Recognition of Fanout-free Functions

Tsung-Lin Lee
Chun-Yao Wang

Presenter: 黃明瀧

Outline

- Preliminaries
- The Old Algorithm
- The New Algorithm
- Complexity and Performance

Preliminaries

Task: Recognize Fanout-free Functions

- f is fanout-free

  $\Leftrightarrow \exists$ form where each variable appears exactly once

  - $f = x_1x_2 + x_1x_3$ is fanout-free

    $(f = x_1(x_2 + x_3))$

  - $f = x_1x_2 + x_2x_3 + x_3x_1$ is NOT fanout-free

- Task: Given f in SOP form

  1. Is f fanout-free?
  2. If true, find a fanout-free form of f

- Assume variables are positive

  If not, $f(x_i) = \overline{x_i} \cdots \Rightarrow g(\overline{x_i}) = x_i \ldots$

Adjacency

- x_i and x_j are **adjacent** in f

  $\Leftrightarrow f(x_i = a) = f(x_j = a)$, for some $a = 0$ or 1
- Notation: $x_i =_a x_j$
- Proven to be an equivalence relation

$$f = x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_4x_6 + x_5x_6$$

$$\implies x_1 =_0 x_2 =_0 x_3,\ x_4 =_1 x_5$$

$\implies$ adjacent classes: $\{x_1, x_2, x_3\}\{x_4, x_5\}\{x_6\}$

The Old Algorithm

JPH's Procedure

Theorem: Given the adjacency partition $\{X_1, \ldots, X_m\}$ for X

we can decompose f like this:

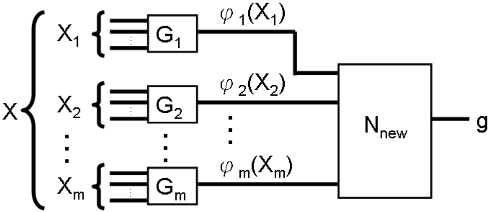

Figure 1: Decomp. with Adjacency Relation

The procedure recursively decompose f,

until g is **mutually adjacent**

JPH's Procedure (cont'd)

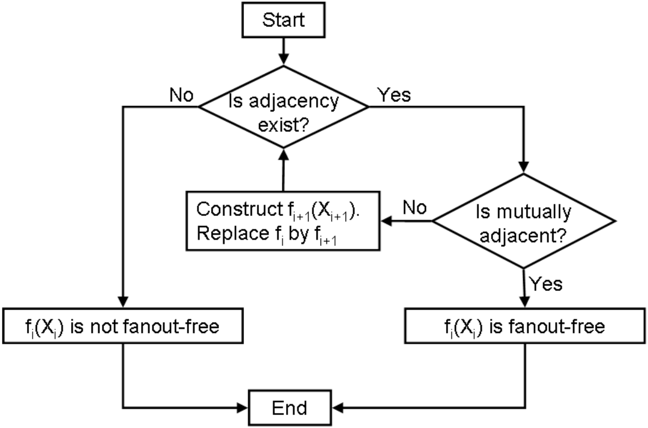

Figure 2: JPH's Procedure

Why is JPH's Procedure slow?

1. Recall $x_i =_a x_j$ iff the cofactors are the same.

   Requires **equivalence checking** of two functions.

2. The new function g is found by building **truth table**.

$\implies$ Exponential time.

The New Algorithm

Disappearance and Adjacency

Lemma 1

Adjacency $\implies$ disappearance

$$x_i =_a x_j \implies \begin{cases} x_j \text{ disappears in } f(x_i = a) \\ x_i \text{ disappears in } f(x_j = a) \end{cases}$$

Lemma 2

Disappearance $\implies$ adjacency

$$\begin{cases} x_j \text{ disappears in } f(x_i = a) \\ x_i \text{ disappears in } f(x_j = a) \end{cases} \implies x_i =_a x_j$$

Theorem

Adjacency $\Leftrightarrow$ disappearance

$\implies$ *Check disappearance (polynomial time) to check adjacency*

Example: Use disappearance to check adjacency

$$f = x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_4x_6 + x_5x_6$$

Compute cofactors.

$$\begin{cases} f(x_1 = 0) = x_4x_6 + x_5x_6 \\ f(x_2 = 0) = x_4x_6 + x_5x_6 \\ f(x_3 = 0) = x_4x_6 + x_5x_6 \\ f(x_4 = 0) = x_1x_2x_3x_5 + x_5x_6 \\ f(x_5 = 0) = x_1x_2x_3x_4 + x_4x_6 \\ f(x_6 = 0) = x_1x_2x_3x_4 + x_1x_2x_3x_5 \end{cases}$$

$$
\begin{cases}
f(x_1 = 0) = x_4 x_6 + x_5 x_6 \\
f(x_2 = 0) = x_4 x_6 + x_5 x_6 \\
f(x_3 = 0) = x_4 x_6 + x_5 x_6 \\
f(x_4 = 0) = x_1 x_2 x_3 x_5 + x_5 x_6 \\
f(x_5 = 0) = x_1 x_2 x_3 x_4 + x_4 x_6 \\
f(x_6 = 0) = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_5
\end{cases}
$$

$$
\begin{array}{lcccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
f(x_1 = 0) & \color{gray}{0} & \color{red}{0} & \color{red}{0} & 1 & 1 & 1 \\
f(x_2 = 0) & \color{red}{0} & \color{gray}{0} & 0 & 1 & 1 & 1 \\
f(x_3 = 0) & \color{red}{0} & 0 & \color{gray}{0} & 1 & 1 & 1 \\
f(x_4 = 0) & 1 & 1 & 1 & \color{gray}{0} & 1 & 1 \\
f(x_5 = 0) & 1 & 1 & 1 & 1 & \color{gray}{0} & 1 \\
f(x_6 = 0) & 1 & 1 & 1 & 1 & 1 & \color{gray}{0}
\end{array}
\implies x_1 =_0 x_2 =_0 x_3
$$

Example: Find new function g

Find all adjacency classes

Repeat for $f(x_i = 1)$, we find $x_4 =_1 x_5$

$\implies$ adjacent classes: $\{x_1, x_2, x_3\}\{x_4, x_5\}\{x_6\}$

$\implies \phi_1 = x_1 x_2 x_3$, $\phi_2 = (x_4 + x_5)$, $\phi_3 = x_6$

Deduce g

$f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_5 + x_4 x_6 + x_5 x_6$

$= g(\phi_1, \phi_2, \phi_3) = \phi_1 \phi_2 + \phi_2 \phi_3$

$= f(\phi_1, \phi_1, \phi_1, \phi_2, \phi_2, \phi_3) = \phi_1 \phi_1 \phi_1 \phi_2 + \phi_1 \phi_1 \phi_1 \phi_2 + \phi_2 \phi_3 + \phi_2 \phi_3$

(This is **not** a coincidence)

Example

After repeating the process (find adj. classes, deduce new function),

$h(\theta_1, \theta_2) = \theta_1\theta_2$

Only one adjacency class $\{\theta_1, \theta_2\} \implies f$ is fanout-less

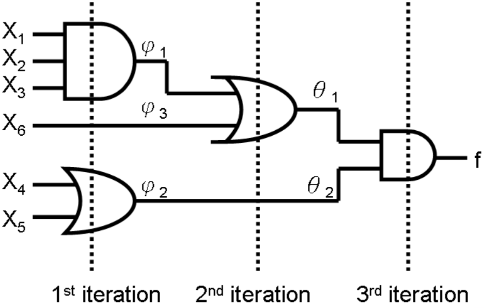

Figure 3: A fanout-less realization of f

Complexity and Performance

Complexity

N = number of variables, K = number of products (in SOP form)

- JPH's procedure: $O(N^2 2^N)$, due to equivalence check
- Proposed method: $O(N^2 K)$

Performance

Name	lits(sop)	#vars	CPU Time (s)		
			IROF	JPH	Ours
l2_b10	10240	20	0.30	4	0.11
l4_b3	3072	24	0.10	6	0.08
l4_b6	7290	24	0.21	277	0.18
l6_b4	672	20	0.02	3	0.02
l6_b8a	132	52	0.02	>1hr	0.29
l6_b8b	24192	52	0.74	>1hr	2.08
l8_b5	3380	29	0.09	>1hr	0.11
l10_b3	2160	30	0.07	>1hr	0.16
l14_b3	6720	42	0.20	>1hr	0.72

Figure 4: Experimental Results

References

- T. Lee and C. Wang, “Recognition of Fanout-free Functions,”