CSEE 6861 **Handout of Examples**

Prof. Steven Nowick

This handout gives detailed examples to illustrate some of the basic routines used in *espresso*, including: cofactor, tautology, complementation, and containment (application = identifying essential primes).

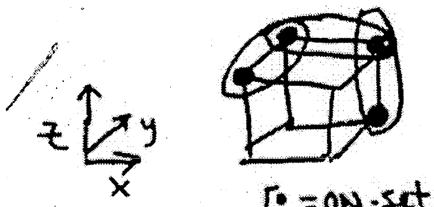
1. COFACTOR

Intuitively, the **cofactor of a cover** is the *restriction* of the cover to a smaller region of interest. The cofactor of a cover with respect to a literal, can be thought of as only considering the cover in one half of the original hypercube (where the literal is true). The cofactor of a cover with respect to a cube, can be thought of as only considering the cover in the region bounded by the cube.

Cofactoring is useful in splitting a complicated cover into smaller pieces, where results on the smaller pieces are easier to obtain (*e.g.* complement, primes, etc.). These small results can then be reassembled into a single final result.

Cofactor with Respect to a Literal.

Consider the following function f and cover F. There are 3 common representations: (i) *hypercube representation*, (ii) algebraic representation, and (iii) *PLA representation.*



[• = ON-set, all other points are in OFF-set here]

hypercube representation

cover $F = \bar{x}z + yz + xy$

algebraic representation

$$\begin{array}{c|ccc|c} & x & y & z & f \\ \hline 1 & 0 & - & 1 & 1 \\ 2 & - & 1 & 1 & 1 \\ 3 & 1 & 1 & - & 1 \end{array}$$

PLA representation

Problem: Find the cofactor of cover F with respect to literal x (called F_x).

Solution:

The cofactor can be computed directly, given any of the 3 representations. First, consider the **PLA representation.** A PLA representation of literal x is:

$$x = \left[\begin{array}{ccc|c} x & y & z & f \\ \hline 1 & - & - & 1 \end{array}\right]$$

To obtain F_x, do *row deletion* and *column deletion.*

Row Deletion Rule: For each row r in F:

(i) if r disagrees with x, *delete row* r;

(ii) if r agrees with x, *keep row* r.

A row r *disagrees* with a literal, if in some column, r has a 1 and the literal has a 0, or r has a 0 and the literal has a 1. In the above example, row 1 disagrees with literal x. Rows 2 and 3 agree with literal x. Result: *delete row 1.*

Column Deletion Rule: Delete every column where x has a 1 (or 0).

In this example, column x is deleted. The final PLA representation of F_x is:

$$F_x = \begin{array}{c|ccc|c} & \cancel{x} & y & z & f \\ \hline 1 & \cancel{0} & \cancel{-} & \cancel{1} & \cancel{1} \\ 2 & \cancel{-} & 1 & 1 & 1 \\ 3 & \cancel{1} & 1 & - & 1 \end{array} \Rightarrow F_x = \begin{array}{rc|cc|c} & & y & z & f \\ \hline yz \rightarrow & 2 & 1 & 1 & 1 \\ y \rightarrow & 3 & 1 & - & 1 \end{array}$$

Next, we show how the cofactor can be computed **algebraically**. The initial cover $F = x'z+yz+xy$. To compute F_x, cofactor each cube of F with respect to literal x. The cofactor of $x'z$ with respect to x is empty, since they disagree in x. (This corresponds to deleting a row of the PLA.) The reason that this cofactor is empty is that *$x'z$ does not intersect x, i.e.,* cube $x'z$ does not intersect the $x = 1$ half of the hypercube. For the remaining 2 cubes: yz cofactored with respect to x is yz (they agree on x), and xy cofactored with respect to x is y (they agree on x; delete x).

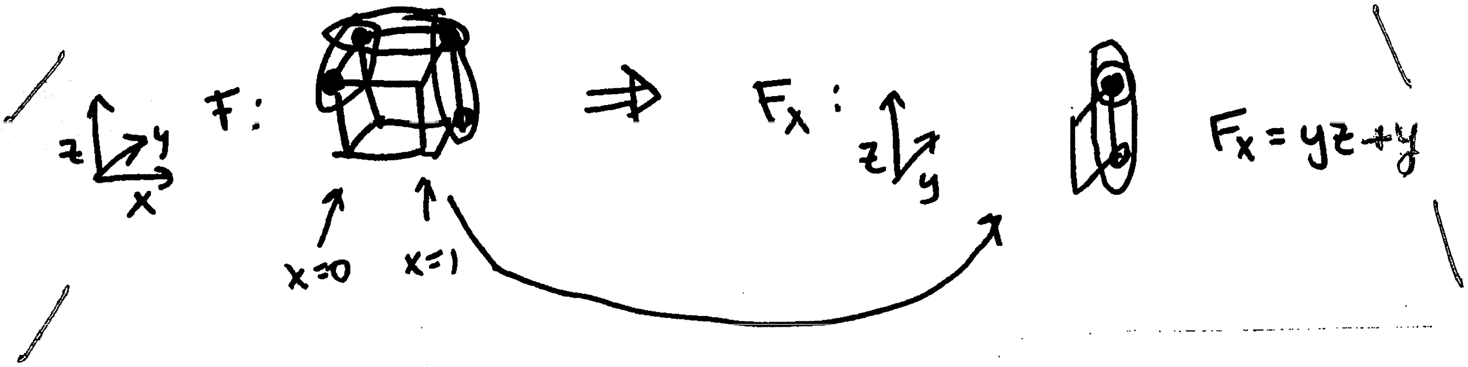
That is, given cover $F = x'z + yz + xy$, and literal x, F_x can be computed algebraically as follows. For each cube in F:

(i) if the cube disagrees with x, *delete the cube*;

(ii) if the cube agrees with x, *keep the cube*; *delete any* x which appears in it.

} result: $yz + y = F_x$

Finally, an equivalent **hypercube representation** of F_x is shown below. It is simply formed by *restricting* the original hypercube to the $x = 1$ plane. It describes a function of only 2 variables: y and z (x does not appear). The cofactored cover is $F_x = yz + y$. Intuitively, F_x describes the portion of cover F which intersects the $x = 1$ plane of the hypercube.



Cofactor with Respect to a Cube.

The cofactor of a cover with respect to a cube, is a simple generalization of 1(a). Consider the same f and F, described above.

Problem: Find the cofactor of F with respect to cube $x'y'$ (called $F_{x'y'}$).

Solution:

A PLA representation of cube $x'y'$ is:

$$x'y' = \left[\begin{array}{ccc|c} x & y & z & f \\ \hline 0 & 0 & - & 1 \end{array}\right]$$

The PLA representation of F (as before):

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \left[\begin{array}{ccc|c} x & y & z & f \\ \hline 0 & - & 1 & 1 \\ - & 1 & 1 & 1 \\ 1 & 1 & - & 1 \end{array}\right]$$

To compute $F_{x'y'}$, use a simple generalization of the rules in 1(a).

Row Deletion Rule: For each row r in F:

(i) if r disagrees with cube $x'y'$, *delete row* r;

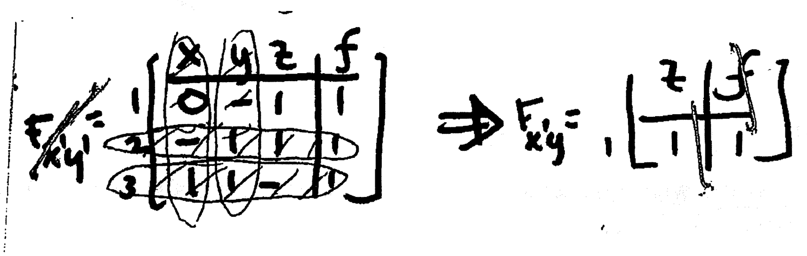
(ii) if r agrees with cube $x'y'$, *keep row* r.

A row r *disagrees* with a cube, if in some column, r has a 1 and the cube has a 0, or r has a 0 and the cube has a 1.

In this example, row 1 agrees with cube $x'y'$. Rows 2 and 3 disagree with cube $x'y'$. Therefore, *delete rows 2 and 3.*

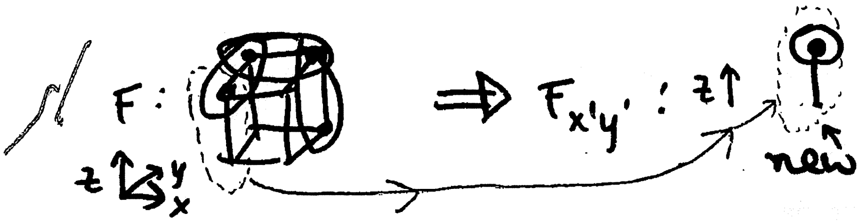
Column Deletion Rule: Delete every column where the cube ($x'y'$) has a 1 or 0.

In this example, columns x and y are deleted. Result:



As before, the same result can be computed **algebraically**. Given the cover $F = x'z + yz + xy$, the goal is to find $F_{x'y'}$. This is again a simple generalization of 1(a). First, delete yz and xy, since they disagree with cube $x'y'$. (yz disagrees with cube $x'y'$ in variable y; xy disagrees with cube $x'y'$ in variables x and y.) Keep $x'z$, since it agrees with cube $x'y'$. Next, delete variable x from the remaining cubes (*i.e.,* from $x'z$). The result is: $F_{x'y'} = z$.

Finally, the same result can be constructed directly in the **hypercube**. Here, $F_{x'y'}$ is the *restriction of F to the cube* $x'y'$. To obtain $F_{x'y'}$, simply restrict F to the portion of the original hypercube bounded by $x'y'$. The result is a 1-dimensional hypercube, where z is the only input variable. That is, the original hypercube has been *restricted* to the region of $x'y'$. Only products in F which intersect $x'y'$ appear in the new hypercube. In this example, only $x'z$ intersects $x'y'$, so it appears in the final hypercube. In the final hypercube, $x'z$ simply becomes z, since there are no x and y variables.
[*Note:* For extensions of cofactor to *multi-outputfunctions,* see the Hachtel/Somenzi ch. 5 assigned reading.]



the cover is $F_{x'y'} = z$.

new hypercube (a function only of z)

2. TAUTOLOGY

The algorithm for tautology checking is in the Tautology Handout. For examples, see Hachtel/Somenzi chapter 5 (including some solved problems in the exercise section at the end of the chapter). An example is worked out below.

3. COMPLEMENTATION

See Complementation Handout. An example is included below.

4. CONTAINMENT

A common problem in espresso algorithms is the **containment problem**: Given a cube c and a cover C, does C cover cube c? (see Hachtel/Somenzi ch. 5). (Usually, c is not a cube in cover C.) That is, do the cubes of C completely cover, or overlap, the region of c?

The containment problem arises in many algorithms in espresso, such as *irredundant* and *essentials*.

A key insight is that **the containment problem is equivalent to a tautology problem.** Intuitively, the containment problem asks if C *covers every minterm in cube c*. This is equivalent to asking: is C, *"restricted"* to the region of cube c, all 1's? Recall that the restriction of a cover to a region, is a *cofactor* operation. Therefore, more formally, this problem is identical to a tautology check on C_c (the cofactor of cover C with respect to cube c): **is C_c a tautology?**

This problem is solved in two steps. First, the cofactor C_c of cover C with respect to cube c is computed, using the procedure described in Section 1 above. Second, one checks if C_c is a tautology, using the procedure in the Tautology Handout.
(See the next section for an application and example.)

5. CONTAINMENT: 1st Application – Identifying essential primes, given the set of all primes

An important application of containment is to identify essential prime implicants. This 1st application of "containment" is a simple one: given the set of *all prime implicants* of a Boolean function, identify which ones are *essential prime implicants.*

Below is a 4-variable Karnaugh map of a function f, with inputs w, x, y, and z. The set P of all prime implicants is shown. The formal notation is as follows:

$$E = \{c \in P : c \nleq (P - \{c\})\}$$

In words, "the set E of essentials is the set of all prime implicants c in P, where c is not covered by the remaining prime implicants $(P - \{c\})$". (The '-' operation is set difference, and means that prime c is subtracted or deleted from the set P; the $\leq$ operator means "is covered by", i.e., in this case, if c is not completely covered by the other cubes in P, then it is essential.) Intuitively, a prime c is essential if, when you *remove it* from the set of all primes P, it is *not completely covered* by the remaining prime implicants. In this case, there is some minterm covered by c, which is not covered by any other prime. (This is a "distinguished minterm", discussed in the Quine-McCluskey handout.)

Identifying essential implicants is a *containment* problem. Suppose that the set of all primes, P, has already been generated. To check if a prime c is essential, do the following. First, produce a new cover $C = P - \{c\}$, which is the set of all primes except for c, which is removed. Second, c is essential *if and only if* c is *not* contained in C.

More formally, to check if c is essential, do the following:

(i) Delete c from P, to obtain a new cover, $C = P - \{c\}$.

(ii) Check if c is contained in C, *i.e.*, $c \leq C$. To check for containment, do the following

(a) compute the cofactor C_c of C with respect to cube c;

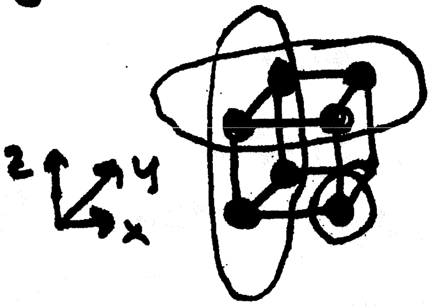
(b) check if C_c is a tautology.

An example is worked out in detail, below.

TAUTOLOGY CHECKING: an example

Note: refer to the TAUT Handout for details of the algorithm.

Example:



cover:

$F = x' + z + xy'z'$

PLA format:

x	y	z	f
0	–	–	1
–	–	1	1
1	0	0	1

Let's follow the TAUT Handout, using all rules except "M1" + "M2"! (If we used M1 + M2, we couldn't show the more interesting cases.)

Step1 : Apply basic termination rules (B1-B3, U1)

- None of these rules apply.

Therefore pick a splitting variable (say "x") & recur.

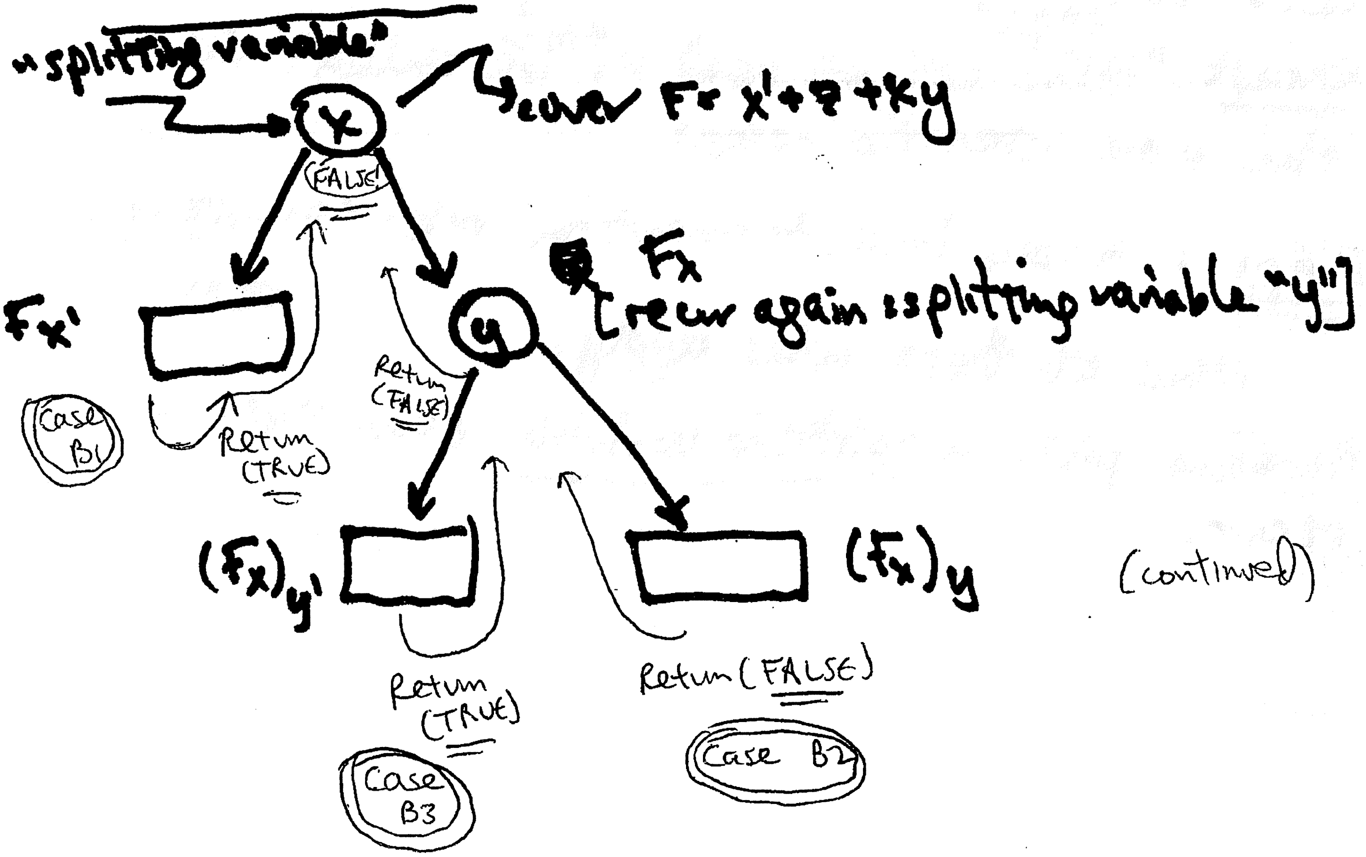
Step 2: using Shannon decomposition, we get:

$$(f=1) \Leftrightarrow (f_x=1) \text{ AND } (f_{x'}=1).$$

Now we start the recursion, taking the cofactors F_x + $F_{x'}$. The recursion continues to "leaf nodes" (shown below as rectangles) where we final can apply basic termination rules, + return final results ("true" or "false" for whether each fragment of the function is a tautology).

Recursion Tree

Final Answer: FALSE (i.e. not a tautology)



(continued)

TAUT Example (cont.)

Here are details of the cofactors computed for each node of the preceding "recursion tree".

$F_{x'}$ =

y	z	f
-	-	1
-	1	1

Terminate!; case B1 (universal cube).

return (TRUE)

(i.e. Fx' is a tautology)

F_x =

y	z	f
-	1	1
0	0	1

no termination condition is satisfied →

- pick splitting variable "y"
- recur

$(F_x)_{y'}$ =

z	f
1	1
0	1

Terminate!: case B3

return (TRUE)

(i.e. it's a tautology)

$(F_x)_y$ =

z	f
1	1

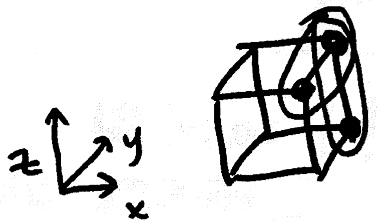
Terminate: case B2

return (FALSE)

(i.e. $(F_x)_y$ is not a tautology.)

COMPLEMENTATION PROBLEM : complement a cover (function)

Hypercube for a function f:



Cover $F = xy + xz$

(Assume contains both ON-set + DC-set of f).

PLA format:

$$F = \begin{array}{c|ccc|c} & x & y & z & f \\ \hline 1 & 1 & 1 & - & 1 \\ 2 & 1 & - & 1 & 1 \end{array}$$

Use the method in the Complementation Handout

Step 1. Termination rules B1-B4 do not apply. Go to step 2.

(recur:) Step 2. Pick splitting variable. Since no variable is binate, let us pick x.

By Recursive Complementation Theorem,

$$F' = x \cdot F'_x + x' \cdot F'_{x'}$$

However, F is unate in x (positive unate), so we can use the simpler Unate Complementation Theorem:

$$F' = F'_x + x' \cdot F'_{x'}$$

(cont.)

COMPLEMENTATION PROBLEM (cont.)

First, compute F_x & $F_{x'}$:

algebraic: $F = xy + xz$

$\boxed{F_x} = y + z$ $\boxed{F_{x'}} = \{\}$

or

PLA: $F = \begin{array}{c} \\ 1 \\ 2 \end{array}\left[\begin{array}{ccc|c} x & y & z & f \\ 1 & 1 & - & 1 \\ 1 & - & 1 & 1 \end{array}\right] \Rightarrow \boxed{F_x} = \begin{array}{c} \\ 1 \\ 2 \end{array}\left[\begin{array}{cc|c} y & z & f \\ 1 & - & 1 \\ - & 1 & 1 \end{array}\right]$ $\boxed{F_{x'}} =$ <empty table>

Next, compute F'_x & $F'_{x'}$:

$F_{x'}$: **Step #1**. Termination condition **B1**: cover is empty.

$\boxed{\text{Return}: F'_{x'} = \{1\}}$ ← universal cube

F_x: **Step#1** No termination conditions apply. Go to step 2.

(recur:) **Step#2** Pick splitting variable y.

Given $F_x = \begin{array}{c} \\ 1 \\ 2 \end{array}\left[\begin{array}{cc|c} y & z & f \\ 1 & - & 1 \\ - & 1 & 1 \end{array}\right]$

Since F is positive unate in y, use the unate complementation theorem:

$$F'_x = (F_x)'_y + y' \cdot (F_x)'_{y'}$$

(cont.)

$\boxed{\text{First, compute } (F_x)_y \text{ and } (F_x)_{y'}\text{:}}$

algebraic: $F_x = y + z$

$\boxed{(F_x)_y} = 1 + z = 1$

↑(universal cube)

$\boxed{(F_x)_{y'}} = z$

or

PLA:

$$F_x = \begin{array}{c|cc|c} & y & z & f \\ \hline 1 & 1 & - & 1 \\ 2 & - & 1 & 1 \end{array} \Rightarrow \boxed{(F_x)_y} = \begin{array}{c|c|c} & z & f \\ \hline 1 & - & 1 \\ 2 & 1 & 1 \end{array}$$

$$\boxed{(F_x)_{y'}} = \begin{array}{c|c|c} & z & f \\ \hline z & 1 & 1 \end{array}$$

$\boxed{\text{Next, compute } (F_x)'_y \text{ and } (F_x)'_{y'}\text{:}}$

$\underline{\underline{(F_x)_y}}$: $\boxed{\text{Step \#1}}$. Termination condition $\boxed{B2}$: cover contains universal cube.

$\boxed{\underline{\text{Return}}: (F_x)'_y = \{\}}$

↖ empty cover

$(F_x)_{y'}$: $\boxed{\text{Step\#1}}$. Termination condition $\boxed{B3}$: cover contains only a single cube.

$\boxed{\underline{\text{Return}}: (F_x)'_{y'} = \{z'\}}$

↑complement of cube

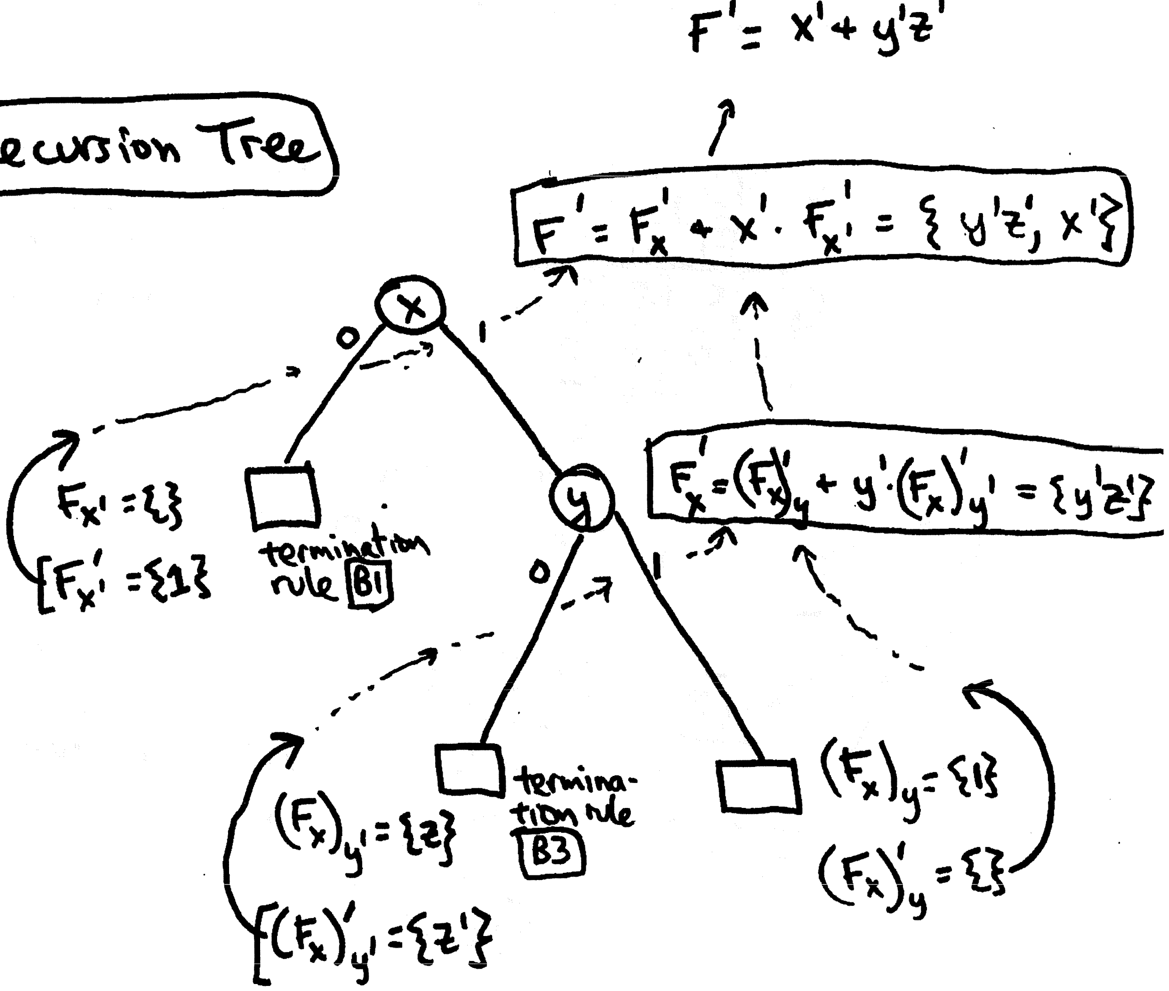
(cont.)

COMPLEMENTATION PROBLEM (cont.)

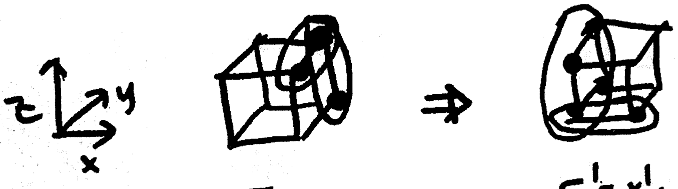
NOTE: All leaf nodes have been reached.

Return result.

Summary: Recursion Tree



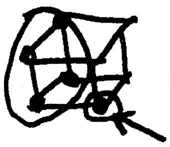
Summary: Hypercube Figure



F = xy+xz
Initial cover

F' = x'+y'z'
Final Result

Note: if we did not use the "Unate Complementation Theorem", the result would be more "fragmented" cubes:

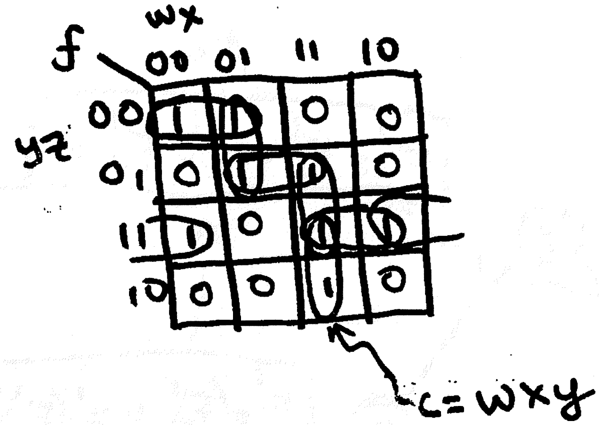


$F' = x' + \underline{xy'z'}$
↑

CONTAINMENT PROBLEM : Checking for an Essential Prime

(given **all** **the** primes of the function)

K-map for a function f:



f $yz \backslash wx$	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	1	0	1	1
10	0	0	1	0

$c = wxy$

cover P = set of all primes

$P = \{w'y'z', w'xy', xy'z, wxz, wxy, wyz, x'yz\}$

Problem: is product c essential?

Solution:

1. Compute $\mathcal{C} = P - \{c\}$ (i.e. delete cube c)

$\mathcal{C} = \{w'y'z', w'xy', xy'z, wxz, wyz, x'yz\}$

2. Check if $c \leq \mathcal{C}$. To do this:

(a) compute $\mathcal{C}_c$ ("cofactor of $\mathcal{C}$ with respect to cube c")

[see next page]

(b) check if $\mathcal{C}_c$ is a tautology

[see next page]

($\mathcal{C}_c$ is TAUTOLOGY → c NOT essential;

$\mathcal{C}_c$ not a TAUTOLOGY → c is essential)

(cont.)

CONTAINMENT PROBLEM (cont.)

Details:

2a. Compute C_c:

$$C = \begin{array}{c|cccc|c|} & w & x & y & z & f \\ \hline 1 & 0 & - & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 & - & 1 \\ 3 & - & 1 & 0 & 1 & 1 \\ 4 & 1 & 1 & - & 1 & 1 \\ 5 & 1 & - & 1 & 1 & 1 \\ 6 & - & 0 & 1 & 1 & 1 \end{array}$$

$c = wxy$

C_c: use rules in this handout

row deletion:

- Cube $c = wxy$ disagrees with rows 1,2,3,6 ⇒ delete rows 1,2,3,6
- cube $c = wxy$ agrees with rows 4+5 ⇒ keep rows 4+5

column deletion: Delete columns: w, x, y

Result: $C_c = \begin{array}{c|c|c|} & z & f \\ \hline 4 & 1 & 1 \\ 5 & 1 & 1 \end{array}$

2b. Check if C_c is a tautology:

C_c is NOT a tautology, by Rule $\boxed{B7}$

Result: c is essential.