Logic Synthesis and Verification

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Fall 2017

Two-Level Logic Minimization (2/2)

Reading:

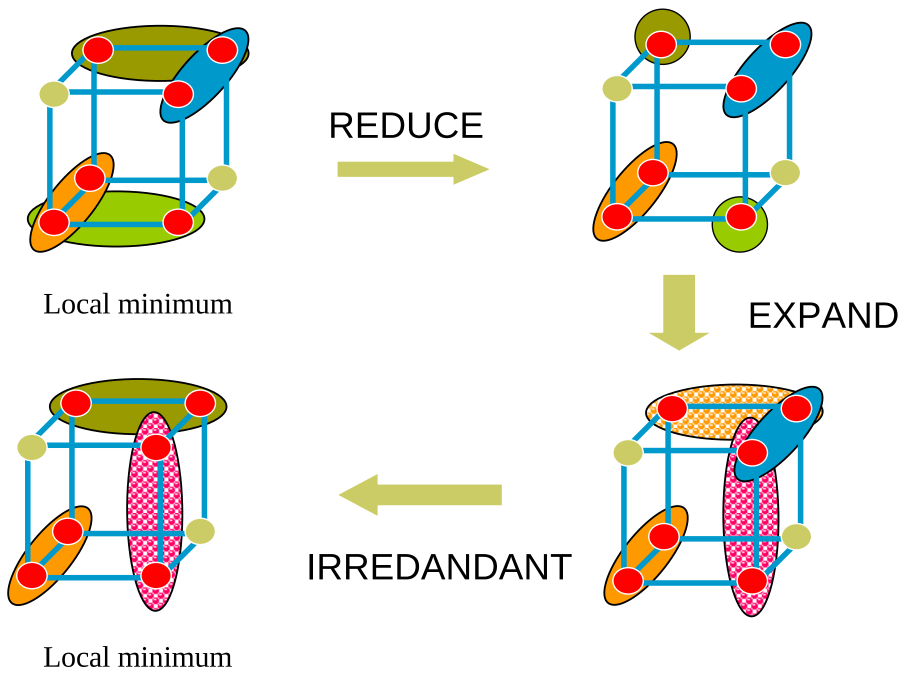
Logic Synthesis in a Nutshell

Section 3 (§3.1-§3.2)

most of the following slides are by courtesy of Andreas Kuehlmann

Heuristic Two-Level Logic Minimization ESPRESSO

- Illustration



Heuristic Two-Level Logic Minimization ESPRESSO

```
ESPRESSO($\Im$)
{
  $(F,D,R) \leftarrow$ DECODE($\Im$)
  $F \leftarrow$ EXPAND(F,R)
  $F \leftarrow$ IRREDUNDANT(F,D)
  $E \leftarrow$ ESSENTIAL_PRIMES(F,D)
  $F \leftarrow F-E$;  $D \leftarrow D+E$
  do{
    do{
      $F \leftarrow$ REDUCE(F,D)
      $F \leftarrow$ EXPAND(F,R)
      $F \leftarrow$ IRREDUNDANT(F,D)
    }while fewer terms in F
    //LASTGASP
    $G \leftarrow$ REDUCE_GASP(F,D)
    $G \leftarrow$ EXPAND(G,R)
    $F \leftarrow$ IRREDUNDANT(F+G,D)
    //LASTGASP
  }while fewer terms in F
  $F \leftarrow F+E$;  $D \leftarrow D-E$
  LOWER_OUTPUT(F,D)
  RAISE_INPUTS(F,R)
  error $\leftarrow (F_{old} \not\subset F)$ or $(F \not\subset F_{old} + D)$
  return (F,error)
}
```

ESPRESSO IRREDUNDANT

- **Problem:**

  Given a cover of cubes C for some incompletely specified function (f,d,r), find a minimum subset of cubes S $\subseteq$ C that is also a cover, i.e.

$$f \subseteq \sum_{c \in S} c \subseteq f + d$$

- **Idea 1:**

  We are going to create a function g(y) and a new set of variables y = {y_i}, one for each cube c_i. A minterm in the y-space will indicate a subset of the cubes {c_i}.

- **Example**

  y = (0,1,1,0,1,0), i.e. $y_1'y_2y_3y_4'y_5y_6'$, represents {c_2,c_3,c_5}

ESPRESSO IRREDUNDANT

- **Idea 2:**

  Create g(y) so that it is the function such that:

$$g(y^*) = 1 \quad \Leftrightarrow \quad \sum_{y^*_i = 1} c_i \quad \text{is a cover}$$

  i.e. g(y*) = 1 if and only if {c_i | y^*_i =1} is a cover.

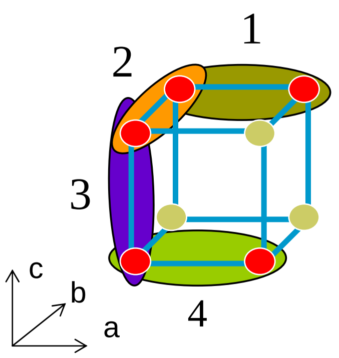
- Note: g(y) can be made positive unate (monotone increasing) in all its variables.

ESPRESSO IRREDUNDANT

- Example

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

$$g(y_1,y_2,y_3,y_4) = y_1 y_4 (y_2 + y_3)$$



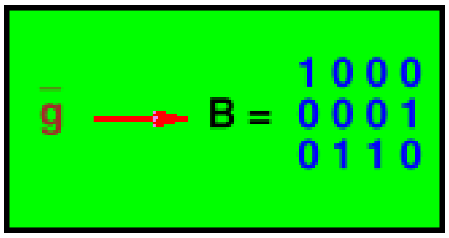
Note:

We want a minimum subset of cubes that covers f, that is, the largest prime of g (least literals).
Consider g': it is monotone decreasing in y (i.e. negative unate in y) e.g.

$$\bar{g}(y_1,y_2,y_3,y_4) = \bar{y}_1 + \bar{y}_4 + \bar{y}_2\bar{y}_3$$

ESPRESSO IRREDUNDANT

- Example
  - Create a Boolean matrix B for g':



$$\bar{g} \longrightarrow B = \begin{matrix}1&0&0&0\\0&0&0&1\\0&1&1&0\end{matrix}$$

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

$$\bar{g}(y_1,y_2,y_3,y_4) = \bar{y}_1 + \bar{y}_4 + \bar{y}_2\bar{y}_3$$

  - Recall a minimal column cover of B is a prime of $g = (g')'$
  - We want a *minimum* column cover of B
    - E.g., {1,2,4} ⇒ y_1 y_2 y_4 (cubes 1,2,4) ⇒ {bc, a'c, b'c'}

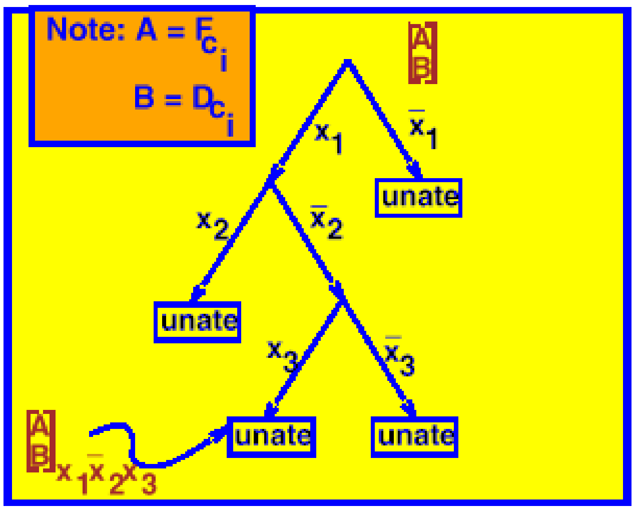
ESPRESSO IRREDUNDANT

- Deriving $g'(y)$
  - Modify tautology algorithm:

    F = cover of $\Im = (f, d, r)$

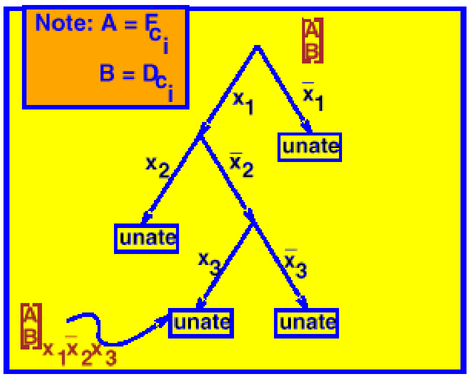
    D = cover of d
  - Pick a cube $c_i \in F$ (Note: $c_i \subseteq F \Leftrightarrow F_{c_i} \equiv 1$)
    - Do the following for each cube $c_i \subseteq F$:

$$\begin{bmatrix} A \\ B \end{bmatrix} \equiv \begin{bmatrix} F_{C_i} \\ D_{C_i} \end{bmatrix}$$



ESPRESSO IRREDUNDANT

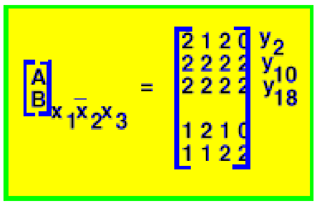
- Deriving $g'(y)$
  1. All leaves must be tautologies
  2. g' means how can we make it not a tautology
     - Must exactly delete all rows of all -'s that are not part of D
  3. Each row came from some row of A/B
  4. Each row of A is associated with some cube of F
  5. Each cube of B is associated with some cube of D
     - Don't need to know which, and cannot delete its rows
  6. Rows that must be deleted are written as a cube
     - E.g. $y_1 y_2 y_7 \Rightarrow$ delete rows 1,3,7 of F



ESPRESSO IRREDUNDANT

- Deriving $g'(y)$
  - Example

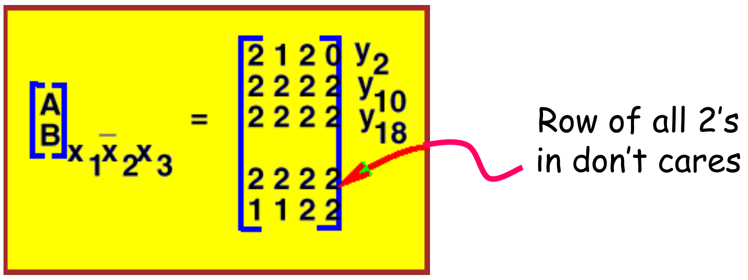
    Suppose unate leaf is in subspace $x_1x'_2x_3$: Thus we write down: $\bar{y}_{10}\,\bar{y}_{18}$ (actually, $\bar{y}_i$ must be one of $\bar{y}_{10}$, $\bar{y}_{18}$). Thus, F is not a cover if we leave out cubes c_{10} , c_{18}.



Unate leaf

Note:

If a row of all 2's is in don't cares, then there is no way not to have tautology at that leaf.

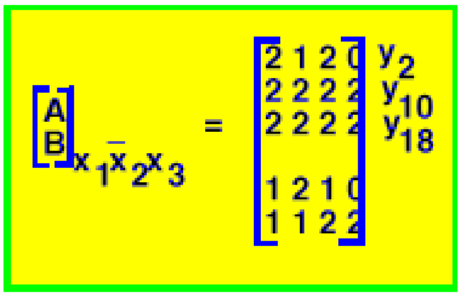


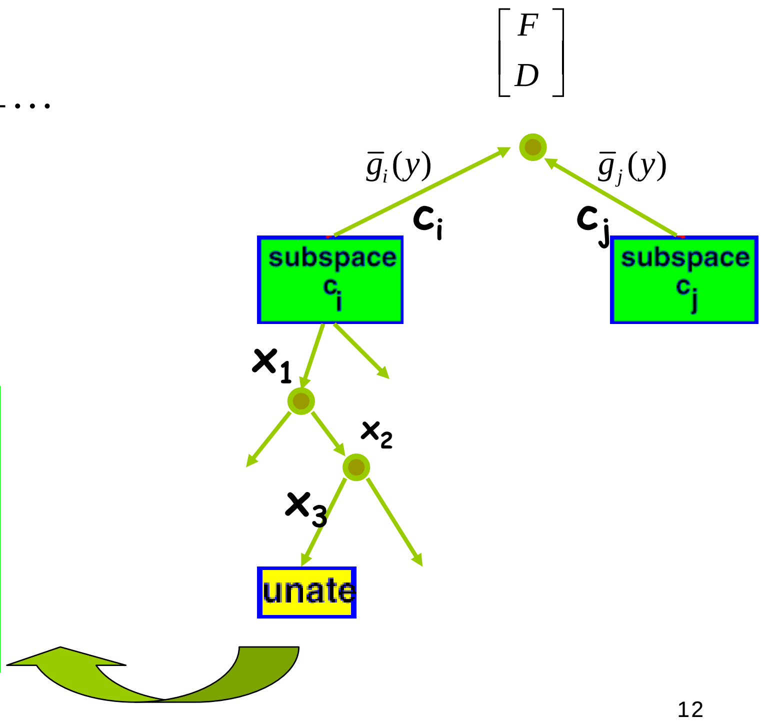
ESPRESSO IRREDUNDANT

- Deriving $g'(y)$

$$\bar{g}(y) = \bar{g}_i(y) + \bar{g}_j(y) + \cdots$$

$$\bar{g}_i(y) = \bar{y}_{10}\bar{y}_{18} + \cdots$$





ESPRESSO IRREDUNDANT

- Summary
  1. Convert $g'(y)$ into a Boolean matrix B
     - Note that $g(y)$ is unate
  2. Find a minimum column cover of B
     - E.g., if $y_1y_3y_{18}$ is a minimum column cover, then the set of cubes $\{c_1, c_3, c_{18}\}$ is a minimum sub-cover of $\{ c_i \mid i=1,\ldots,k \}$. (Recall that a minimal column cover of B is a prime of $g(y)$, and $g(y)$ gives all possible sub-covers of F).
  - Note: We are just doing tautology in constructing $g'(y)$, so unate reduction is applicable

$$F = \left[\begin{array}{c:c} A & C \\ \hdashline T & F^* \end{array}\right]$$

ESPRESSO IRREDUNDANT

- Summary
  - In Q-M, we want a maximum prime of $g(y)$

All primes

B = Minterms of f $\begin{bmatrix} 1011010 \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \end{bmatrix}$

$$B \cong \bar{g}\ (y) = \bar{y}_1\bar{y}_3\bar{y}_4\bar{y}_6 + \cdots$$

Note: A row of B says if we leave out primes $\{p_1, p_3, p_4, p_6\}$, then we cease to have a cover

  - So basically, the only difference between Q-M and IRREDUNDANT is that for the latter, we just constructed a $g'(y)$ where we did not consider all primes, but only those in some cover: F = $\{c_1, c_3, \ldots, c_k\}$

ESPRESSO
EXPAND

- $F \leftarrow$ EXPAND(F,R)
  - Problem: Take a cube c and make it prime by removing literals
  - Greedy way: (uses D and not R)
    - Remove literal l_i from c (results in, say c*)
    - Test if c* $\subseteq$ f+d (i.e. test if $(f+d)_{c^*} \equiv 1$)
    - Repeat, removing valid literals in order found
  - Better way: (uses R and not D)
    - Want to see all possible ways to remove maximal subset of literals
    - Idea: Create a function $g(y)$ such that $g(y)=1$ iff literals $\{l_i \mid y_i = 0\}$ can be removed (or $\{l_i \mid y_i = 1\}$ is a subset of literals such that if kept in c, will still make c* $\subseteq$ f+d, i.e. $c^* \wedge r \equiv 0$)

ESPRESSO
EXPAND

- Main idea

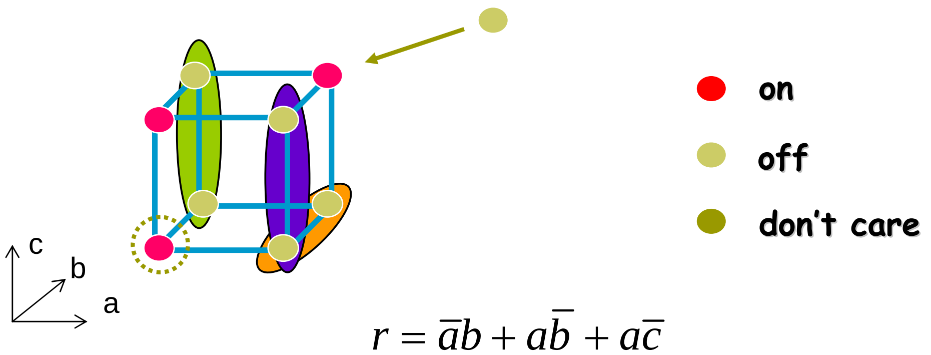
  Outline:

  1. Expand one cube, c_i , at a time
  2. Build “blocking” matrix $B = B^{c_i}$
  3. See which other cubes c_j can be feasibly covered using B
  4. Choose expansion (literals to be removed) to cover most other c_j

  Note:
  - $g(y)$ is monotone increasing
  - $B \cong \overline{g}(y)$ is easily built if we have R, a cover of r.
  - We do not need all of R. (reduced offset)

ESPRESSO EXPAND

- Reduced offset



$$r = \bar{a}b + a\bar{b} + a\bar{c}$$

Make *r* unate by adding (1,1,1) to offset. Then the new offset R_{new} = a + b $\cong$ g'(y). This is simpler and easier to deal with.

ESPRESSO EXPAND

- Blocking matrix B (for some cube *c*)
  - Given R = $\{r_i\}$, a cover of *r*. [$\Im$ = (*f*,*d*,*r*)]

$$B_{ij} = 1 \Leftrightarrow \begin{cases} l_j \in c \text{ and } \bar{l}_j \in r_i \\ \bar{l}_j \in c \text{ and } l_j \in r_i \end{cases}$$

B: rows indexed by offset cubes, columns indexed by literals of c

  - What does row *i* of B say?
    - It says that if literals $\{j \mid B_{ij} = 1\}$ are removed from *c*, then $c^* \wedge r_i \neq 0$, i.e., B_{ij} = 1 is one reason why *c* is orthogonal to offset cube r_i
    - Thus B $\rightarrow$ $g'(y) = y_1'y_3'y_{10}' + \cdots$ gives all ways that literals of *c* can be removed to get $c^* \not\subset f+d$ (i.e. $c^* \wedge r \neq 0$)

ESPRESSO EXPAND

- Example

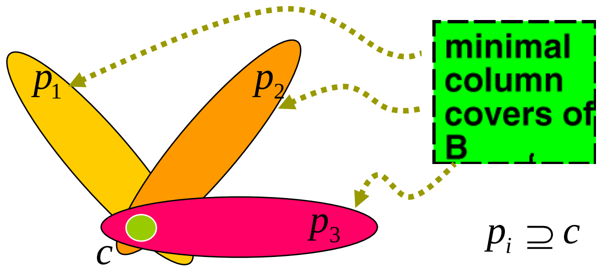
$$c = ab\bar{d}$$
$$r_i = \bar{a}bd\bar{e}$$
$$y_1 y_2 y_3 \propto a, b, \bar{d}$$

$$y_1 = 1 \Leftrightarrow \text{ keep } a$$
$$y_2 = 1 \Leftrightarrow \text{ keep } b$$
$$y_3 = 1 \Leftrightarrow \text{ keep } d$$
$$(B_i) = 101 = \bar{y}_1 \bar{y}_3 + \ldots = \bar{g}_i(y)$$

- Suppose g(y)=1
  - If y_1 = 1, we keep literal *a* in cube *c*.
  - B_i means do not keep literals 1 and 3 of c (implies that subsequent c^* is not an implicant)
    - If literals 1, 3 are removed we get $c \rightarrow c^* = b$. But $c^* \wedge r_i \neq 0$: $b \wedge a'bde' = a'bde' \neq 0$. So b is not an implicant.

ESPRESSO EXPAND

- Example (cont'd)
  - Thus all minimal column covers ($\cong g(y)$) of B are the minimal subsets of literals of *c* that must be kept to ensure that $c^* \subseteq f + d$ (i.e. $c^* \wedge r_i = 0$)
  - Thus each minimal column cover is a prime *p* that covers *c*, i.e. $p \supseteq c$



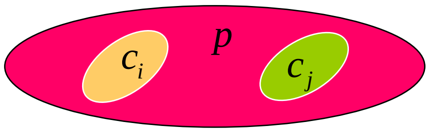
$$p_i \supseteq c$$

ESPRESSO EXPAND

- Expanding c_i

  F = { c_i }, $\mathfrak{I}$ = (f,d,r) $f \subseteq F \subseteq f+d$

  Q: Why do we want to expand c_i ?
  A: To cover some other c_j's



  Q: Can we cover c_j ?
  A: If and only if (SCC = "smallest cube containing" also called "supercube")

  equivalent to: $SCC\left(c_i \cup c_j\right) \subseteq f+d$

  equivalent to: $SCC\left(c_i \cup c_j\right) \wedge r = 0$

  literals "conflicting" between c_i, c_j can be removed and still have an implicant

ESPRESSO EXPAND

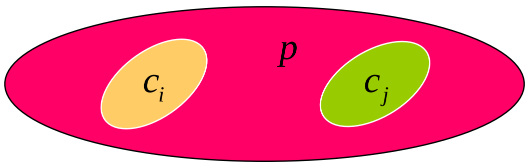
- Expanding c_i

  Can check SCC(c_i, c_j) with blocking matrix:

  c_i = 12012
  c_j = 12120

  implies that literals 3 and 4 must be removed for c_i^* to cover c_j

  Check if columns 3, 4 of B can be removed without causing a row of all 0's



ESPRESSO EXPAND

- Covering function
  - The objective of EXPAND is to expand c_i to cover as many cubes c_j as possible. The blocking function g’(y)=1 whenever the subset of literals $\{l_i \mid y_i = 1\}$ yields a cube $c^* \not\subset f + d$.
    - Note: $c^* = \prod_{(y_j=1)} l_j$
  - We now build the covering function *h*, such that:

    h(y) = 1, whenever the cube $c^* \supseteq c_i$ covers another cube $c_j \subseteq$ F
    - Note: *h(y)* is easy to build
    - Thus a minterm *m* of $g(y) \wedge h(y)$ is such that it gives $c^* \subseteq f + d$ (*g(m)* =1) and covers at least one cube (*h(m)* =1). We seek *m* which results in the most cubes covered.

ESPRESSO EXPAND

- Covering function

  Define h(y) by a set of cubes where d_k = kth cube is:

$$d_k = \varnothing \ \text{ if } \ SCC[c_i \cup c_k] \not\subset f + d \ \text{ else}$$

$$d_k^j = \begin{cases} \overline{y}_j & \text{if } c_k^j \not\subset c_i^j \text{ i.e. } \begin{cases} 2 \not\subset 1 \\ 2 \not\subset 0 \\ 0 \not\subset 1 \\ 1 \not\subset 0 \end{cases} \\ 2 & \text{otherwise} \end{cases}$$

d_k^j: jth literal of kth cube

Every d_k indicates the minimal expansion to cover c_k, that is, which literals that we have to leave out to minimally cover c_k. Essentially $d_k \neq \varnothing$ if cube c_k can be feasibly covered by expanding cube c_i.

Note that $h(y) = d_1 + d_2 + \cdots + d_{|F|-1}$ (one for each cube of F, except c_i) is monotone decreasing.

ESPRESSO EXPAND

- Covering function
  - We want a minterm m of $g(y) \wedge h(y)$ contained in a maximum number of d_k's
  - In Espresso, we build a Boolean covering matrix C (note that h(y) is negative unate) representing h(y) and solve this problem with greedy heuristics

Note:

$$B \cong \overline{g}(y)$$

but $C \cong \tilde{h}(y) \supseteq h(y)$

$$C = \left\{ \begin{array}{c} \cdots \\ 010110 \\ 101011 \\ 100101 \\ \cdots \end{array} \right\} \quad B = \left\{ \begin{array}{c} \cdots \\ 110110 \\ 101010 \\ 101001 \\ \cdots \end{array} \right\}$$

$\tilde{h}(y)$ is an over-approximation of $h(y)$, e.g., by removing the $d_k = \varnothing$ rule in the previous slide

ESPRESSO EXPAND

- Covering function

$$C = \left\{ \begin{array}{c} \cdots \\ 010110 \\ 101011 \\ 100101 \\ \cdots \end{array} \right\} \quad B = \left\{ \begin{array}{c} \cdots \\ 110110 \\ 101010 \\ 101001 \\ \cdots \end{array} \right\}$$

  - Want a set of columns such that if eliminated from B and C results in no empty rows of B and a maximum of empty rows in C
  - Note: A “1” in C can be interpreted as a reason why c* does not cover c_j

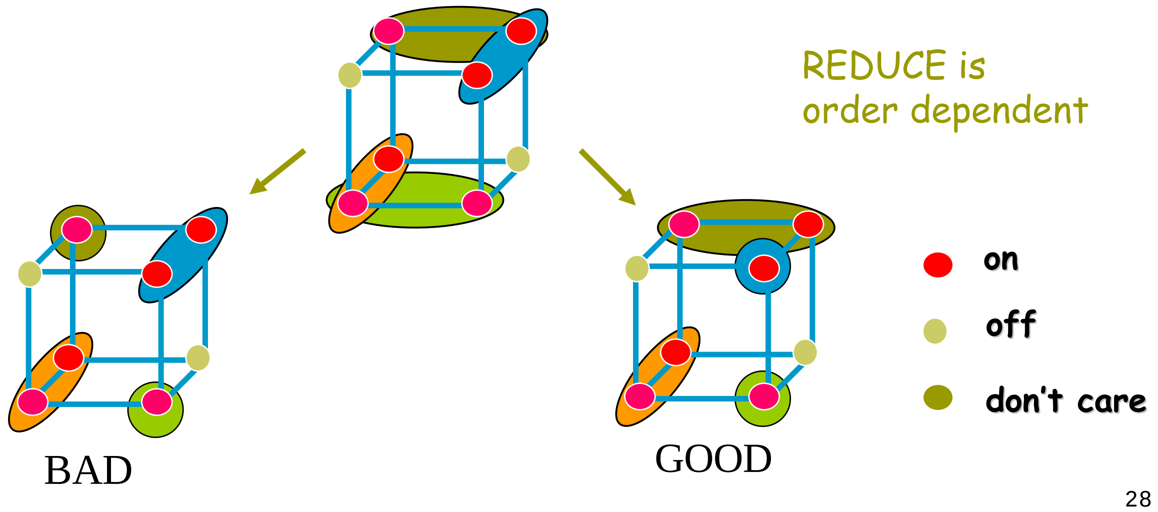
ESPRESSO EXPAND

- Endgame
  - What do we do if $h(y) \equiv 0$?
    - This could be important in many hard problems, since it is often the case that $h(y) \equiv 0$
  - Some things to try:
    - Generate largest prime covering c_i
    - Generate largest prime covering cover most care points of another cube c_k
    - Coordinate two or more cube expansions, i.e. try to cover another cube by a combination of several other cube expansions

ESPRESSO REDUCE

- Problem:

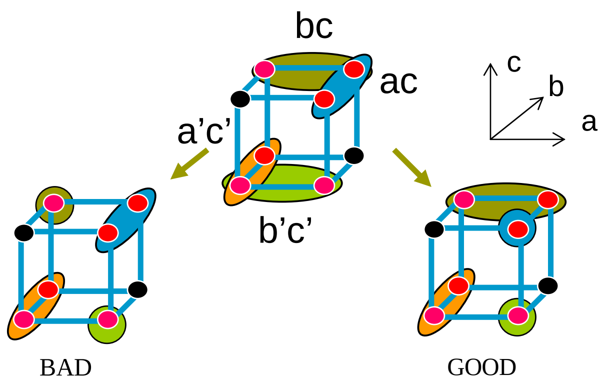
  Given a cover F and $c \in F$, find the smallest cube $\underline{c} \subseteq c$ such that $F \backslash \{ c \} + \{ \underline{c} \}$ is still a cover
  - $\underline{c}$ is called the maximally reduced cube of c



ESPRESSO
REDUCE

- Example

$$F = ac + bc + \bar{b}\bar{c} + \bar{a}\bar{c}$$



Two orders:

1. $\mathrm{REDUCE}\left(F = \{ac, bc, \bar{b}\bar{c}, \bar{a}\bar{c}\}\right) = a\bar{b}c + bc + a\bar{b}\bar{c} + \bar{a}\bar{c}$
2. $\mathrm{REDUCE}\left(F = \{bc, \bar{b}\bar{c}, ac, \bar{a}\bar{c}\}\right) = \bar{a}bc + ac + a\bar{b}\bar{c} + \bar{a}\bar{c}$

- REDUCE is order dependent !

ESPRESSO
REDUCE

```
Algorithm REDUCE(F,D){
  F ← ORDER(F)
  for(1 ≤ j ≤ |F|){
      $\underline{c}_j$ ← MAX_REDUCE(c,F,D)
      F ← (F∪{$\underline{c}_j$})\{$c_j$}
  }
  return F
}
```

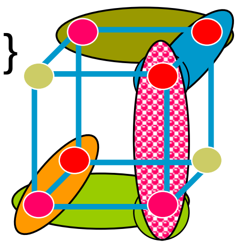
ESPRESSO REDUCE

- Main Idea: Make a prime not a prime but still maintain cover:

  $\{c_1, \ldots, c_i, \ldots, c_k\} \rightarrow \{c_1, \ldots, c_{i-1}, \underline{c}_i, c_{i+1}, \ldots, c_k\}$

  But

$$f \subseteq \sum_{j=0}^{i-1} c_j + \underline{c}_i + \sum_{j=i+1}^{k} c_j \subseteq f + d$$



  - To get out of a local minimum (prime and irredundant is local minimum)
  - After reduce, have non-primes and can expand again in different directions
    - Since EXPAND is “smart”, it may know best direction

ESPRESSO REDUCE

$F = \{c_1, c_2, \ldots, c_k\}$, D = $\{d_1, \ldots, d_m\}$
(F and D are covers of an incompletely specified function and a completely specified function, respectively.)

$F(i) = (F + D) \setminus \{ c_i \}$
$= \{c_1, c_2, \ldots, c_{i-1}, c_{i+1}, \ldots, c_k, d_1, \ldots, d_m\}$

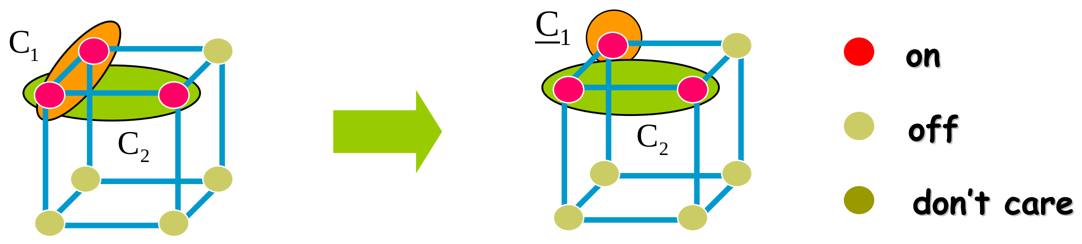
- Reduced cube:

  $\underline{c}_i$ = smallest cube containing ($c_i \cap \overline{F}(i)$)

  - Note that $c_i \cap \overline{F}(i)$ is the set of points uniquely covered by c_i (and not by any other c_j or D).
  - Thus, $\underline{c}_i$ is the smallest cube containing the minterms of c_i which are not in F(i).

ESPRESSO REDUCE

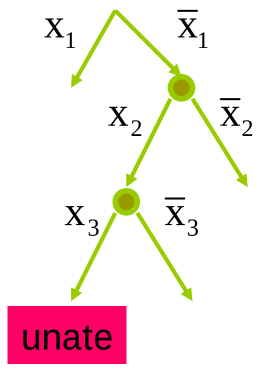
- SCC: “smallest cube containing”, i.e., supercube
- SCCC: “smallest cube containing complement”



$$
\begin{aligned}
\underline{c}_i &= SCC\left(c_i \cap \overline{F(i)}\right)\\
&= SCC\left(c_i \overline{F_{c_i}(i)}\right)\\
&= c_i SCC\left(\overline{F_{c_i}(i)}\right)\\
&= c_i SCCC\left(F_{c_i}(i)\right)
\end{aligned}
$$

ESPRESSO REDUCE

- SCCC computation
  - Unate recursive paradigm
    - Select most binate variable
    - Cofactor until unate leaf



What is SCCC (unate cover) ?

- Note that for a cube c with at least 2 literals, SCCC(c) is the universe:

$$\text{cube} = 01222 \quad \longrightarrow \quad \overline{\text{cube}} = \begin{matrix}12222\\20222\end{matrix}$$

Hence, SCCC(cube) = 22222

- Implies only need to look at 1-literal cubes

ESPRESSO REDUCE

- SCCC computation
  - SCCC(U) = γ for a unate cover U

Claim

- If unate cover has row of all 2’s except one 0, then complement is in x_i , i.e. γ_i = 1
- If unate cover has row of all 2’s except one 1, then complement is in x_i', i.e. γ_i = 0
- Otherwise, in both subspaces, i.e. γ_i = 2

Finally

$$SCCC\left(c_1 + c_2 + \ldots + c_k\right) = SCC\left(\overline{c}_1\overline{c}_2\ldots\overline{c}_k\right)$$
$$= SCC\left(\overline{c}_1\right)\cap\ldots\cap SCC\left(\overline{c}_k\right)$$

ESPRESSO REDUCE

- SCCC computation

Example 1: $f = a + bc + \overline{d} \Rightarrow \overline{f} = \overline{a}(\overline{b} + \overline{c})d \subseteq \overline{a}d$

- Note: 0101 and 0001 are both in $\overline{f}$. So SCCC could not have literal b or $\overline{b}$.

Example 2:

$$U(unate) = \begin{array}{ccccc} 2 & 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 1 & 0 \\ \uparrow & & & & \uparrow \end{array}$$

- Note that columns 1 and 5 are essential: they must be in every minimal cover. So $\neg U = x_1x_5(\ldots)$. Hence SCCC(U) = x_1x_5

ESPRESSO
REDUCE

- SCCC computation
  Example 2 (cont'd):

$$U = \overline{x}_1 + \overline{x}_5 + x_2(x_3 + x_4)$$
$$\overline{U} = x_1 x_5 (\overline{x}_2 + \overline{x}_3\overline{x}_4)$$
$$\begin{array}{lccccc} & 1 & 0 & 2 & 2 & 1 \\ \overline{U}(unate) = & 1 & 2 & 0 & 0 & 1 \subseteq 12221 \\ & & \uparrow & \uparrow & \uparrow & \end{array}$$

$$\text{minterms of } \overline{U} = \begin{array}{ccccc} 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{array}$$

The marked columns contain both 0's and 1's. But every prime of $\overline{U}$ contains literals x_1 , x_5

ESPRESSO
REDUCE

- SCCC computation
  - At unate leaves

$$n = \text{SCCC(unate)} = \emptyset \quad \text{if row of all 2's}$$

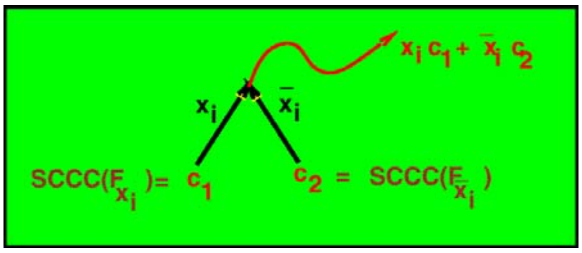
$$n_j = \begin{cases} x_j & \text{if column } j \text{ has a row singleton with a 0 in it} \\ \overline{x}_j & \text{if column } j \text{ has a row singleton with a 1 in it} \\ 2 & \text{otherwise} \end{cases}$$

- Hence unate leaf is easy !

ESPRESSO REDUCE

- SCCC computation
  - Merging
    - We need to produce $SCCC(f) = SCC(x_i c_1 + \bar{x}_i c_2) = \gamma$

$$\gamma = l_1 l_2 \ldots l_k$$
$$x_i \in \gamma \Leftrightarrow c_2 = \emptyset$$
$$\bar{x}_i \in \gamma \Leftrightarrow c_1 = \emptyset$$
$$l_{j \neq i} \in \gamma \Leftrightarrow (l_j \in c_1) \wedge (l_j \in c_2)$$



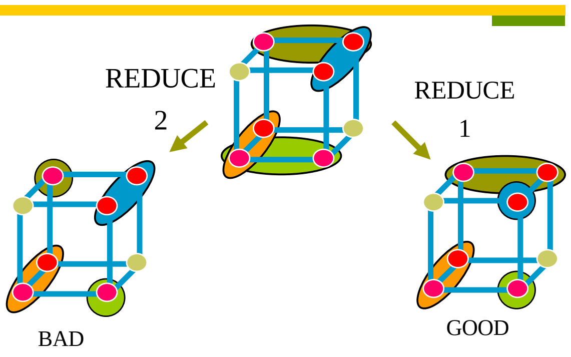
- If $c_1 \wedge c_2 \neq \emptyset$, then $\gamma_i = 2$
  - because minterms with x_i and $\neg x_i$ literals both exist, and thus $(SCC(x_i c_1 + x_i c_2))_i = 2$
- If $l_j \notin c_1$ or $l_j \notin c_2$, then $\gamma_j = 2$ (where $l_j = x_j$ or $\neg x_j$)
  - because minterms with x_j and $\neg x_j$ literals both exist
- If $l_j \in c_1$ and $\neg l_j \in c_2$, then $\gamma_j = 2$.

ESPRESSO

```
ESPRESSO(ℑ)
{
    (F,D,R) ← DECODE(ℑ)
    F ← EXPAND(F,R)
    F ← IRREDUNDANT(F,D)
    E ← ESSENTIAL_PRIMES(F,D)
    F ← F-E;  D ← D + E
    do{
        do{
            F ← REDUCE(F,D)
            F ← EXPAND(F,R)
            F ← IRREDUNDANT(F,D)
        }while fewer terms in F
        //LASTGASP
        G ← REDUCE_GASP(F,D)
        G ← EXPAND(G,R)
        F ← IRREDUNDANT(F + G,D)
        //LASTGASP
    }while fewer terms in F
    F ← F + E;  D ← D-E
    LOWER_OUTPUT(F,D)
    RAISE_INPUTS(F,R)
    error ← (F_old ⊄ F) or (F ⊄ F_old + D)
    return (F,error)
}
```

ESPRESSO LASTGASP

- Reduce is order dependent:
  E.g., expand can't do anything with that produced by REDUCE 2.



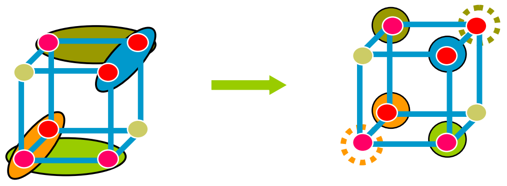
- Maximal Reduce:

$$\underline{c}_i^M = SCC\left(c_i \cap \overline{F(i)}\right) = c_i \cap SCCC\left(F(i)_{c_i}\right) \quad \forall i$$

  i.e., we reduce all cubes as if each were the first one.

  Note:

  $\{\underline{c}_1^M, \underline{c}_2^M, \ldots\}$ is not a cover



ESPRESSO LASTGASP

- Now EXPAND, but try to cover only $\underline{c}_j^M$'s.
  - We call EXPAND(G,R), where G = $\{\underline{c}_1^M, \underline{c}_2^M, \ldots, \underline{c}_k^M\}$
  - If a covering is possible, take the resulting prime:

$$f + d \supseteq p_i \supseteq \underline{c}_i^M \cup \underline{c}_j^M$$

  and add to F:

$$\tilde{F} = F \cup \{p_i\}$$

  Since F is a cover, so is $\tilde{F}$. Now apply IRREDUNDANT on $\tilde{F}$.

What about "supergasp" ?

Main Idea: Generally, think of ways to throw in a few more primes and then use IRREDUNDANT. If all primes generated, then just Quine-McCluskey