DATA STRUCTURES FOR LOGIC OPTIMIZATION

©*Giovanni De Micheli*

Stanford University

Outline

- Review of Boolean algebra.
- Representations of logic functions.
- Matrix representations of covers.
- Operations on logic covers.

Background

- Function $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$.
- *Cofactor* of f with respect to variable x_i:
  - $f_{x_i} \equiv f(x_1, x_2, \ldots, 1, \ldots, x_n)$.
- *Cofactor* of f with respect to variable x'_i:
  - $f_{x'_i} \equiv f(x_1, x_2, \ldots, 0, \ldots, x_n)$.
- *Boole's expansion theorem*:
  - $f(x_1, x_2, \ldots, x_i, \ldots, x_n) = x_i \cdot f_{x_i} + x'_i \cdot f_{x'_i}$

Example

- Function: $f = ab + bc + ac$
- Cofactors:
  - $f_a = b + c$
  - $f_{a'} = bc$
- Expansion:
  - $f = af_a + a'f_{a'} = a(b + c) + a'bc$

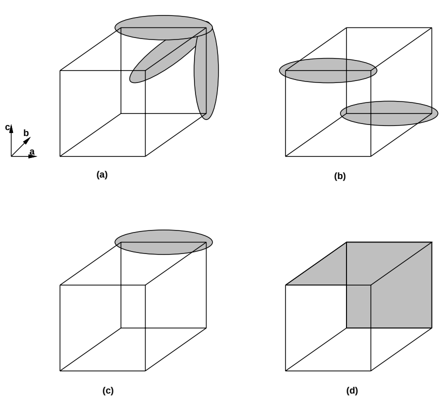
Background

- Function $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$.
- *Positive unate* in x_i when:
  - $f_{x_i} \geq f_{x'_i}$
- *Negative unate* in x_i when:
  - $f_{x_i} \leq f_{x'_i}$
- A function is positive/negative unate when positive/negative unate in all its variables.

Background

- Function $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$.
- *Boolean difference* of f w.r.t. variable x_i:
  - $\partial f / \partial x_i \;\equiv\; f_{x_i} \oplus f_{x'_i}$.
- *Consensus* of f w. r. to variable x_i:
  - $\mathcal{C}_{x_i} \;\equiv\; f_{x_i} \cdot f_{x'_i}$.
- *Smoothing* of f w. r. to variable x_i:
  - $\mathcal{S}_{x_i} \;\equiv\; f_{x_i} + f_{x'_i}$.

Example
$f = ab + bc + ac$



- The Boolean difference $\partial f / \partial a = f_a \oplus f_{a'} = b'c + bc'$.
- The consensus $\mathcal{C}_a = f_a \cdot f_{a'} = bc$.
- The smoothing $\mathcal{S}_a = f_a + f_{a'} = b + c$.

Generalized expansion

- Given:
  - A Boolean function f.
  - Orthonormal set of functions: $\phi_i, \quad i = 1, 2, \ldots, k$.
- Then:
  - $f \;=\; \sum_i^k \phi_i \cdot f_{\phi_i}$
  - Where f_{ϕ_i} is a *generalized cofactor*.
- The generalized cofactor is not unique, but satisfies:
  - $f \cdot \phi_i \subseteq f_{\phi_i} \subseteq f + \phi'_i$

Example

- Function $f = ab + bc + ac$.
- Basis: $\phi_1 = ab$ and $\phi_2 = a' + b'$.
- Bounds:
  - $ab \subseteq f_{\phi_1} \subseteq 1$
  - $a'bc + ab'c \subseteq f_{\phi_2} \subseteq ab + bc + ac$.
- Cofactors: $f_{\phi_1} = 1$ and $f_{\phi_2} = a'bc + ab'c$.

$$\begin{aligned} f &= \phi_1 f_{\phi_1} + \phi_2 f_{\phi_2} \\ &= ab1 + (a' + b')(a'bc + ab'c) \\ &= ab + bc + ac \end{aligned}$$

Generalized expansion theorem

- Given:
  - Two functions f and g.
  - Orthonormal set of functions: $\phi_i, \quad i = 1, 2, \dots, k$.
  - Boolean operator $\odot$.
- Then:
  - $f \odot g = \sum_i^k \phi_i \cdot (f_{\phi_i} \odot g_{\phi_i})$
- Corollary:
  - $f \odot g = x_i \cdot (f_{x_i} \odot g_{x_i}) + x_i' \cdot (f_{x_i'} \odot g_{x_i'})$

Matrix representations of logic covers

- Used in logic minimizers.
- Different formats.
- Usually one row per implicant.
- Symbols: 0,1,*. (and other)

The positional cube notation

- Encoding scheme:

$\emptyset$	00
0	10
1	01
$*$	11

- Operations:
  - Intersection – AND
  - Union – OR

Example
$f = a'd' + a'b + ab' + ac'd$

10	11	11	10
10	01	11	11
01	10	11	11
01	11	10	01

Cofactor computation

- Cofactor of α w.r. to β.
  - Void when α does not intersect β.
  - $a_1 + b'_1 \quad a_2 + b'_2 \quad \ldots \quad a_n + b'_n$
- Cofactor of a set $C = \{\gamma_i\}$ w.r. to β:
  - Set of cofactors of γ_i w.r. to β.

Example
$f = a'b' + ab$

10	10
01	01

- Cofactor w.r. to 01 11:
  - First row – void.
  - Second row – 11 01 .
- Cofactor $f_a = b$

Multiple-valued-input functions

- Input variables can have many values.
- Representations:
  - Literals: set of valid values.
  - Sum of products of literals.
- Extension of positional cube notation.
- Key fact:
  - *Multiple-output binary-valued functions represented as mvi single-output functions.*

Example

- 2-input, 3-output function:
  - $f_1 = a'b' + ab$
  - $f_2 = ab$
  - $f_3 = ab' + a'b$
- Mvi representation:

10	10	100
10	01	001
01	10	001
01	01	110

Operations on logic covers

- *Recursive paradigm*:
  - Expand about a mv-variable.
  - Apply operation to cofactors.
  - Merge results.
- *Unate heuristics*:
  - Operations on unate functions are simpler.
  - Select variables so that cofactors become unate functions.

Tautology

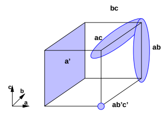
- Check if a function is always TRUE.
- Recursive paradigm:
  - Expand about a mv-variable.
  - If all cofactors are TRUE then function is a tautology.
- Unate heuristics:
  - If cofactors are unate functions additional criteria to determine tautology.
  - Faster decision.

Recursive tautology

- TAUTOLOGY: the cover has a row of all 1s. (Tautology cube).
- NO TAUT.: the cover has a column of 0s. (A variable that never takes a value).
- TAUTOLOGY: the cover depends on one variable, and there is no column of 0s in that field.
- When a cover is the union of two subcovers, that depend on disoint subsets of variables, then check tautology in both subcovers.

Example

$f = ab + ac + ab'c' + a'$



01	01	11
01	11	01
01	10	10
10	11	11

- Select variable a.
- Cofactor w.r.to a' is 11 11 11 – Tautology.
- Cofactor w.r.to a is:

11	01	11
11	11	01
11	10	10

Example

11	01	11
11	11	01
11	10	10

- Select variable b.
- Cofactor w.r.to b' is:

11	11	01
11	11	10

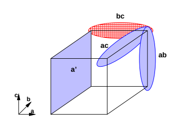
- No column of 0 – Tautology.
- Cofactor w.r.to b is: 11 11 11.
- *Function is a TAUTOLOGY.*

Containment

- Theorem:
  - *A cover F contains an implicant α iff F_α is a tautology.*
- Consequence:
  - Containment can be verified by the tautology algorithm.

Example

$f = ab + ac + a'$



- Check covering of bc – $C(bc) =$ 11 01 01.
- Take the cofactor:

01	11	11
01	11	11
10	11	11

- Tautology – bc is contained by f.

Complementation

- Recursive paradigm:
  - $f' = x \cdot f'_x + x' \cdot f'_{x'}$
- Steps:
  - Select variable.
  - Compute cofactors.
  - Complement cofactors.
- Recur until cofactors can be complemented in a straightforward way.

Termination rules

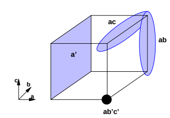
- The cover F is void.
  Hence its complement is the universal cube.
- The cover F has a row of 1s.
  Hence F is a tautology and its complement is void.
- The cover F consists of one implicant.
  Hence the complement is computed by De Morgan's law.
- All the implicants of F depend on a single variable, and there is not a column of 0s.
  The function is a tautology, and its complement is void.

Unate functions

- Theorem:
  - If f be positive unate: $f' = f'_x + x' \cdot f'_{x'}$.
  - If f be negative unate: $f' = x \cdot f'_x + f'_{x'}$.
- Consequence:
  - Complement computation is simpler.
  - One branch to follow in the recursion.
- Heuristic:
  - Select variables to make the cofactors unate.

Example

$f = ab + ac + a'$



- Select binate variable a'.
- Compute cofactors:
  - $F_{a'}$ is a tautology, hence $F'_{a'}$ is void.
  - F_a yields:

    11 01 11
    11 11 01

Example (2)

© GDM

- Select unate variable b.

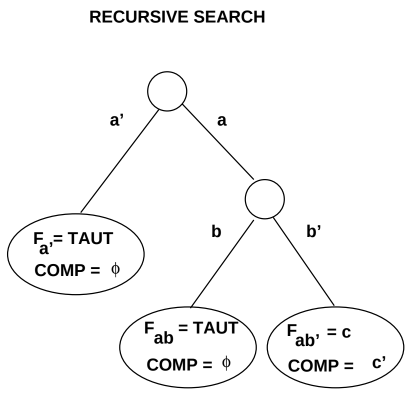
- Compute cofactors:
  - F_{ab} is a tautology, hence F'_{ab} is void.
  - $F_{ab'} =$ 11 11 01 and its complement is 11 11 10.

- Re-construct complement:
  - 11 11 10 intersected with $C(b') =$ 11 10 11 yields 11 10 10.
  - 11 10 10 intersected with $C(a) =$ 01 11 11 yields 01 10 10.

- Complement: $F' =$ 01 10 10.

Example (3)

© GDM



Summary

© GDM

- Matrix oriented representation:
  - Used in two-level logic minimizer.
  - May be wasteful of space (sparsity).
  - Good heuristics tied to this representation.

- Efficient Boolean manipulation exploits recursion.