Two-Level Logic Optimization
Heuristic Minimization using the Unate Recursive Paradigm

Priyank Kalla

Associate Professor
Electrical and Computer Engineering, University of Utah
kalla@ece.utah.edu
http://www.ece.utah.edu/~kalla
Two-Level Heuristic Minimization: Basic Ideas

- Generation of all primes can be infeasible
- Exact minimization might require a lot of work, large table covering problems, particularly for multi-output functions
- Heuristic minimization: Solve large problems quickly, maybe sub-optimally, but the solutions are quite close to optimal
- Espresso: a two-level logic minimizer
- Espresso: The quintessential case-study of CAD heuristics
- Think Primality & Irredundancy
  - Not every prime and irredundant cover is minimum, but the converse is true.
  - Search for prime and irredundant covers, with lower cost
  - Search should be fast, should hill climb, and be intelligent
The Basic Espresso Loop

Input: $F =$ ON-SET cover, $D =$ DC-SET cover

$F =$ Expand($F, D$);
$F =$ Irredundant($F, D$);

repeat
  
  cost = $|F|$;
  $F =$ Reduce($F, D$);
  $F =$ Expand($F, D$);
  $F =$ Irredundant($F, D$);

until $|F| < \text{cost}$;

$F =$ Make_Sparse($F$);
The Actual Espresso Algorithm

**Input:** $F =$ ON-SET cover, $D =$ DC-SET cover

$F =$ Expand($F, D$);

$F =$ Irredundant($F, D$);

$E =$ Essentials($F, D$);

$F =$ $F - E$;

repeat

  cost$_1 =$ |$F$|;

  repeat

    cost$_2 =$ |$F$|;

    $F =$ Reduce($F, D$);

    $F =$ Expand($F, D$);

    $F =$ Irredundant($F, D$);

  until |$F$| < cost$_2$;

  $F =$ last_gasp($F, D$);

until |$F$| < cost$_1$;

$F =$ Make_Sparse($F$);
The **Expand** operator

- Increase the size of each implicant, such that the smaller ones can be covered and dropped
- Maximally expanded implicants = primes
- IOW, **Expand** makes a cover prime and minimal w.r.t. **SCC**

**Approach:**

- Take a cube (e.g. $abc$), drop a literal (e.g. $ab$)
- Check if the expansion is valid. If valid, continue expansion.
- If invalid, **Expand** in another direction (e.g. $abc \rightarrow ac$)
How to Check if Expanded Cube is Valid?

Two ways:

- Is the Expanded cube $\alpha \subseteq (F \cup D)$? This is “containment check”!
  - Containment: $\alpha \in f \iff f_\alpha$ is Tautology
  - Another approach: containment: $\alpha \in f \iff (\bar{\alpha} + f)$ is Tautology

- Does the Expanded cube intersect with the OFF-set?
  - Requires OFF-set computation: $f' = x \cdot (f_x)' + x' \cdot (f_{x'})'$
  - Once again: use recursive paradigm for complement computation
Containment as Tautology Check: Implementation

Tautology Check using Shannon’s Expansion: \( f = xf_x + x'f_x' \)

- A cover \( f \) is **TAUTOLOGY** iff both cofactors are **TAUTOLOGY**
- Use the **Unate Recursive Paradigm**
  - Choice of splitting variable: pick the highest binate variable for expansion
  - Terminal cases of recursion?
    - When the cover of \( f \) is a single cube, \( f \neq 1 \)
    - When the cover of \( f \) is unate in (at least) one variable
      - Exploit unateness: A +ve unate \( f \) is **TAUTOLOGY** iff \( f_x' = 1 \)
      - Exploit unateness: A -ve unate \( f \) is **TAUTOLOGY** iff \( f_x = 1 \)
      - Exploit unateness: A unate \( f \) is **TAUTOLOGY** iff the contained cofactor is **TAUTOLOGY**

Example: \( f = ab + ac + ab'c' + a' \), is \( f = 1? \)
Example: \( f = ab + ac + a' \), apply \( \text{Expand}(f) \) operator.
Detect Essential Primes

**Theorem**

Let $F = G \cup \alpha$, where $\alpha$ is a prime disjoint from $G$. Then $\alpha$ is an essential prime iff $\text{CONSENSUS}(G, \alpha)$ does not cover $\alpha$.

- $G = \text{Remove from } F \text{ the minterms covered by } \alpha$
- $\alpha$ is NOT essential if it can be covered by other primes
- Some cubes in $G$ should be expandable to cover $\alpha$
- Analyze those cubes in $G$ that are distance 1 from $\alpha$
- Example: $f = a'b' + b'c + ac + ab$, is $\alpha = a'b'$ essential?
What is the Reduce Operator?

- Decrease the size of each implicant, so that successive expansion may lead to another cover of smaller cardinality.
- Reduced implicant’s validity — function should still be covered.
- Cardinality of $F$ should not increase.
- A redundant implicant be reduced to void!
- To reduce $\alpha$, remove from $F$ those minterms that are covered by $F - \{\alpha\}$.
- Can be done by $\alpha \cap (F - \{\alpha\})$?
- However, ensure that the result yields a single implicant, otherwise the cardinality of $F$ may increase!
  - Need to analyze the “supercube” of $(F - \{\alpha\})$.
  - Supercube of $(\alpha, \beta) = \text{smallest single cube containing both.}$
More on the Reduce Operation....

Example: \( f = c' + a'b' \). Draw the cover on a 3-D cube.

- Reduce \( \alpha = c' \), so \( F - \alpha = \beta = a'b' \)
- \( \overline{F} - \alpha = a + b \)
- Intersect: \( \alpha \cap (a + b) = ac' + bc' \). Supercube of \( ac', bc' = 1 \). So \( c' \cap 1 = c' \) implies no valid reduction!
- Now reduce \( \alpha = a'b' \). So, \( F - \alpha = \beta = c' \)
- Compute \( \overline{F} - \alpha = c \), and supercube of \( c = c \) itself!
- \( \alpha \cap c = a'b'c \), so the cube \( a'b' \) reduces to \( a'b'c \) without reducing the cardinality of \( F \). Reduced \( F = \{c', a'b'c\} \)
- Now this cover can be expanded in other directions for hill-climbing