Logic Synthesis and Verification

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Fall 2010

SOPs and Incompletely Specified Functions

Reading:

Logic Synthesis in a Nutshell

Section 2

most of the following slides are by courtesy of Andreas Kuehlmann

Boolean Function Representation

Sum of Products

- A function can be represented by a sum of cubes (products):
  - E.g., f = ab + ac + bc
    Since each cube is a product of literals, this is a "sum of products" (SOP) representation
- An SOP can be thought of as a set of cubes F
  - E.g., F = {ab, ac, bc}
- A set of cubes that represents f is called a cover of f
  - E.g.,
    F_1={ab, ac, bc} and F_2={abc, abc', ab'c, a'bc} are covers of f = ab + ac + bc.

List of Cubes (Cover Matrix)

- We often use a matrix notation to represent a cover:
  - Example
    F = ac + c'd =

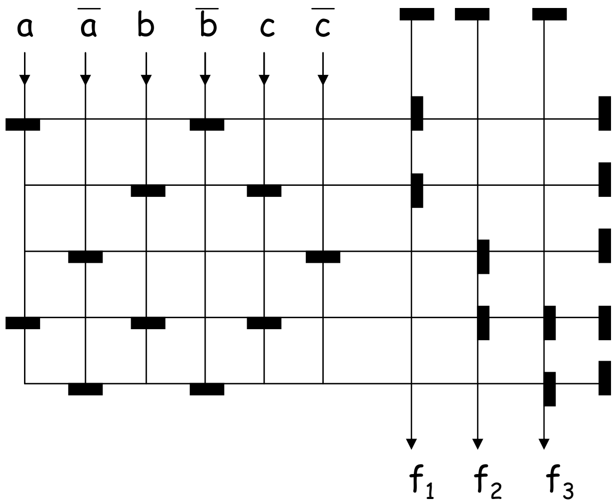
	a	b	c	d		a	b	c	d
a c →	1	2	1	2	or	1	-	1	-
c'd →	2	2	0	1		-	-	0	1

- Each row represents a cube
- 1 means that the positive literal appears in the cube
- 0 means that the negative literal appears in the cube
- 2 (or -) means that the variable does not appear in the cube. It implicitly represents both 0 and 1 values.

PLA

- A PLA is a (multiple-output) function $f : B^n \rightarrow B^m$ represented in SOP form

$n=3, m=3$



cover matrix

abc	f_1	f_2	f_3
10-	1	-	-
-11	1	-	-
0-0	-	1	-
111	-	1	1
00-	-	-	1

PLA

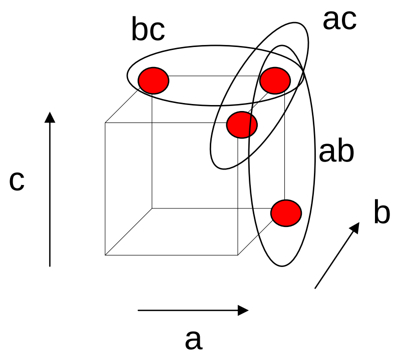
- Each distinct cube appears just once in the AND-plane, and can be shared by (multiple) outputs in the OR-plane, e.g., cube (abc)

- Extensions from single-output to multiple-output minimization theory are straightforward

SOP

- The cover (set of SOPs) can efficiently represent many practical logic functions (i.e., for many practical functions, there exist small covers)

- Two-level minimization seeks the cover of minimum size (least number of cubes)

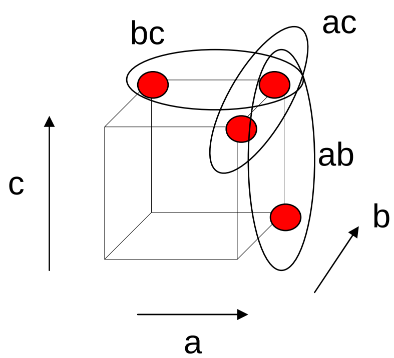


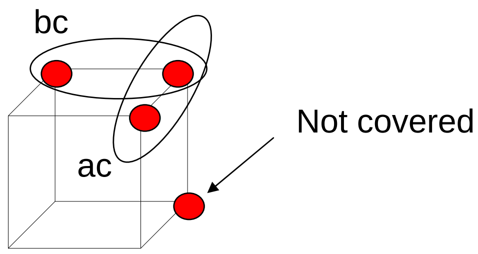
Note that each onset minterm is "covered" by at least one of the cubes!
None of the offset minterms is covered

Irredundant Cube

- Let $F = \{c_1, c_2, \ldots, c_k\}$ be a cover for f, i.e.,
  $f = \sum_{i=1}^{k} c_i$
  A cube $c_i \in F$ is irredundant if $F \setminus \{c_i\} \neq f$

  - Example
    $f = ab + ac + bc$





$F \setminus \{ab\} \neq f$

Prime Cube

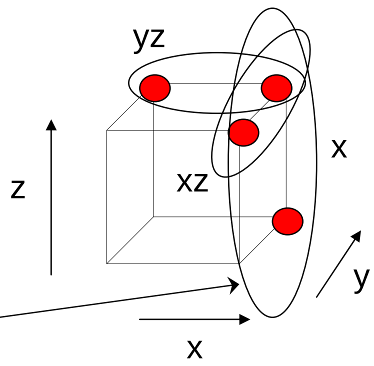
- A literal x (a variable or its negation) of cube $c \in F$ (cover of f) is prime if $(F \setminus \{c\}) \cup \{c_x\} \neq f$, where c_x (cofactor w.r.t. x) is c with literal x of c deleted
- A cube of F is prime if all its literals are prime

Example

$f = xy + xz + yz$

$c = xy$; $c_y = x$ (literal y deleted)

$F \setminus \{c\} \cup \{c_y\} = x + xz + yz$



inequivalent to f since
offset vertex is covered

Prime and Irredundant Cover

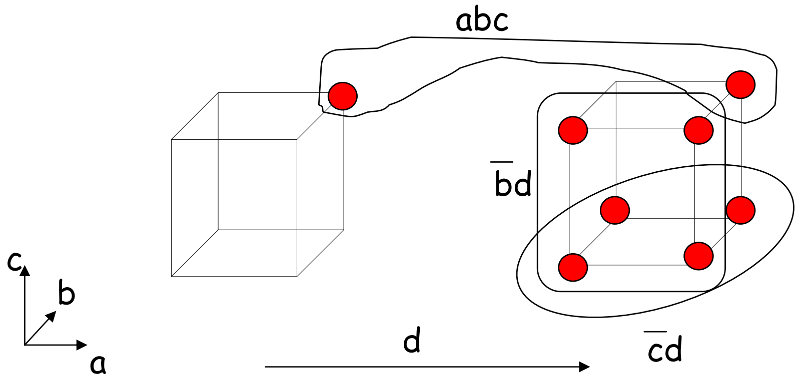
- Definition 1. A cover is prime (resp. irredundant) if all its cubes are prime (resp. irredundant)
- Definition 2. A prime (cube) of f is essential (essential prime) if there is a onset minterm (essential vertex) in that prime but not in any other prime.
- Definition 3. Two cubes are orthogonal if they do not have any minterm in common
  - E.g. $c_1 = x\,y$ $c_2 = y'z$ are orthogonal
    $c_1 = x'y$ $c_2 = y\,z$ are not orthogonal

Prime and Irredundant Cover

Example

f = abc + b'd + c'd is prime and irredundant.

abc is essential since abcd'∈abc, but not in b'd or c'd or ad



Why is abcd not an essential vertex of abc?
What is an essential vertex of abc?
What other cube is essential? What prime is not essential?

Incompletely Specified Function

- Let F = (f, d, r) : $B^n \rightarrow \{0, 1, *\}$, where * represents "don't care".
  - f = onset function $f(x)=1 \leftrightarrow F(x)=1$
  - r = offset function $r(x)=1 \leftrightarrow F(x)=0$
  - d = don't care function $d(x)=1 \leftrightarrow F(x)=*$

- (f,d,r) forms a *partition* of B^n, i.e,
  - $f + d + r = B^n$
  - $(f \cdot d) = (f \cdot r) = (d \cdot r) = \emptyset$ (pairwise disjoint)

    (Here we don't distinguish characteristic functions and the sets they represent)

Incompletely Specified Function

- A completely specified function g is a cover for F = (f,d,r) if

  $f \subseteq g \subseteq f+d$
  - $g \cdot r = \varnothing$
  - if $x \in d$ (i.e. d(x)=1), then g(x) can be 0 or 1; if $x \in f$, then g(x) = 1; if $x \in r$, then g(x) = 0
    - We "don't care" which value g has at $x \in d$

Prime of Incompletely Specified Function

- Definition. A cube c is a prime of F = (f,d,r) if $c \subseteq f+d$ (an implicant of f+d), and no other implicant (of f+d) contains c (i.e., it is simply a prime of f+d)

- Definition. Cube c_j of cover $G = \{c_i\}$ of F = (f,d,r) is redundant if $f \subseteq G \backslash \{c_j\}$; otherwise it is irredundant

- Note that $c \subseteq f+d \leftrightarrow c \cdot r = \varnothing$

Prime of Incompletely Specified Function

- Example

  Consider logic minimization of $F(a,b,c)=(f,d,r)$ with $f=a'bc'+ab'c+abc$ and $d = abc'+ab'c'$

on

off

don't care

$F_1=\{a'bc', ab'c, abc\}$

Expand $abc \rightarrow a$

$F_2=\{a, a'bc', ab'c\}$

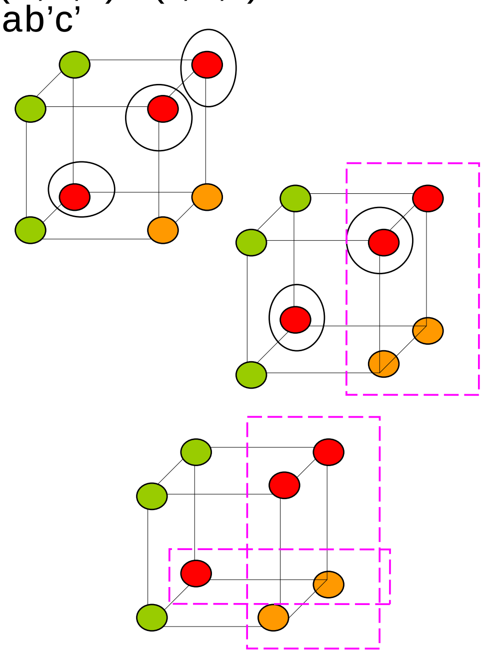
$ab'c$ is redundant

a is prime

$F_3= \{a, a'bc'\}$

Expand $a'bc' \rightarrow bc'$

$F_4= \{a, bc'\}$



Checking of Prime and Irredundancy

Let G be a cover of $F = (f,d,r)$. Let D be a cover for d

- $c_i \in G$ is redundant iff

  $c_i \subseteq (G \backslash \{c_i\}) \cup D$ (1)

  (Let $G^i \equiv G\backslash\{c_i\} \cup D$. Since $c_i \subseteq G^i$ and $f \subseteq G \subseteq f+d$, then $c_i \subseteq c_i f + c_i d$ and $c_i f \subseteq G\backslash\{c_i\}$. Thus $f \subseteq G\backslash\{c_i\}$.)

- A literal $l \in c_i$ is prime if $(c_i\backslash\{ l \})$ ($= (c_i)_l$) is not an implicant of F
- A cube c_i is a prime of F iff all literals $l \in c_i$ are prime

  Literal $l \in c_i$ is not prime $\Leftrightarrow (c_i)_l \subseteq f+d$ (2)

Note: Both tests (1) and (2) can be checked by tautology (to be explained):

- $(G^i)_{c_i} \equiv 1$ (implies c_i redundant)
- $(f \cup d)_{(c_i)_l} \equiv 1$ (implies l not prime)

  The above two cofactors are with respect to cubes instead of literals

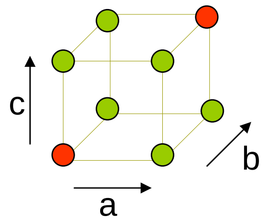
(Literal) Cofactor

- Let $f : B^n \rightarrow B$ be a Boolean function, and $x = (x_1, x_2, \ldots, x_n)$ the variables in the support of f; the cofactor f_a of f by a literal $a = x_i$ or $a = \neg x_i$ is
- $f_{x_i}(x_1, x_2, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$
- $f_{\neg x_i}(x_1, x_2, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$

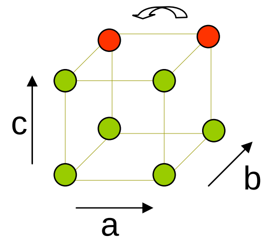
  The computation of the cofactor is a fundamental operation in Boolean reasoning!

- Example

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



$f_a = bc$



(Literal) Cofactor

The cofactor C_{x_j} of a cube C (representing some Boolean function) with respect to a literal x_j is

- C   if x_j and x_j' do not appear in C
- $C \backslash \{x_j\}$ if x_j appears positively in C, i.e., $x_j \in C$
- $\emptyset$ if x_j appears negatively in C, i.e., $x_j' \in C$

Example

$C = x_1 x_4' x_6$,

$C_{x_2} = C$ (x_2 and x_2 do not appear in C)

$C_{x_1} = x_4' x_6$ (x_1 appears positively in C)

$C_{x_4} = \emptyset$ (x_4 appears negatively in C)

(Literal) Cofactor

- Example

  F = abc' + b'd + cd

  F_b = ac' + cd

  (Just drop *b* everywhere and throw away cubes containing literal *b*')

  Cofactor and disjunction commute!

Shannon Expansion

Let $f : B^n \rightarrow B$

Shannon Expansion:

$$f = x_i f_{x_i} + x_i' f_{x_i'}$$

Theorem: F is a cover of f. Then

$$F = x_i F_{xi} + x_i' F_{x_i'}$$

We say that f and F are expanded about x_i, and x_i is called the splitting variable

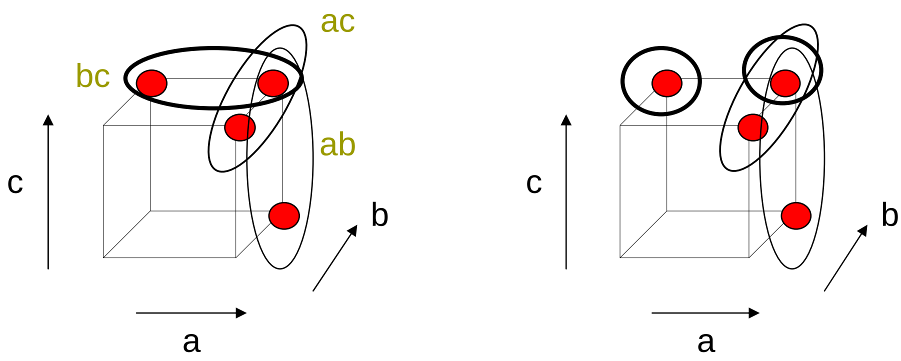
Shannon Expansion

- Example

  $F = ab + ac + bc$

  $F = a\,F_a + a'\,F_{a'}$
  $= a\,(b{+}c{+}bc){+}a'\,(bc)$
  $= ab{+}ac{+}abc{+}a'bc$

  Cube bc got split into two cubes



(Cube) Cofactor

- The cofactor f_C of f by a cube C is f with the fixed values indicated by the literals of C
  - E.g., if C = $x_i\, x_j'$, then $x_i = 1$ and $x_j = 0$
  - For C = $x_1\, x_4'\, x_6$, f_C is just the function f restricted to the subspace where $x_1 = x_6 = 1$ and $x_4 = 0$
    - Note that f_C does not depend on x_1, x_4 or x_6 anymore (However, we still consider f_C as a function of all *n* variables, it just happens to be independent of x_1, x_4 and x_6)
  - $x_1 f \neq f_{x_1}$
    - E.g., for f = ac + a'c, $a{\cdot}f_a = a{\cdot}f = a{\cdot}c$ and $f_a = c$

(Cube) Cofactor

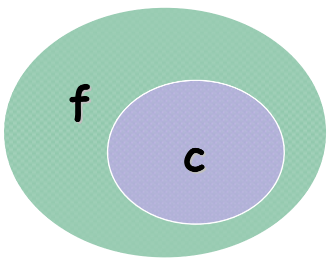
- The cofactor of the cover F of some function f is the sum of the cofactors of each of the cubes of F
- If $F=\{c_1, c_2,\ldots, c_k\}$ is a cover of f, then $F_c= \{(c_1)_c, (c_2)_c,\ldots, (c_k)_c\}$ is a cover of f_c

Containment vs. Tautology

- A fundamental theorem that connects functional containment and tautology:

  Theorem. Let c be a cube and f a function. Then $c \subseteq f \Leftrightarrow f_c \equiv 1$.

  Proof.
  We use the fact that $xf_x = xf$, and f_x is independent of x.
  ($\Leftarrow$)
  Suppose $f_c \equiv 1$. Then $cf = f_c c = c$. Thus, $c \subseteq f$.
  ($\Rightarrow$)
  Suppose $c \subseteq f$. Then $f+c=f$. In addition, $(f+c)_c = f_c+1=1$. Thus, $f_c=1$.



Checking of Prime and Irredundancy (Revisited)

Let G be a cover of F = (f,d,r). Let D be a cover for d

- $c_i \in G$ is redundant iff
  $c_i \subseteq (G \backslash \{c_i\}) \cup D$ (1)

  (Let $G^i \equiv G \backslash \{c_i\} \cup D$. Since $c_i \subseteq G^i$ and $f \subseteq G \subseteq f+d$, then $c_i \subseteq c_i f + c_i d$ and $c_i f \subseteq G \backslash \{c_i\}$. Thus $f \subseteq G \backslash \{c_i\}$.)

- A literal $l \in c_i$ is prime if $(c_i \backslash \{ l \})$ $(= (c_i)_l)$ is not an implicant of F
- A cube c_i is a prime of F iff all literals $l \in c_i$ are prime
  Literal $l \in c_i$ is not prime $\Leftrightarrow (c_i)_l \subseteq f+d$ (2)

Note: Both tests (1) and (2) can be checked by tautology (explained):

- $(G^i)_{c_i} \equiv 1$ (implies c_i redundant)
- $(f \cup d)_{(c_i)_l} \equiv 1$ (implies l not prime)
  The above two cofactors are with respect to cubes instead of literals

Generalized Cofactor

- Definition. Let f, g be completely specified functions. The generalized cofactor of f with respect to g is the incompletely specified function:

$$co(f,g) = (f \cdot g, \overline{g}, \overline{f} \cdot g)$$

- Definition. Let $\Im$ = (f, d, r) and g be given. Then

$$co(\Im, g) = (f \cdot g, d + \overline{g}, r \cdot g)$$

Shannon vs. Generalized Cofactor

- Let $g = x_i$. Shannon cofactor is

$$f_{x_i}(x_1, x_2, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

- Generalized cofactor with respect to $g=x_i$ is

$$co(f, x_i) = (f \cdot x_i, \overline{x}_i, \overline{f} \cdot x_i)$$

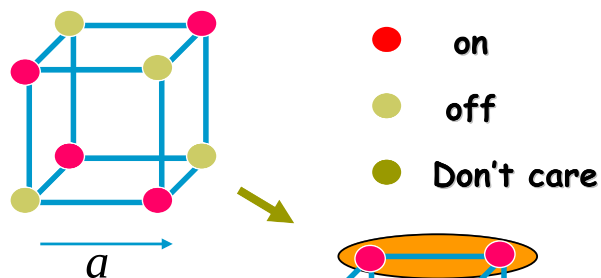
- Note that

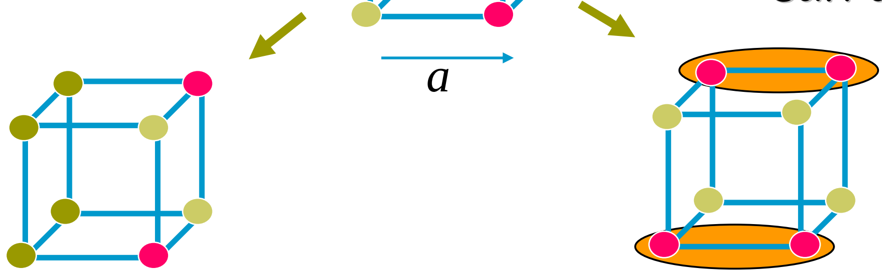
$$f \cdot x_i \subseteq f_{x_i} \subseteq f \cdot x_i + \overline{x}_i = f + \overline{x}_i$$

In fact f_{x_i} is the unique cover of $co(f, x_i)$ independent of the variable x_i .

Shannon vs. Generalized Cofactor

$$f = abc + a\overline{b}\overline{c} + \overline{a}\overline{b}c + \overline{a}b\overline{c}$$

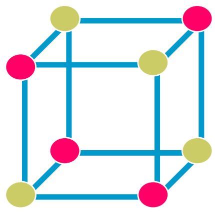




$$co(f, a) = (f \cdot a, \overline{a}, \overline{f} \cdot a)$$

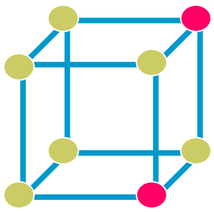
$$f_a = bc + \overline{b}\overline{c}$$

Shannon vs. Generalized Cofactor



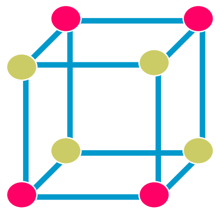
f

$$co(f,a) = (f \cdot a, \bar{a}, \bar{f} \cdot a)$$



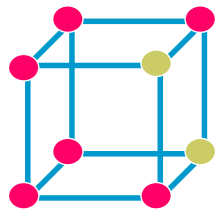
$f \cdot a$

$\subseteq$



f_a

$\subseteq$



$fa + \bar{a}$

So $f \cdot a \subseteq f_a \subseteq f + \bar{a}$

Shannon vs. Generalized Cofactor

Shannon Cofactor	Generalized Cofactor
$x \cdot f_x + \bar{x} \cdot f_{\bar{x}} = f$	$f = g \cdot co(f,g) + \bar{g} \cdot co(f,\bar{g})$
$(f_x)_y = f_{xy}$	$co(co(f,g),h) = co(f,gh)$
$(f \cdot g)_y = f_y \cdot g_y$	$co(f \cdot g,h) = co(f,h) \cdot co(g,h)$
$\left(\bar{f}\right)_x = \overline{(f_x)}$	$co(\bar{f},g) = \overline{co(f,g)}$

We will get back to the use of generalized cofactor later

Data Structure for SOP Manipulation

most of the following slides are by courtesy of Andreas Kuehlmann

Operation on Cube Lists

- AND operation:
  - take two lists of cubes
  - compute pair-wise AND between individual cubes and put result on new list
  - represent cubes in computer words
  - implement set operations as bit-vector operations

```
Algorithm AND(List_of_Cubes C1,List_of_Cubes C2) {
  C = ∅
  foreach c1 ∈ C1 {
    foreach c2 ∈ C2 {
      c = c1 & c2
      C = C ∪ c
    }
  }
  return C
}
```