f_{\bar{a}}}$ = (De Morgan) = $\bar{b} + \bar{c}$
- $\bar{f} = a\,\overline{f_a} + \bar{a}\,\overline{f_{\bar{a}}}$
- $\bar{f} = a\,\bar{b}\,\bar{c} + \bar{a}\,\bar{b} + \bar{a}\,\bar{c}$

$f_{\bar{a}}$
```
11 01 01
```

$\overline{f_{\bar{a}}}$
```
11 10 11
11 11 10
```

$a\,\overline{f_a}$
```
01 10 10
```

$\bar{a}\,\overline{f_{\bar{a}}}$
```
10 10 11
10 11 10
```

$\bar{f}$
```
01 10 10
10 10 11
10 11 10
```

		c	c	
			b	b
	1	1	0	1
a	1	0	0	0

TTÜ1918

Espresso

- **Expand – heuristic hints**
  - **Expanding first those intervals that are unlikely to be covered by the other implicants**
  - **Weighted intervals – larger weight hints to a smaller possibility to be covered (“sparsely populated neighborhood”)**
  - **Calculating wights**
    - **find vector counting 1-s in columns**
    - **wight of an implicant – sum of multiplications of vector and positions**

$$f = ab + ac + a\bar{b}\bar{c} + \bar{a}$$

```
01 01 11        vector - [133333]
01 11 01
01 10 10        weights - (12,12,9,13)
10 11 11
```

- **Reduce – heuristic hints**
  - **Weighted intervals – smaller weight hints to a larger possibility to be covered (“densely populated neighborhood”)**

- **Redundancy removal**
  - **Identifying essential intervals**
  - **Solving the cover problem heuristically**

  - **Set of relatively essential implicants E^r**
    - **implicants covering minterms not covered by the other implicants**
  - **Set of fully redundant implicants R^t**
    - **implicants covered by the relatively essential implicants**
  - **Set of partially redundant implicants R^p**
    - **the rest of the implicants**

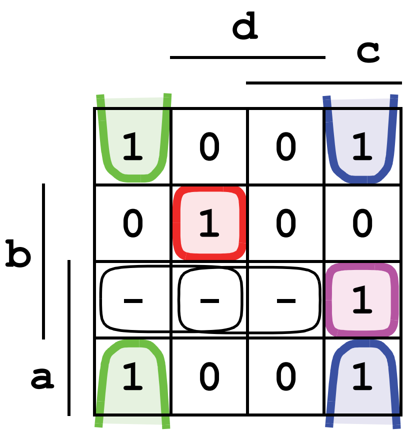
- **Espresso-Exact (exact minimizer)**
  - **Finding the exact cover using branch and bound method**
  - **Compact implicant table**
    - **minterms covered by the same implicants are seen as a single minterm**

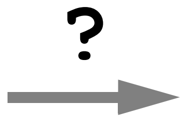
TTÜ1918

```
espresso ( F, D ) {
    R = complement ( F ∪ D );
    F = expand ( F, R );
    F = irredundant ( F, D );
    E = essentials ( F, D );
    F = F - E;
    D = D ∪ E;
    repeat {
        f2 = cost(F);
        repeat {
            f1 = |F|;
            F = reduce ( F, D );
            F = expand ( F, R );
            F = irredundant ( F, D );
        } until ( |F| ≥ f1 );
    } until ( cost(F) ≥ f2 );
    F = F ∪ E;
    D = D - E;
    F = make_sparse ( F, D, R );
}
```

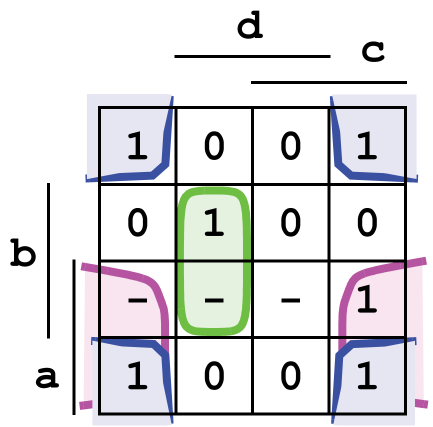
Example

```
task
 .i 4
 .o 1
 -000 1
 -010 1
 0101 1
 1110 1
 110- -
 11-1 -
 .e
```






```
result
 .i 4
 .o 1
 1--0 1
 -101 1
 -0-0 1
 .e
```




```
F:     D:    | R:    (espr)
-000   110-  | 011-   -0-1
-010   11-1  | 01-0   011-
0101         | 0-11   01-0
1110         | -0-1
             | -011
```

```
inverting             DeMorgan
-000: 11 10 10 10 (inv) 11 01 11 11 + 11 11 01 11 + 11 11 11 01
-010: 11 10 01 10 (inv) 11 01 11 11 + 11 11 10 11 + 11 11 11 01

&: 11 01 11 11 & 11 01 11 11 -> 11 01 11 11
   11 01 11 11 & 11 11 10 11 -> 11 01 10 11 [covered, st.A&B=A]
   11 01 11 11 & 11 11 11 01 -> 11 01 11 01 [covered]
   ...
   11 11 11 01 & 11 11 11 01 -> 11 11 11 01

-1--
---1

etc. etc.
```

Example (cont.)

- **Expand: starting from the smallest** (espresso starts from the largest)

```
-000: 11 10 10 10 =13
-010: 11 10 01 10 =13
0101: 10 01 10 01 =8 *
1110: 01 01 01 10 =10
      33 22 22 31
```

- **Removing a variable -> can not cover with any implicant from R!**
  - **few examples only, must be checked against <u>all</u> implicants**

```
d -> 010-: 10 01 10 11 -> & 01-0: 10 01 11 10 = 0100: 10 01 10 10 !!
c -> 01-1: 10 01 11 01 -> & 0-11: 10 11 01 01 = 0111: 10 01 01 01 !!
b -> 0-01: 10 11 10 01 -> & -0-1: 11 10 11 01 = 0001: 10 10 10 01 !!
a -> -101: 11 01 10 01 -> & -011: 11 10 01 01 = -xx1: 11 00 00 01 OK
```

- **Next expand**

```
-000: 11 10 10 10 =14
-010: 11 10 01 10 =14
-101: 11 01 10 01 =12
1110: 01 01 01 10 =11*
      34 22 22 31
```

Example (cont.)

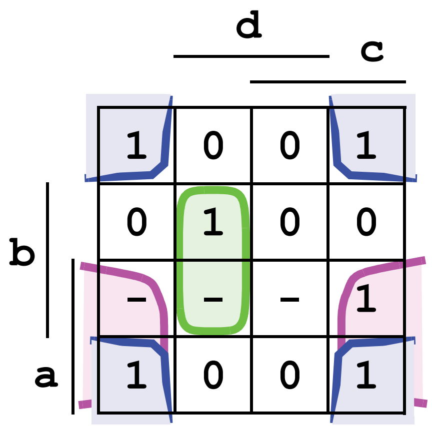
```
b -> 1-10: 01 11 01 10 -> & -0-1: 11 10 11 01 = 101x: 01 10 01 00 OK
c -> 1--0: 01 11 11 10 -> & 01-0: 10 01 11 10 = x1-0: 00 10 11 10 OK

-000: 11 10 10 10 =16
-010: 11 10 01 10 =15*
-101: 11 01 10 01 =14 (can not be extended more)
1--0: 01 11 11 10 =17
      34 32 32 31

c -> -0-0: 11 10 11 10 -> & 011-: 10 01 01 11 = 0x10: 10 00 01 10 OK
```

- **Check coverage: A & B == A**

```
-0-0: 11 10 11 10 & -000: 11 10 10 10 = -0-0: 11 10 11 10  [remove B]
```

- **Result:**

```
-0-0: 11 10 01 10
-101: 11 01 10 01
1--0: 01 11 11 10
```



Defining Output Polarity

- Single function
  - separate minimal cover by 1-s and 0-s
- Impractical with m outputs – there are 2^m combinations
- Heuristic approach (Sasao 1984)
  - new function with 2m outputs – extra m outputs are inverted
  - heuristic or exact minimization
  - new cover – direct and inverted outputs
  - solving covering ~ Petrik's method

	f_1	f_2	f_3	$\overline{f}_1$	$\overline{f}_2$	$\overline{f}_3$
a	1	0	0	0	0	0
b	0	1	0	1	0	0
c	0	1	0	0	0	0
d	1	0	0	0	1	0
e	0	0	1	0	0	0
f	0	0	0	1	0	0
g	0	0	0	0	1	0
h	0	0	0	0	0	1

(ad+bf)(bc+dg)(e+h)=1

abcde+abcdh+adeg+adgh+
+bcef+bcfh+bdefg+bdfgh=1

a d e g -> f_1, $\overline{f}_2$, f_3

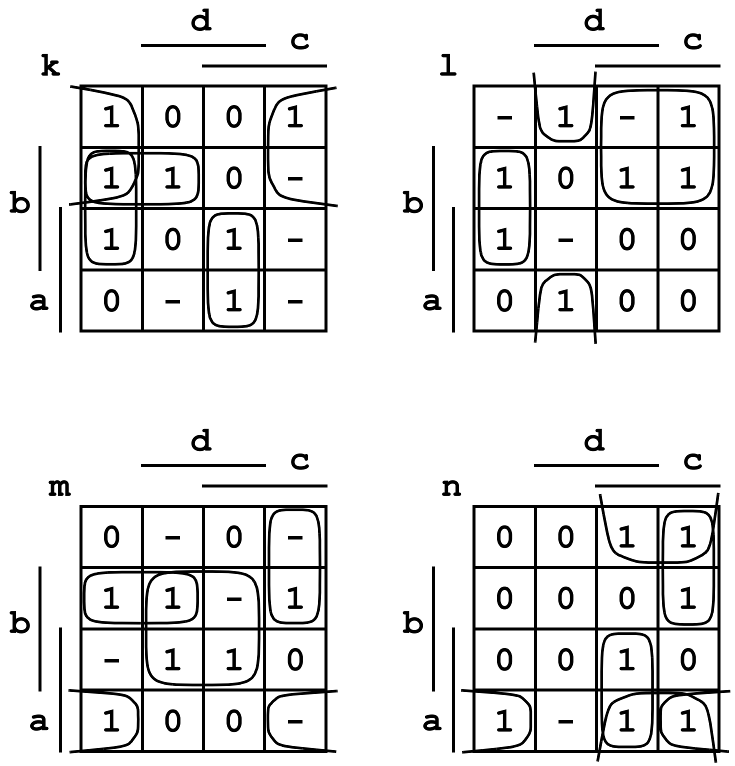
TTÜ1918

Defining Output Polarity – Example

- **Direct outputs only**

```
.i 4
.o 4
0000 1-00
0001 01-0
0010 11-1
0011 0-01
0100 1110
0101 1010
0110 -111
0111 01-0
1000 0011
1001 -10-
1010 -0-1
1011 1001
1100 11-0
1101 0-10
1110 -000
1111 1011
.e
```

```
espresso
.i 4
.o 4
.p 10
-001 0100
-100 1100
1-11 1001
10-0 0011
010- 1010
-1-1 0010
0-10 0011
0-1- 0100
-01- 0001
0--0 1000
.e
```



4 2-AND, 6 3-AND, 1 3-OR, 3 4-OR -- 41 literals in total

Defining Output Polarity – Example

- **Inverted outputs?**

```
espresso -Dopoall
.i 4
.o 4
.p 9
#ph. 0110
11-0 0001
10-0 1010
0111 1101
00-1 1100
0-10 0110
-100 0110
0-0- 0001
-1-1 0010
1-01 1101
.e
```

```
Summary of possibilities
0000 -- c=10(0) in=28 out=14 tot=42
0001 -- c=10(0) in=29 out=16 tot=45
0010 -- c=11(0) in=29 out=14 tot=43
0011 -- c=10(0) in=28 out=15 tot=43
0100 -- c=10(0) in=28 out=15 tot=43
0101 -- c=10(0) in=28 out=16 tot=44
0110 -- c=9(0) in=26 out=17 tot=43
0111 -- c=9(0) in=25 out=17 tot=42
1000 -- c=9(0) in=25 out=15 tot=40
1001 -- c=10(0) in=26 out=14 tot=40
1010 -- c=10(0) in=25 out=14 tot=39
1011 -- c=9(0) in=24 out=15 tot=39
1100 -- c=9(0) in=25 out=17 tot=42
1101 -- c=10(0) in=27 out=15 tot=42
1110 -- c=11(0) in=29 out=16 tot=45
1111 -- c=10(0) in=26 out=15 tot=41
```

k (d, c, b, a)

1	0	0	1
1	1	0	-
1	0	1	-
0	-	1	-

l (d, c, b, a)

-	1	-	1
1	0	1	1
1	-	0	0
0	1	0	0

m (d, c, b, a)

0	-	0	-
1	1	-	1
-	1	1	0
1	0	0	-

n (d, c, b, a)

0	0	1	1
0	0	0	1
0	0	1	0
1	-	1	1

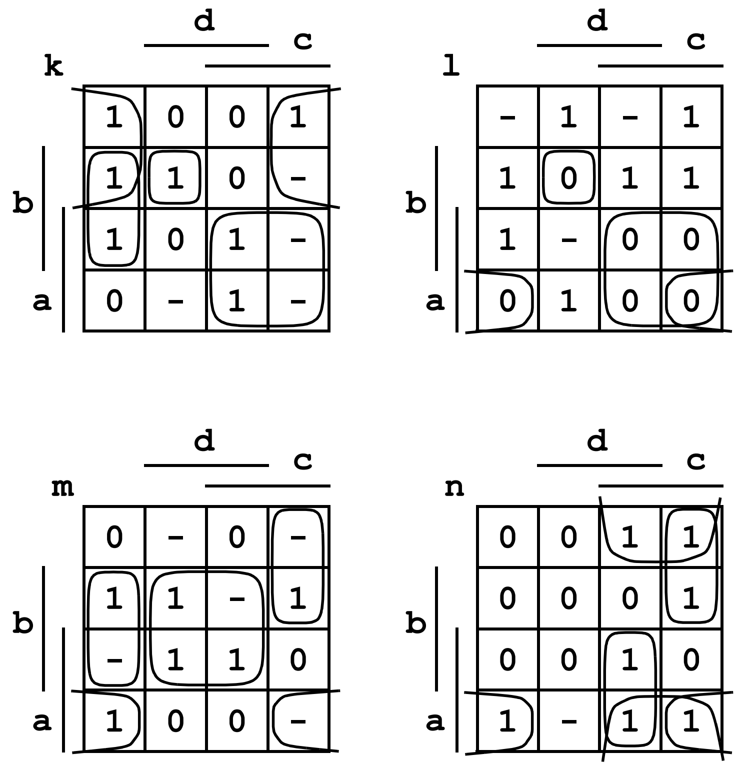
2 2-AND, 6 3-AND, 1 4-AND, 3 4-OR, 1 5-OR -- 43 literals in total

Defining Output Polarity – Example

- Not the number of implicants but the number of literals?

```
.i 4
.o 4
0000 1-00
0001 00-0
0010 10-1
0011 0-01
0100 1010
0101 1110
0110 -011
0111 00-0
1000 0111
1001 -00-
1010 -1-1
1011 1101
1100 10-0
1101 0-10
1110 -100
1111 1111
.e
```

```
...
1011 -- c=9(0) in=24 out=15 tot=39
...
1111 -- c=10(0) in=26 out=15 tot=41

Invert the second output?

.p 9
0101 1100
1-11 0001
-100 1010
-1-1 0010
-01- 0001
0-10 0011
0--0 1000
10-0 0111
1-1- 1100
```



4 2-AND, 4 3-AND, 1 4-AND, 1 3-OR, 3 4-OR -- 39 literals in total

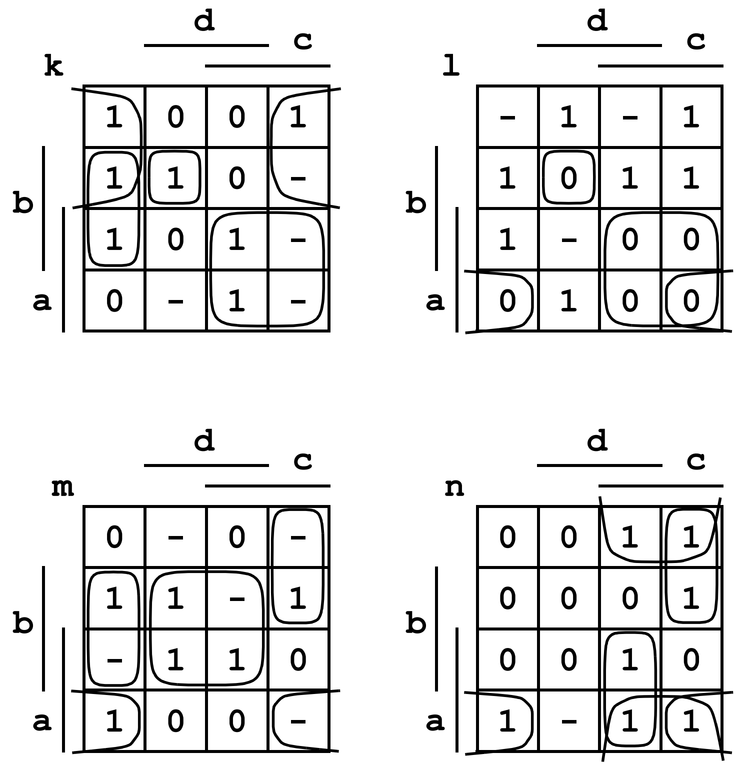
Defining Output Polarity – Example

- **More possibilities from *espresso***

```
.i 4
.o 4
.phase 1011
0000 1-00
0001 01-0
0010 11-1
0011 0-01
0100 1110
0101 1010
0110 -111
0111 01-0
1000 0011
1001 -10-
1010 -0-1
1011 1001
1100 11-0
1101 0-10
1110 -000
1111 1011
.e
```

- **Use ".phase 1011"**
- **No need to change outputs**

```
.p 9
0101 1100
1-11 0001
-100 1010
-1-1 0010
-01- 0001
0-10 0011
0--0 1000
10-0 0111
1-1- 1100
```

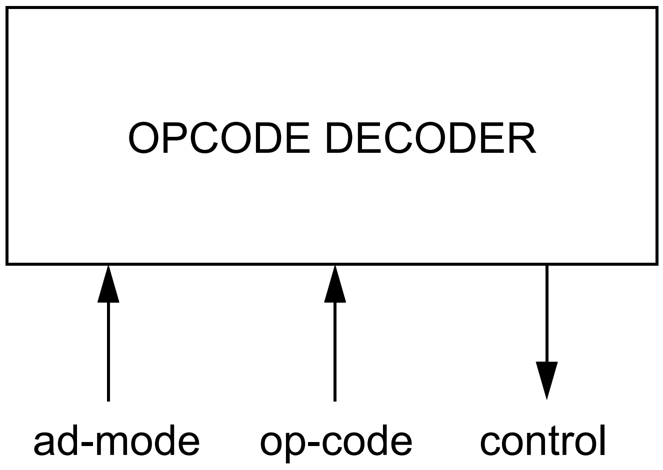


`4 2-AND, 4 3-AND, 1 4-AND, 1 3-OR, 3 4-OR -- 39 literals in total`

Symbolic Minimization and Encoding

- **Symbols as 1-hot encoded words**
- **MVL minimizer treated as**

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



Example – Getting Constraints

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

espresso input

```
.mv 3 0 3 4 4
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
.e
```

espresso output

```
.mv 3 0 3 4 4
.p 6
 001 1000 0100
 001 0001 0010
 010 1100 0100
 010 0011 0010
 001 0110 0001
 100 1111 1000
.e
```

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

Example – Encoding

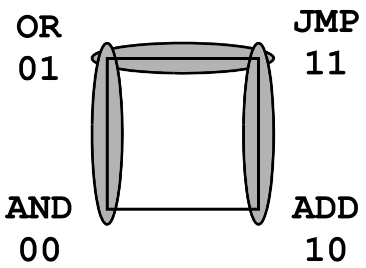
1-hot / MV		
100	1111	1000
010	1100	0100
001	1000	0100
010	0011	0010
001	0001	0010
001	0110	0001

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

Constraints as Literals

Literals

AND,OR,JMP,ADD
AND,OR
JMP,ADD
OR,JMP



AND	00
OR	01
JMP	11
ADD	10

INDEX	00
DIR	01
IND	11

Encoded Cover

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