Recognition of Fanout-free Functions

Tsung-Lin Lee
Chun-Yao Wang

Presenter: 黃明瀧

Outline

- Preliminaries
- The Algorithm

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, because $x_1(x_2 + x_3)$
  - $f = x_1x_2 + \overline{x_1}x_3$ 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

Adjacency

- x_i and x_j are **adjacent**

  $\Leftrightarrow f(x_i = a) = f(x_j = a)$, for some $a = 0$ or 1
- i.e. $\exists$ the same cofactor for both vars
- Notation: $x_i =_a x_j$
- Proven to be an equivalence relation

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

$$\Longrightarrow \; x_1 =_0 x_2 =_0 x_3, \; x_4 =_1 x_5$$

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

The Old Algorithm

Simple Disjunctive Decomposition

$\{Y, Z\}$ must be a partition of X.

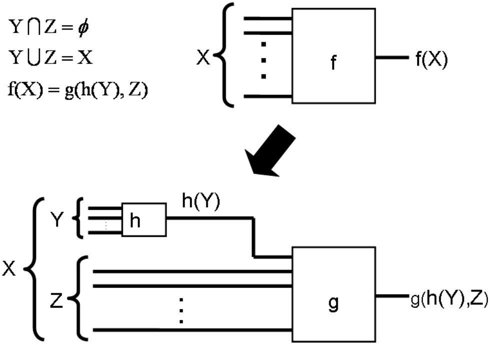

Figure 1: Simple Disjunctive Decomposition

JPH's Procedure

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

we can decompose f like this:

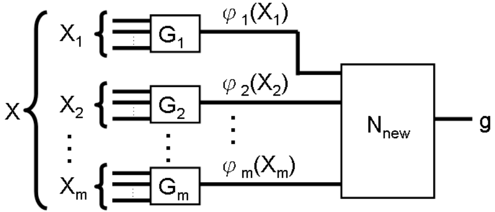

Figure 2: Decomp. with Adjacency Relation

The procedure recursively decompose f,

until g is **mutually adjacent**

JPH's Procedure (cont'd)

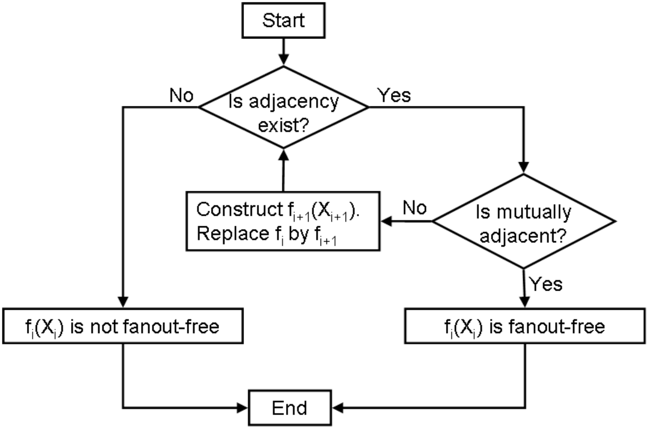

Figure 3: JPH's Procedure

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**.

$\Longrightarrow$ 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