ALGEBRAIC METHODS

© *Giovanni De Micheli*

Stanford University

Outline

- Algebraic model.
- Division and substitution.
- Kernel theory.
  - Kernel and cube extraction.
- Decomposition.

Algebraic model

- Boolean algebra:
  - Complement.
  - Symmetric distribution laws.
  - *Don't care* sets.
- Algebraic methods:
  - Boolean functions →polynomials.
  - Expressions (*sum of product* forms).

Algebraic division

- Given two algebraic expressions:

- $f_{quotient} = f_{dividend}/f_{divisor}$ when:
  - $f_{dividend} = f_{divisor} \cdot f_{quotient} + f_{remainder}$
  - $f_{divisor} \cdot f_{quotient} \neq 0$
  - and the support of $f_{divisor}$ and $f_{quotient}$ is disjoint.

Example

- Algebraic division:
  - Let $f_{dividend} = ac + ad + bc + bd + e$ and $f_{divisor} = a + b$
  - Then $f_{quotient} = c + d \quad f_{remainder} = e$
  - Because $(a + b) \cdot (c + d) + e = f_{dividend}$ and $\{a, b\} \cap \{c, d\} = \emptyset$.
- Non-algebraic division:
  - Let $f_i = a + bc$ and $f_j = a + b$.
  - Then $(a + b) \cdot (a + c) = f_i$ but $\{a, b\} \cap \{a, c\} \neq \emptyset$.

An algorithm for division

- $A = \{C_j^A\ ,\ j = 1, 2, \ldots, l\}$ set of cubes (monomials) of the dividend.

- $B = \{C_i^B\ ,\ i = 1, 2, \ldots, n\}$ set of cubes (monomials) of the divisor.

- Quotient Q and remainder R are sum of cubes (monomials).

An algorithm for division

$ALGEBRAIC\_DIVISION(A, B)$ {

for ($i = 1$ to n) {

$D = \{C_j^A$ such that $C_j^A \supseteq C_i^B\}$;

if ($D == \emptyset$) **return**($\emptyset, A$);

$D_i = D$ with var. in $sup(C_i^B)$ dropped ;

if $i = 1$

$Q = D_i$;

else

$Q = Q \cap D_i$;

}

$R = A - Q \times B$;

return(Q, R);

}

Example

$$f_{dividend} = ac + ad + bc + bd + e;$$
$$f_{divisor} = a + b;$$

- $A = \{ac, ad, bc, bd, e\}$and $B = \{a, b\}$.

- $i = 1$:
  - $C_1^B = a$, $D = \{ac, ad\}$ and $D_1 = \{c, d\}$.
  - Then $Q = \{c, d\}$.

- $i = 2 = n$:
  - $C_2^B = b$, $D = \{bc, bd\}$ and $D_2 = \{c, d\}$.
  - Then $Q = \{c, d\} \cap \{c, d\} = \{c, d\}$.

- Result:
  - $Q = \{c, d\}$ and $R = \{e\}$.
    $f_{quotient} = c + d$ and $f_{remainder} = e$.

Theorem

- Given f_i and f_j, then f_i/f_j is empty when:
  - f_j contains a variable not in f_i.
  - f_j contains a cube whose support is not contained in that of any cube of f_i.
  - f_j contains more terms than f_i.
  - The count of any variable in f_j than in f_i.

Substitution

- Consider expression pairs.

- Apply division (in any order).

- If quotient is not void:

  - Evaluate area/delay gain

  - Substitute $f_{dividend}$ by $j \cdot f_{quotient} + f_{remainder}$ where $j = f_{divisor}$.

- Use filters to reduce divisions.

Substitution algorithm

$SUBSTITUTE(\ G_n(V,E)\ )\{$
    for $(i = 1, 2, \ldots, |V|)\ \{$
        for $(j = 1, 2, \ldots, |V|; j \neq i)\ \{$
            $A =$ set of cubes of f_i;
            $B =$ set of cubes of f_j;
            if $(A, B$ pass the filter test $)\ \{$
                $(Q, R) = ALGEBRAIC\_DIVISION(A, B)$
                if $(Q \neq \emptyset)\ \{$
                    $f_{quotient} =$ sum of cubes of Q;
                    $f_{remainder} =$ sum of cubes of R;
                    if (substitution is favorable)
                        $f_i = j \cdot f_{quotient} + f_{remainder}$;
                $\}$
            $\}$
        $\}$
    $\}$
$\}$

Extraction

- Search for common sub-expressions:
  - Single-cube extraction: monomial.
  - Multiple-cube (kernel) extraction.
- Search for appropriate divisors.

Definitions

- *Cube-free* expression:
  - Cannot be factored by a cube.
- *Kernel* of an expression:
  - Cube-free quotient of the expression divided by a cube, called *co-kernel*.
- *Kernel set* $K(f)$ of an expression:
  - Set of kernels.

Example

$$f_x = ace + bce + de + g$$

- Divide f_x by a. Get ce. Not cube free.
- Divide f_x by b. Get ce. Not cube free.
- Divide f_x by c. Get $ae + be$. Not cube free.
- Divide f_x by ce. Get $a + b$. Cube free. **Kernel!**
- Divide f_x by d. Get e. Not cube free.
- Divide f_x by e. Get $ac+bc+d$. Cube free. **Kernel!**
- Divide f_x by g. Get 1. Not cube free.
- Expression f_x is a kernel of itself because cube free.
- $K(f_x) = \{(a+b); (ac+bc+d); (ace+bce+de+g)\}$.

Theorem (Brayton and McMullen)

- Two expressions f_a and f_b have a common multiple-cube divisor f_d if and only if:
  - there exist kernels $k_a \in K(f_a)$ and $k_b \in K(f_b)$ s.t. f_d is the sum of 2 (or more) cubes in $k_a \cap k_b$.

- Consequence:
  - If kernel intersection is void, then the search for common sub-expression can be dropped.

Example

$$\begin{aligned} f_x &= ace + bce + de + g \\ f_y &= ad + bd + cde + ge \\ f_z &= abc \end{aligned}$$

- $K(f_x) = \{(a+b); (ac+bc+d); (ace+bce+de+g)\}$.
- $K(f_y) = \{(a+b+ce); (cd+g); (ad+bd+cde+ge)\}$.
- The kernel set of f_z is empty.
- Select intersection $(a+b)$

$$\begin{aligned} f_w &= a + b \\ f_x &= wce + de + g \\ f_y &= wd + cde + ge \\ f_z &= abc \end{aligned}$$

Kernel set computation

- Naive method:
  - Divide function by elements in power set of its support set.
  - Weed out non cube-free quotients.
- Smart way:
  - Use recursion:
    - Kernels of kernels are kernels.
  - Exploit commutativity of multiplication.

Recursive kernel computation simple algorithm

$R\_KERNELS(f)\{$

$K = \emptyset$;

foreach variable $x \in sup(f)$ $\{$

if($|CUBES(f,x)| \geq 2$) $\{$

$f^C =$ largest cube containing x,

s.t. $CUBES(f,C) = CUBES(f,x)$;

$K = K \cup R\_KERNELS(f/f^C)$;

$\}$

$\}$

$K = K \cup f$;

return(K);

$\}$

$CUBES(f,C)\{$

return the cubes of f whose support $\supseteq C$;

$\}$

Analysis

- Some computation may be redundant:
  - Example:
    - Divide by a and then by b.
    - Divide by b and then by a.
  - Obtain duplicate kernels.

- Improvement:
  - Keep a *pointer* to literals used so far.

Recursive kernel computation

$KERNELS(f, j)\{$

$K = \emptyset$;

for $i = j$ to n $\{$

if$(|CUBES(f, x_i)| \geq 2)$ $\{$

f^C = largest cube containing x,

s.t. $CUBES(f, C) = CUBES(f, x_i)$;

if $(x_k \notin C \ \forall k < i)$

$K = K \cup KERNELS(f/f^C, i + 1)$;

$\}$

$\}$

$K = K \cup f$;

return(K);

$\}$

Example

$$f = ace + bce + de + g$$

- Literals a or b. No action required.
- Literal c. Select cube ce:
  - Recursive call with arguments: $(ace+bce)/ce = a + b$; pointer $j = 3 + 1$.
  - Call considers variables $\{d, e, g\}$. No kernel.
  - Adds $a + b$ to the kernel set at the last step.
- Literal d. No action required.
- Literal e. Select cube e:
  - Recursive call with arguments: $ac + bc + d$ and pointer $j = 5 + 1$.
  - Call considers variable $\{g\}$. No kernel.
  - Adds $ac + bc + d$ to the kernel set at the last step.
- Literal g. No action required.
- Adds $ace + bce + de + g$ to the kernel set.
- $K = \{(ace + bce + de + g), (ac + bc + d), (a + b)\}$.

Matrix representation of kernels

- Boolean matrix:
  - Rows: cubes. Columns: variables.
- Rectangle (R, C):
  - Subset of rows and columns
    with all entries equal to 1.
- Prime rectangle:
  - Rectangle not inside any other rectangle.
- Co-rectangle (R, C') of a rectangle (R, C):
  - C' are the columns not in C.
- A co-kernel corresponds to a
  prime rectangle with at least two rows.

Example

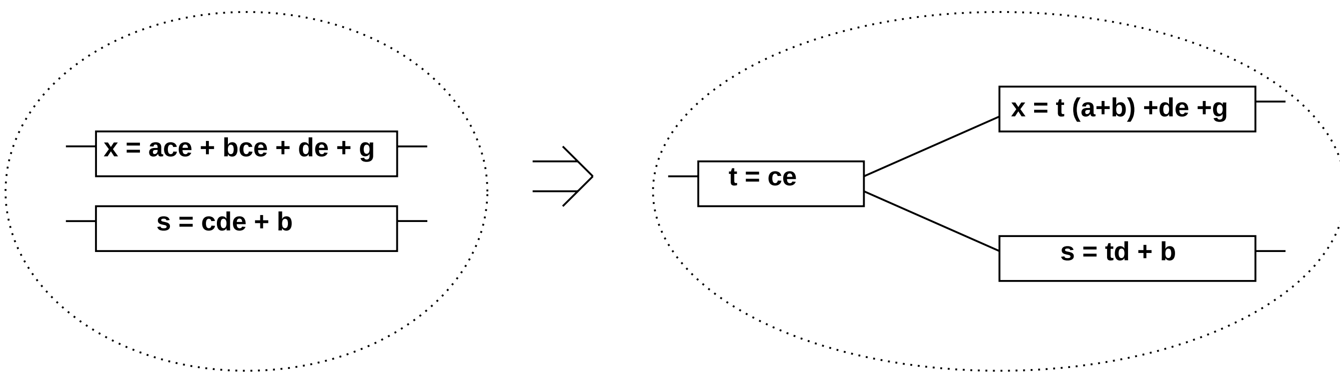
$$f_x = ace + bce + de + g$$

	var	a	b	c	d	e	g
cube	$R \backslash C$	1	2	3	4	5	6
ace	1	1	0	1	0	1	0
bce	2	0	1	1	0	1	0
de	3	0	0	0	1	1	0
g	4	0	0	0	0	0	1

- Rectangle (prime): $(\{1, 2\}, \{3, 5\})$
  - Co-kernel ce.
- Co-rectangle: $(\{1, 2\}, \{1, 2, 4, 6\})$.
  - Kernel $a + b$.

Single-cube extraction

© GDM



Single-cube extraction

- Form *auxiliary* function:
  - Sum of all local functions.
- Form matrix representation:
  - A rectangle with two rows represents a *common cube.*
  - Best choice is a prime rectangle.
- Use function ID for cubes:
  - Cube intersection from different functions.

Example

- Expressions:
  - $f_x = ace + bce + de + g$
  - $f_s = cde + b$

- Auxiliary function:
  - $f_{aux} = ace + bce + de + g + cde + b$

- Matrix:

		var	a	b	c	d	e	g
cube	ID	$R \backslash C$	1	2	3	4	5	6
ace	x	1	1	0	1	0	1	0
bce	x	2	0	1	1	0	1	0
de	x	3	0	0	0	1	1	0
g	x	4	0	0	0	0	0	1
cde	s	5	0	0	1	1	1	0
b	s	6	0	1	0	0	0	0

- Prime rectangle: $(\{1, 2, 5\}, \{3, 5\})$

- Extract cube ce.

Cube extraction algorithm

$CUBE\_EXTRACT(\ G_n(V, E) \ )\{$

while (some favorable common cube exist) $\{$

$C =$ select common cube to extract;

Generate new label l;

Add to network v_l and $f_l = f^C$;

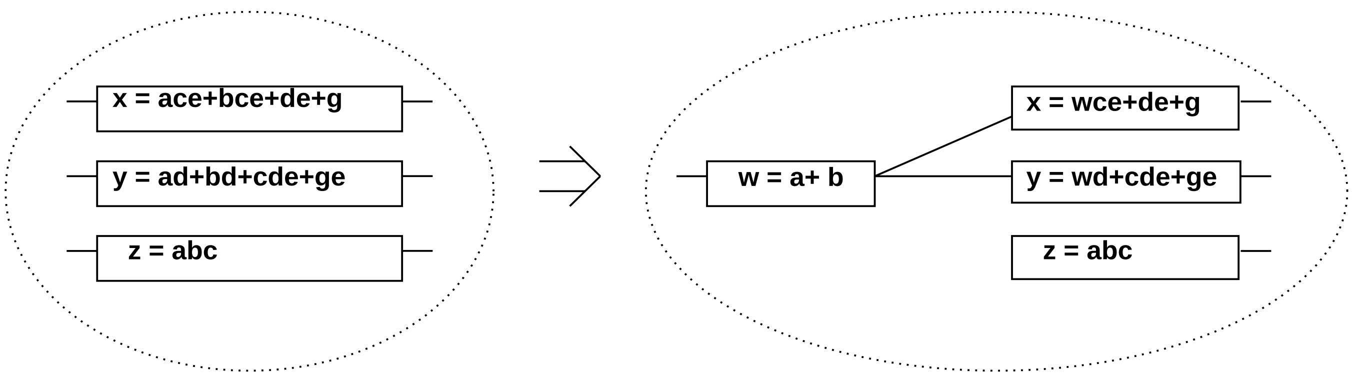
Replace all functions f, where f_l is a divisor,

by $l \cdot f_{quotient} + f_{remainder}$;

$\}$

$\}$

Multiple-cube extraction



Multiple-cube extraction

- We need a kernel/cube matrix.

- Relabeling:
  - Cubes by new variables.
  - Kernels by cubes.

- Form *auxiliary* function:
  - Sum of all kernels.

- Extend cube intersection algorithm.

Example

- $f_p = ace + bce$.
  - $K(f_p) = \{(a + b)\}$.

- $f_q = ae + be + d$.
  - $K(f_q) = \{(a + b); (ae + be + d)\}$.

- Relabeling:
  - $x_a = a; x_b = b; x_{ae} = ae; x_{be} = be; x_d = d;$
    * $K(f_p) = \{\{x_a, x_b\}\}$
    * $K(f_q) = \{\{x_a, x_b\}; \{x_{ae}, x_{be}, x_d\}\}$.

Example (2)

- $f_{aux} = x_a x_b + x_a x_b + x_{ae} x_{be} x_d$.

- Co-kernel: $x_a x_b$.

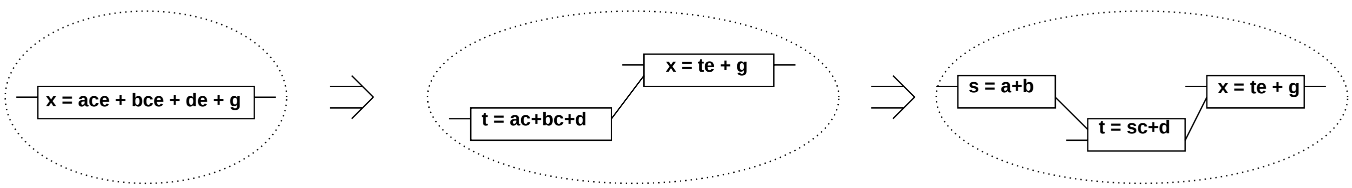
  - $x_a x_b$ corresponds to kernel intersection $a + b$.

  - Extract $a + b$ from f_p and f_q.

Kernel extraction algorithm


```
KERNEL_EXTRACT( G_n(V,E) , n, k){
  while (some favorable common kernel intersection exist) {
    Compute kernel set of level ≤ k;
    for (i = 1 to n) {
      Compute kernel intersections;
      f = select kernel intersection to extract;
      Generate new label l;
      Add v_l to the network with expression f_l = f;
      Replace all functions f where f_l is a divisor
        by l · f_quotient + f_remainder;
      }
    }
  }
```

Decomposition



Decomposition

- Different ways:
  - Method of Ashenhurst and Curtis.
  - NAND/NOR decomposition.
- Kernel-based decomposition:
  - Divide expression recursively.

Example

$$f_x = ace + bce + de + g$$

- Select kernel $ac + bc + d$.
- Decompose: $f_x = te + g$; $f_t = ac + bc + d$;
- Recur on the quotient f_t:
  - Select kernel $a + b$:
  - Decompose: $f_t = sc + d$; $f_s = a + b$;

Decomposition algorithm

$DECOMPOSE(\ G_n(V,E)\ ,\ k)\{$

repeat $\{$

v_x = selected vertex with expression whose size is above k;

if $(v_x = \emptyset)$ **return**;

decompose expression f_x;

$\}$

$\}$

Summary
Algebraic transformations

- View Boolean functions as algebraic expression.
- Fast manipulation algorithms.
- Some optimality lost,
  because Boolean properties are neglected.
- Useful to reduce large networks.