(Two-level) Logic Synthesis
Quine/McCluskey algorithm

Becker/Molitor, Chapter 7.3

Jan Reineke
Universität des Saarlandes

Algorithm to compute a minimal polynomial

1. Quine/McCluskey's algorithm
   to compute all prime implicants
2. Solution of the "covering problem", i.e., selecting a subset of the prime implicants, such that their disjunction is a polynomial for f that has minimal cost.

Quine's algorithm

Quine-Prime-Implicants(f: $\mathbf{B}^n$ -> $\mathbf{B}$)
begin
L_0 := Minterm(f)
$i := 1$
$Prime(f) := \emptyset$
while $(L_{i-1} \neq \emptyset)$ and $(i \leq n)$
loop
$L_i := \{m \mid |m| = n\text{-}i,\ m \cdot x \text{ and } m \cdot x' \text{ are in } L_{i-1} \text{ for some } x\}$
//*Comment*: L_i contains all implicants of f of length n-i
$P_i := \{m \mid m \in L_{i-1} \text{ and } m \text{ is not covered by any } m' \in L_i\}$
$Prime(f) := Prime(f) \cup P_i$
$i := i+1$
pool
return $Prime(f) \cup L_{i-1}$
end

Improvement by McCluskey

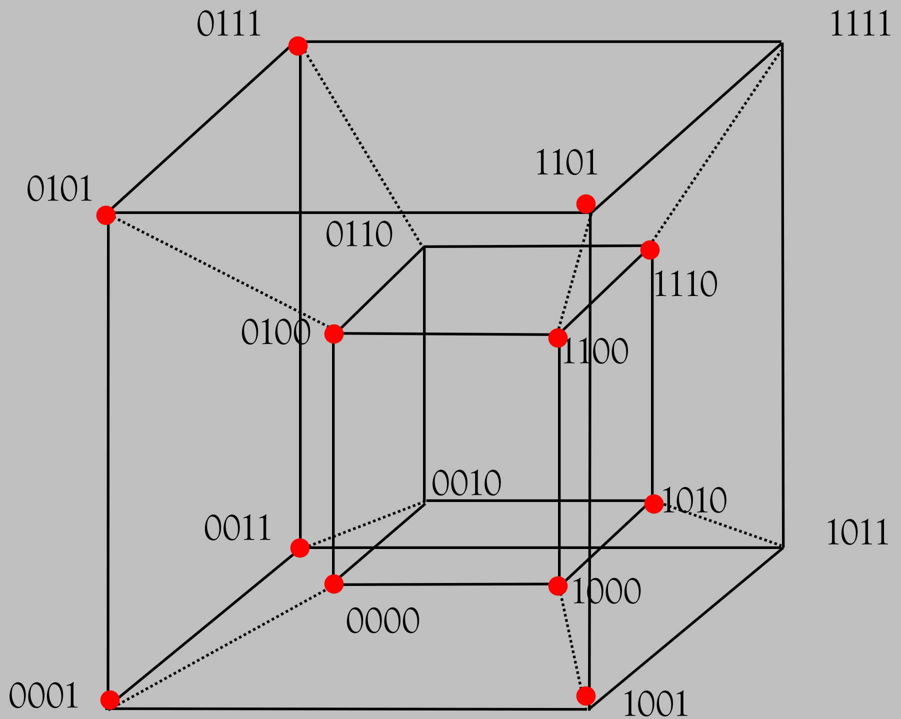
Compare only those monomials

- that contain the **same variables**, and
- whose number of **positive literals differs by one**.

Can be achieved as follows:

- Partition L_i into classes L_i^M with $M \subseteq \{x_1, \ldots x_n\}$ *and* $|M|=n-i$. L_i^M contains the implicants of L_i *whose literals are* M.
- Order the monomials in L_i^M according to their number of positive literals.

Quine-McCluskey algorithm: Example

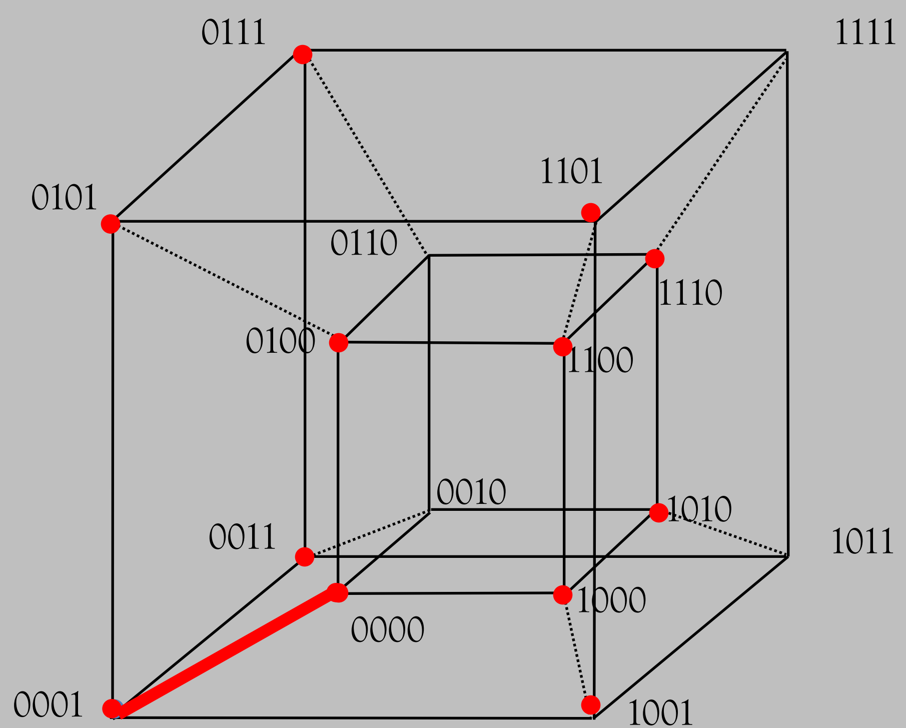


$L_0^{\{x1,x2,x3,x4\}}$:

0 0 0 0 ← stands for $x_1'x_2'x_3'x_4'$
0 0 0 1
0 1 0 0
1 0 0 0
0 0 1 1
0 1 0 1
1 0 0 1 ← stands for $x_1x_2'x_3'x_4$
1 0 1 0
1 1 0 0
0 1 1 1
1 1 0 1
1 1 1 0

Need to only compare monomials from adjacent blocks!

Quine-McCluskey algorithm: Example



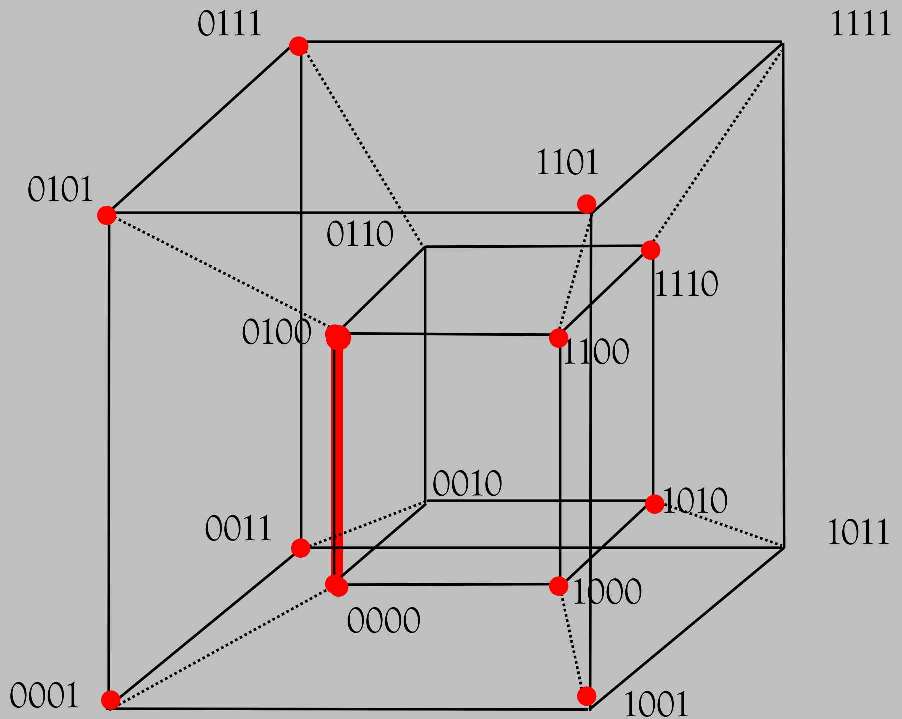
$L_0^{\{x1,x2,x3,x4\}}$:

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

$L_1^{\{x1,x2,x3\}}$:

0 0 0 -

Quine-McCluskey algorithm: Example



$L_0^{\{x1,x2,x3,x4\}}$:

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

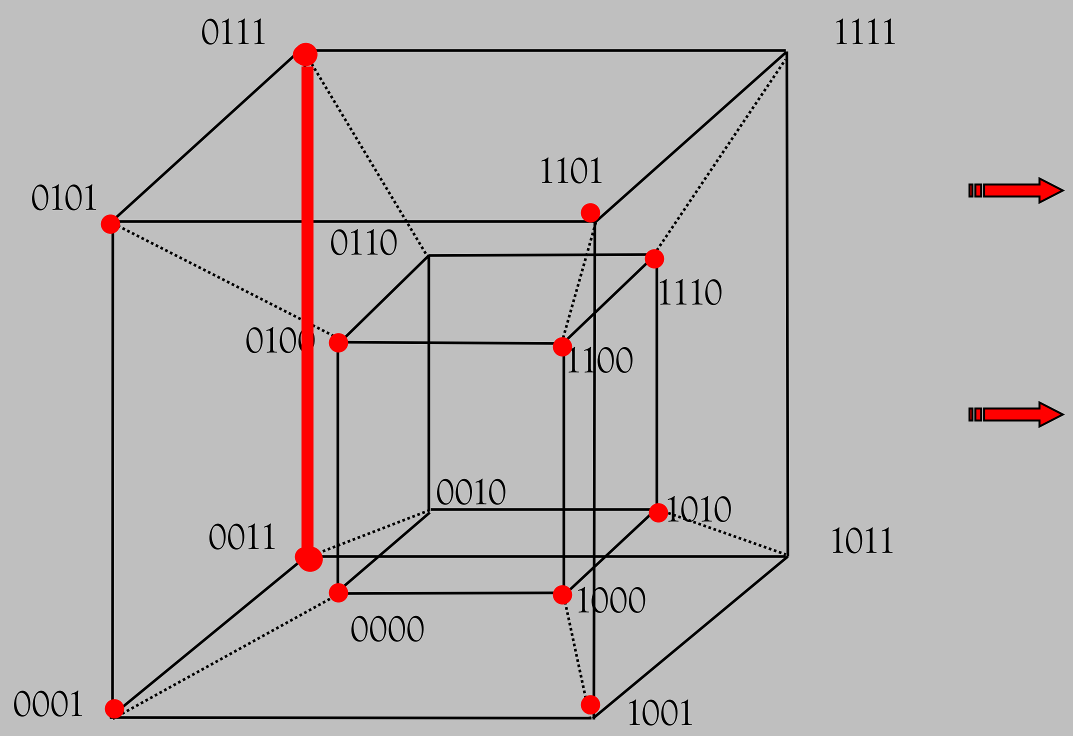
$L_1^{\{x1,x2,x3\}}$:

0 0 0 -

$L_1^{\{x1,x3,x4\}}$:

0 - 0 0

Quine-McCluskey algorithm: Example



$L_0^{\{x1,x2,x3,x4\}}$:

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

$L_1^{\{x1,x2,x3\}}$:

0 0 0 -

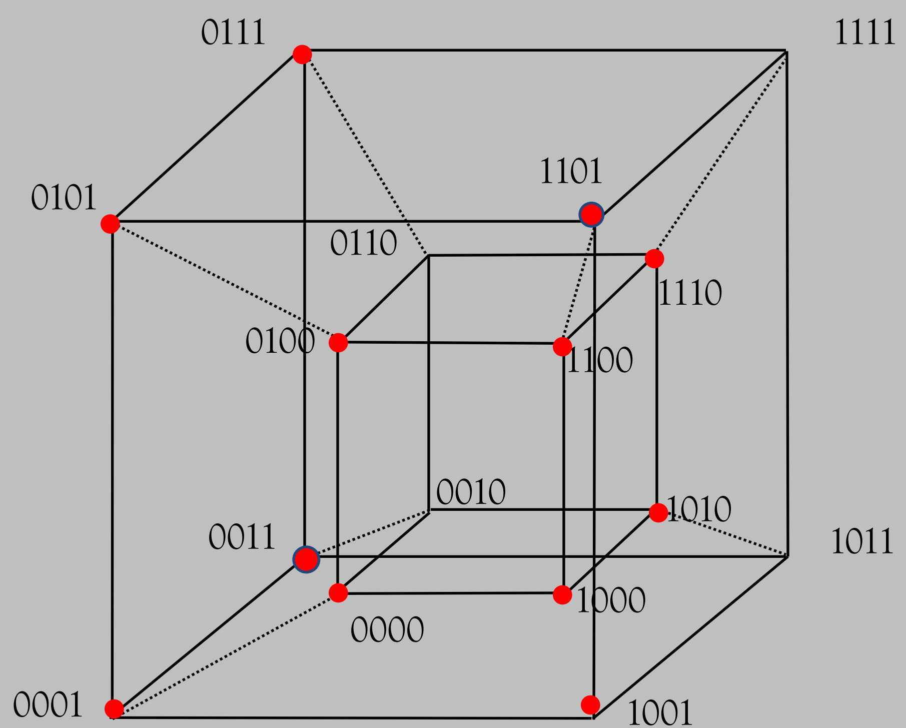
$L_1^{\{x1,x3,x4\}}$:

0 - 0 0

....

0 - 1 1

Quine-McCluskey algorithm: Example



$L_0^{\{x1,x2,x3,x4\}}$:

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

$L_1^{\{x1,x2,x3\}}$:

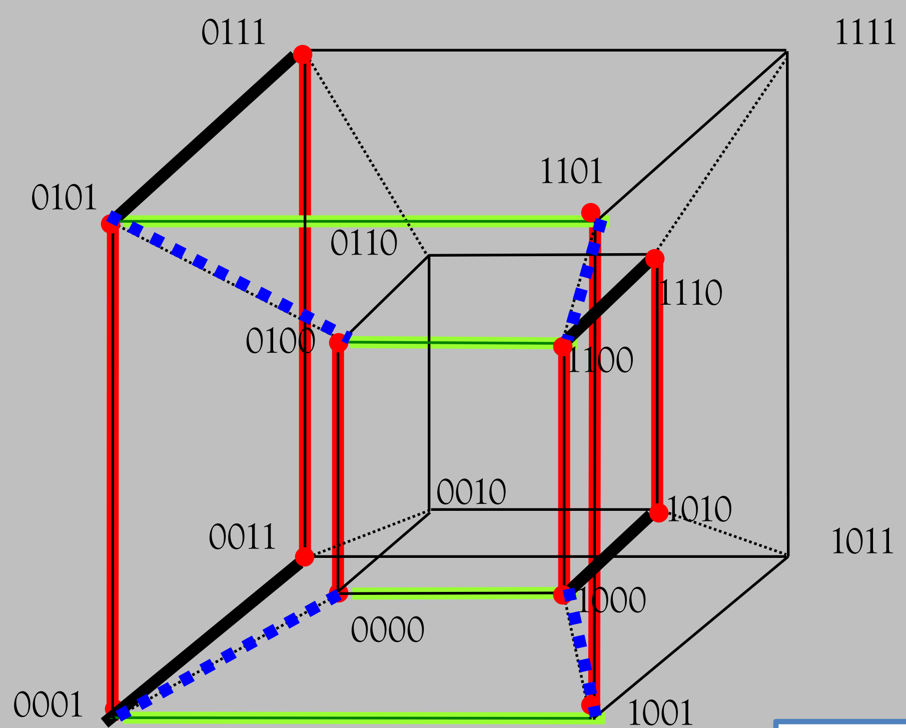
0 0 0 -

$L_1^{\{x1,x3,x4\}}$:

0 - 0 0
....
0 - 1 1

Cannot be simplified, as they are not adjacent.

Quine-McCluskey algorithm: Example
... some steps later



$L_1^{\{x1,x2,x4\}}$:

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

$L_1^{\{x1,x2,x3\}}$:

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

$L_1^{\{x2,x3,x4\}}$:

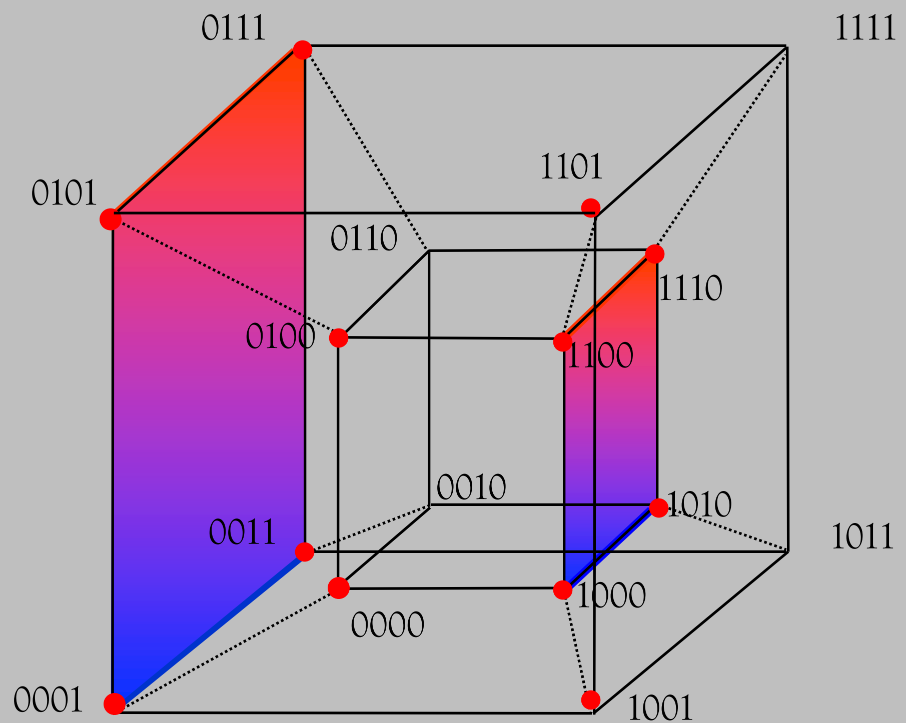
- 0 0 0
- 0 0 1
- 1 0 0
- 1 0 1

$L_1^{\{x1,x3,x4\}}$:

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

All minterms of *f* are covered by edges that are implicants ⇒ *Prime*(f) = ∅.

Quine-McCluskey algorithm: Example



$L_1^{\{x1,x2,x4\}}$:

0 0 - 1

1 0 - 0

0 1 - 1

1 1 - 0

$L_1^{\{x1,x2,x3\}}$:

0 0 0 -

0 1 0 -

1 0 0 -

1 1 0 -

$L_1^{\{x2,x3,x4\}}$:

- 0 0 0

- 0 0 1

- 1 0 0

- 1 0 1

$L_1^{\{x1,x3,x4\}}$:

0 - 0 0

0 - 0 1

1 - 0 0

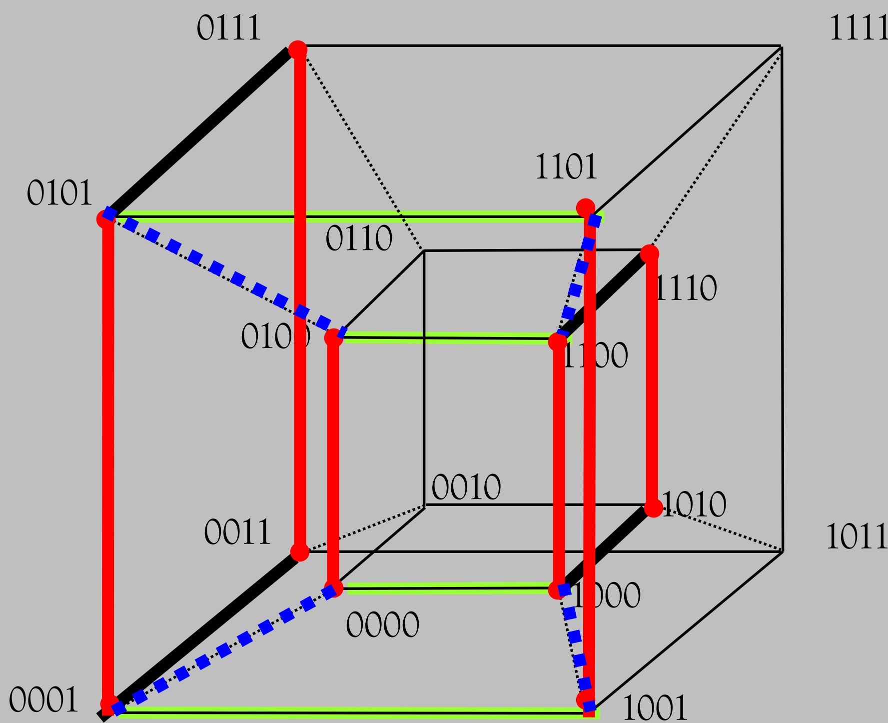
0 - 1 1

1 - 0 1

1 - 1 0

All implicants from $L_1^{\{x1,x2,x4\}}$ are covered by surfaces that are implicants $\Rightarrow$ *Prime*(f) = $\emptyset$.

Quine-McCluskey algorithm: Example



$L_1^{\{x1,x2,x4\}}$:

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

$L_1^{\{x1,x2,x3\}}$:

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

$L_1^{\{x2,x3,x4\}}$:

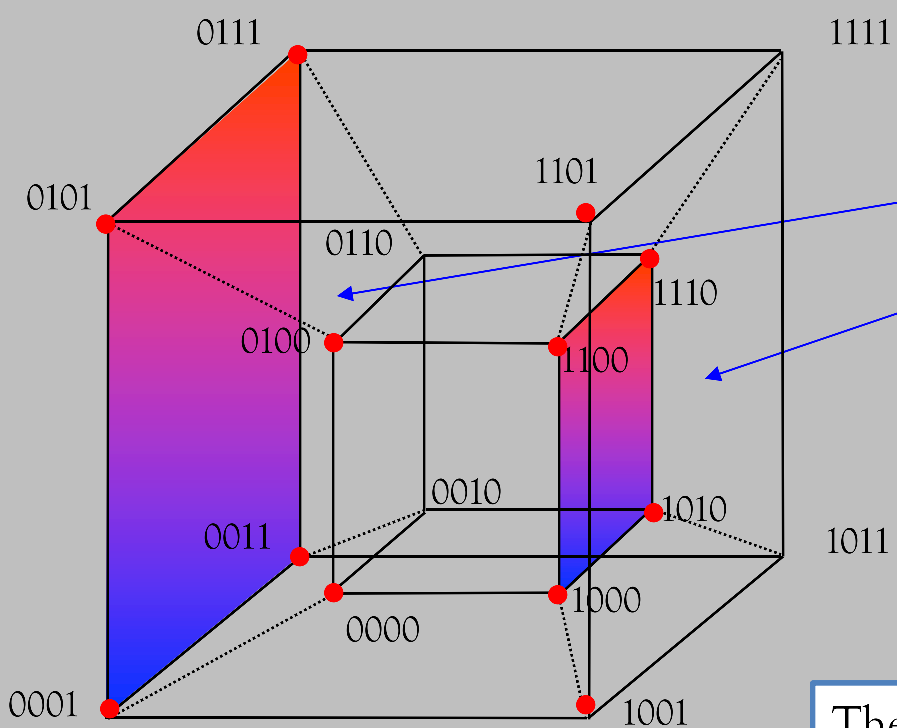
- 0 0 0
- 0 0 1
- 1 0 0
- 1 0 1

$L_1^{\{x1,x3,x4\}}$:

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

All implicants from L_1^M are covered by surfaces that are implicants $\Rightarrow$ *Prime*(f) = $\emptyset$.

Quine-McCluskey algorithm: Example



$L_2^{\{x1,x2\}}$:

$L_2^{\{x1,x3\}}$:
<u>0 - 0 -</u>
1 - 0 -

$L_2^{\{x1,x4\}}$:
0 - - 1
<u>1 - - 0</u>

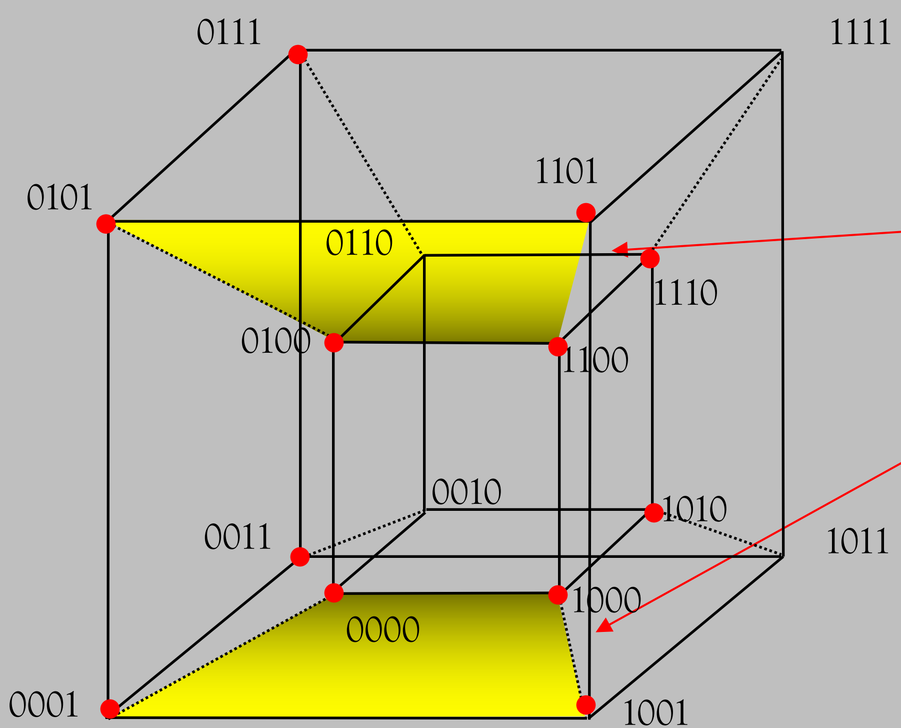
$L_2^{\{x2,x3\}}$:
<u>- 0 0 -</u>
- 1 0 -

$L_2^{\{x2,x4\}}$:

$L_2^{\{x3,x4\}}$:
<u>- - 0 0</u>
- - 0 1

The marked 2D implicants are not part of 3D implicants.
So they are **prime**! ⇒ *Prime(f)* = $\{x_1'x_4 , x_1 x_4'\}$

Quine-McCluskey algorithm: Example



$L_2^{\{x1,x2\}}$:

$L_2^{\{x1,x3\}}$:

0 - 0 -

1 - 0 -

$L_2^{\{x1,x4\}}$:

0 - - 1

1 - - 0

$L_2^{\{x2,x3\}}$:

- 0 0 -

- 1 0 -

$L_2^{\{x2,x4\}}$:

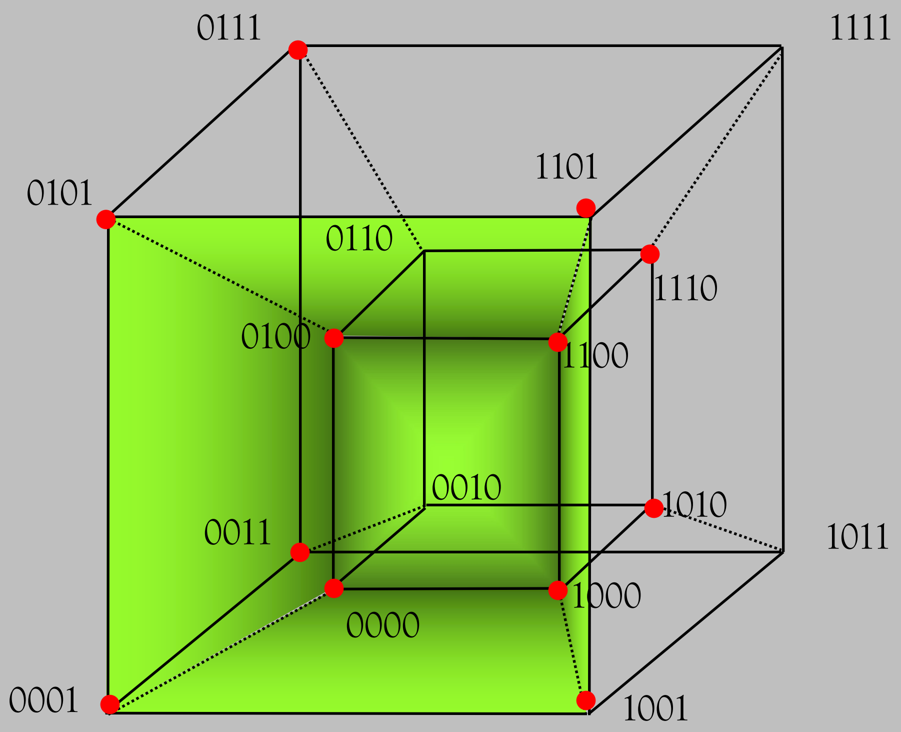
$L_2^{\{x3,x4\}}$:

- - 0 0

- - 0 1

The marked 2D implicants are covered by a 3D implicant.
So they are **not prime**! $\Rightarrow$ *Prime(f)* = $\{x_1'x_4 , x_1x_4'\}$

Quine-McCluskey algorithm: Example



$L_3^{\{x1\}}$: $L_3^{\{x2\}}$:

$L_3^{\{x3\}}$: $L_3^{\{x4\}}$:

- - 0 -

→ *Prime(f)* = $\{x_1'x_4 , x_1 x_4', x_3'\}$

Correctness of Quine-McCluskey

Quine-Prime-Implicants(f: $\mathbf{B}^n$ -> $\mathbf{B}$)
begin
L_0 := Minterm(f)
$i := 1$
$Prime(f) := \varnothing$
while $(L_{i-1} \neq \varnothing)$ and $(i \leq n)$
loop
$L_i := \{m \mid |m| = n\text{-}i,\ m \cdot x$ and $m \cdot x'$ are in L_{i-1} for some $x\}$
//Comment: L_i contains all implicants of f of length $n\text{-}i$
$P_i := \{m \mid m \in L_{i-1}$ and m is not covered by any $m' \in L_i\}$
$Prime(f) := Prime(f) \cup P_i$
$i:=i+1$
pool
return $Prime(f) \cup L_{i-1}$
end

Correctness of Quine's algorithm

Theorem:

After any iteration i, for $i=0, 1, \ldots, n$, we have:

(1) L_i contains all implicants with *exactly* $n-i$ literals

(2) Prime(f) contains the prime implicants of f with at least $n-i+1$ literals

Theorem:

After any iteration i, for i=0,1, ..., n, we have:

(1) L_i contains all implicants with *exactly* n-i literals

(2) Prime(f) contains the prime implicants of f with at least n-i+1 literals

Proof of (1): (by induction over i:) [We initially ignore the optimized termination condition $L_i \neq \emptyset$]

Induction base (i=0):

Then, $L_i = L_0$ = Minterm(f).

From the Theorem on Implicants, it follows immediately that the implicants with *n* literals (if there are exactly *n* variables) correspond to the minterms (there cannot be any implicants with *n+1* literals).

Induction step (i+1):

From the Theorem on Implicants we know that for each implicant m with n-(i+1)=n-i-1 literals, there must be implicants m·x and m·x' with n-i literals. Due to our inductive hypothesis, those implicants must be contained L_i. Thus, each implicant m with n-(i+1)=n-i-1 literals must be contained in L_{i+1} after the assignment to L_{i+1}.

Theorem:

After any iteration i, for i=0,1, ..., n, we have:

(1) L_i contains all implicants with *exactly* n-i literals

(2) Prime(f) contains the prime implicants of f with at least n-i+1 literals

Proof of (2): (by induction over i:) [We initially ignore the optimized termination condition $L_i \neq \varnothing$]

We assume (1) to be proven based on the previous proof.

Induction base (i=0):

Then Prime(f) = $\varnothing$.

As there are only *n* variables, there cannot be any implicants nor prime implicants with *n+1* literals. And thus Prime(f) = $\varnothing$ is correct.

Induction step (i+1):

By definition, prime implicants are maximal implicants. If an implicant is non-maximal, then, in particular, there are larger implicants that contain exactly one literal less. An implicant is declared prime by the algorithm, if no such larger implicant exists.

Termination condition: If the termination condition applies, i.e. if we have $L_i = \varnothing$, then L_i would also have been empty in all future iterations, had the loop not terminated.

Complexity of the algorithm

Lemma:

There are 3^n distinct monomials in n variables.

Proof

For every monomial m and every variable x among the n variables exactly one of the following 3 possibilities applies:

- m contains neither the positive nor the negative literal of x
- m contains the positive literal x
- m contains the negative literal x'

Complexity of the algorithm

Theorem **(Complexity of the Quine-McCluskey algorithm):**
The runtime of the algorithm is in $O(n^2 3^n)$ and $O(\log^2 N \cdot N^{\log 3})$, where $N=2^n$ is the size of the truth table.

Proof

- Each of the (maximally) 3^n monomials is compared with at most n other monomials throughout the algorithm. (Why?)
- Given a monomial $m \cdot x$. Searching for $m \cdot x'$ in L_i can be performed in $O(n)$ using appropriate data structures.

Part 2 follows by simple calculation:

$3^n = (2^{\log 3})^n = (2^n)^{\log 3} = N^{\log 3}$, and

$n^2 = (\log N)^2 = \log^2 N$.

The matrix covering problem

Given the set of prime implicants Prime(f) of f.

Wanted:

A cost-optimal subset M of Prime(f), such that the disjunction of the monomials in M describes the function f.

The matrix covering problem: Formalization

Let us define a Boolean matrix **PIT(f)**, the prime implicant table of f:

- The rows correspond to the prime implicants Prime(f) of f
- The columns correspond to the minterms of f
- Let $min(\alpha)$ be an arbitrary minterm of f.
  Then, for each prime implicant m, we have:

  $$\mathbf{PIT}(f)[m, \min(\alpha)] = \psi(m)(\alpha).$$

  So the table entry at [m, min(α)] is 1, if and only if,
  min(α) describes a node of the subcube m.

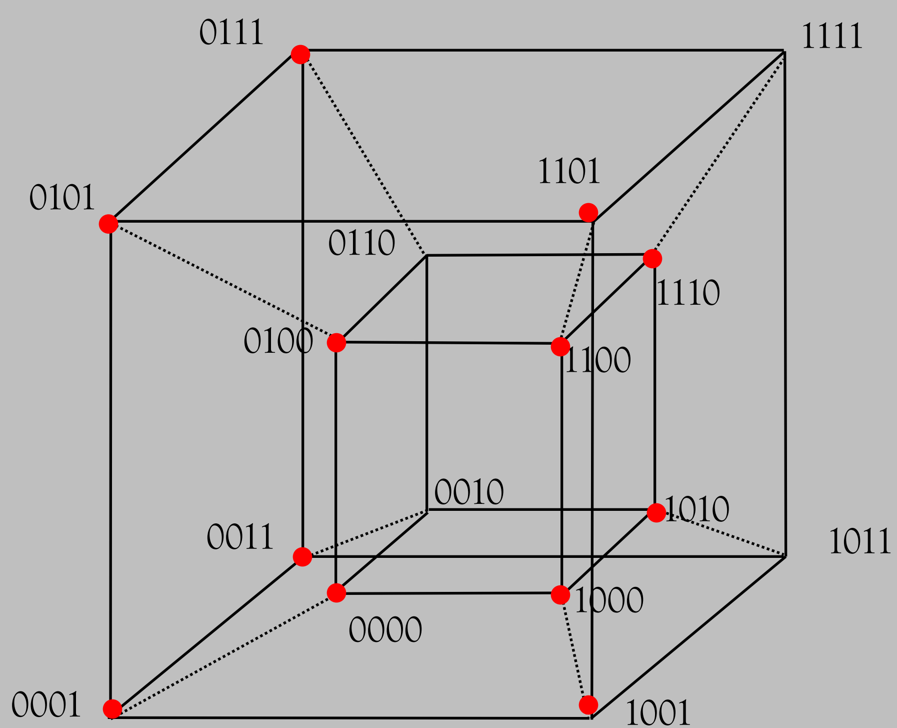
Wanted:

A cost-optimal subset M of Prime(f),

such that every column of **PIT(f)** is covered,

i.e. $\forall \alpha \in \mathbf{ON}(f)\ \exists m \in M$ with $\mathbf{PIT}(f)[m, min(\alpha)]=1$.

The matrix covering problem: Example



$\textit{Prime}(f) = \{x_1'x_4,\ x_1x_4',\ x_3'\}$

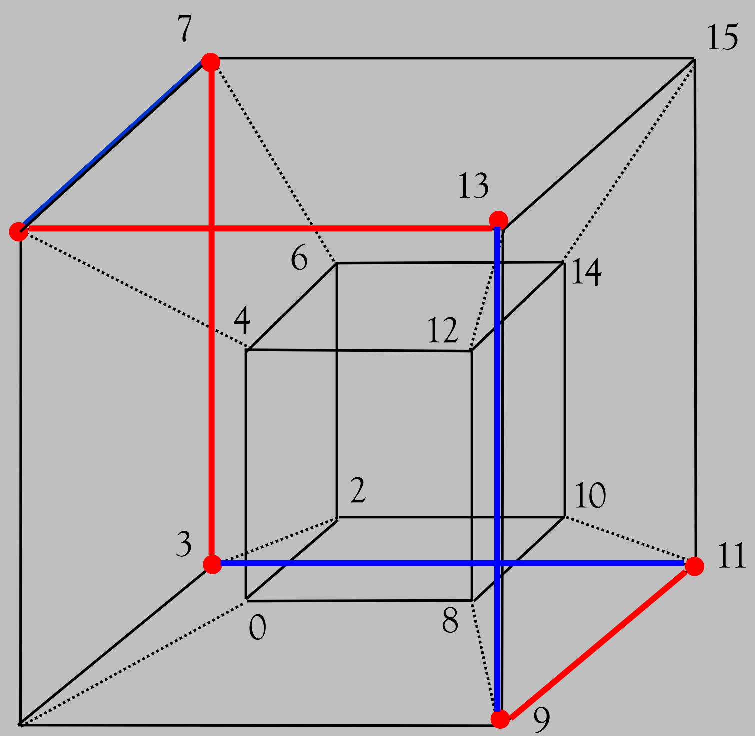
Which subset of the prime implicants solves the matrix covering problem?

Prime implicant table PIT(f):

	0	1	3	4	5	7	8	9	10	12	13	14
$x_1'x_4$		1	(1)		1	1						
x_1x_4'							1		1	1		(1)
x_3'	1	1		1	1		1	(1)		1	1	

⇒ All prime implicants are **essential!**

The matrix covering problem: Another example!



Prime implicant table PIT(f):

	3	5	7	9	11	13
{7,5}		1	1			
{5,13}		1				1
{13,9}				1		1
{9,11}				1	1	
{11,3}	1				1	
{3,7}	1		1			

No prime implicant is essential!

$$Prime(f) = \{\{7,5\},\{5,13\},\{13,9\}\{9,11\},\{11,3\},\{3,7\}\}$$

First reduction rule

Definition:

A prime implicant m of f is called **essential**, if there is a minterm $min(\alpha)$ of f, that is only covered by m. Formally:

- **PIT**$(f)[m,min(\alpha)]=1$
- **PIT**$(f)[m',min(\alpha)]=0$ for all other prime implicants m' of f

Lemma:

Every minimal polynomial of f contains all **essential** prime implicants of f.

1. Reduction Rule:

Remove from the prime implicant table **PIT**(f) all essential prime implicants and all minterms that are covered by these prime implicants.

First reduction rule: Example

essential (rows 1, 2, 3, 4)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1				1												
2		1				1											
3			1				1										
4				1				1									
5					1				1								1
6						1				1							1
7							1				1						
8								1				1					
9									1				1				
10										1				1			1
11											1				1		
12												1				1	
13													1	1	1	1	

First reduction rule: Example

Covering problem after the application of the first reduction rule:

	9	10	11	12	13	14	15	16	17
5	1								1
6		1							1
7			1						
8				1					
9	1				1				
10		1				1			1
11			1				1		
12				1				1	
13					1	1	1	1	

The matrix does not contain any further essential rows!

Second reduction rule

Definition:

Let A be a Boolean matrix.

Column j of matrix A **dominates** column i of matrix A,

if $A[k,i] \leq A[k,j]$ for every row k.

Benefit for our problem:

If minterm w' of f dominates another minterm w of f, then we do not need to further consider w', as w has to be covered and covering w guarantees that w' will also be covered. Every prime implicant p in **PIT(f)** that covers w also covers w'.

2. Reduction Rule:

Remove all minterms from the prime implicant table **PIT(f)** that dominate another minterm in **PIT(f)**.

Second reduction rule: Example

	9	10	11	12	13	14	15	16	17
5	1								1
6		1							1
7			1						
8				1					
9	1				1				
10		1				1			1
11			1				1		
12				1				1	
13					1	1	1	1	

Column 17 dominates Column 10
=> Column 17 can be deleted!

Third reduction rule

Definition:

Let A be a Boolean matrix.

Row i of matrix A **dominates** Row j of matrix A,
if $A[i,k] \geq A[j,k]$ for every column k.

Benefit for our problem :

If prime implicant m dominates another prime implicant m',
then we do not need to further consider m', if $cost(m') \geq cost(m)$ holds.
(Convince yourself that the last condition is required.)

3. Reduction Rule

Remove all prime implicants from the prime implicant table PIT(f) that are dominated by other prime implicants that are not more expensive.

Third reduction rule: Example

Let's assume that rows 5 to 12 have the same cost.

	9	10	11	12	13	14	15	16
5	1							
6		1						
7			1					
8				1				
9	1				1			
10		1				1		
11			1				1	
12				1				1
13					1	1	1	1

Rows 5–8: are dominated

Third reduction rule

Covering problem after the application of the third reduction rule:

	9	10	11	12	13	14	15	16
9	1				1			
10		1				1		
11			1				1	
12				1				1
13					1	1	1	1

Note that the **first reduction rule** is now applicable again, as rows 9, 10, 11, 12 are **essential**.

→ The resulting matrix is empty
→ The minimal polynomial is 1+2+3+4+9+10+11+12

... does not contain the row with the maximal number of ones!

Cyclic covering problems

Definition:

A prime implicant table is called **reduced** if none of the three reduction rules is applicable.

If a reduced table is non-empty, the remaining problem is called a **cyclic covering problem**.

Prime implicant table PIT(f):

	3	5	7	9	11	13
{7,5}		1	1			
{5,13}		1				1
{13,9}				1		1
{9,11}				1	1	
{11,3}	1				1	
{3,7}	1		1			

Approaches to solve the cyclic covering problem:

- heuristic approaches
- Petrick‘s method

Petrick's method

Method:

1. Translate the PIT into a **conjunctive normal form** that contains all covering possibilities.
2. "Multiply" these out.

The minimal covering is given by the **monomial** that corresponds to the selection of prime implicants of minimal cost.

	1	2	3	4
1	1	1		
2			1	1
3	1		1	
4		1		1
5	1			1
6		1	1	

required to cover column 4

... is translated into:

(1+3+5)(1+4+6)(2+3+6)(2+4+5)

= (1+1·4+1·6+1·3+3·4+3·6+1·5+4·5+5·6)*
(2+2·4+2·5+2·3+3·4+3·5+2·6+4·6+5·6)

= 1·2+1·2·4+1·2·5+1·2·3+1·3·4+…+3·4+…+5·6

assuming the same cost for all prime implicants 1·2, 3·4 and 5·6 are minimal

Summary, Outlook

Theorem (Quine):
Every minimal polynomial p of a Boolean function f consists only of prime implicants of f.

Quine/McCluskey algorithm

1. Compute all prime implicants
   – Cleverly group the implicants
2. Search for cost-optimal covering
   – Reduction rules:
     - essential prime implicants
     - dominated rows
     - dominated columns

Outlook: Multi-level circuits