Boolean Functions and Circuits

Gianpiero Cabodi *Stefano Quer*

Politecnico di Torino

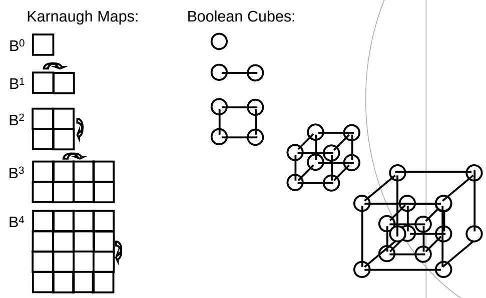
Torino, Italy

{gianpiero.cabodi,stefano.quer}@polito.it

http://staff.polito.it/{gianpiero.cabodi,stefano.quer}/

The Boolean Space B^n

- B = { 0,1}
- B^2 = {0,1} X {0,1} = {00, 01, 10, 11}



Boolean Expressions

- **If B = {0, 1} a Boolean Function is**
  - **$y = f(X) : B^n \rightarrow B$**
- **With**
  - **$X = (x_1, x_2, ..., x_n) \in B^n$**
  - **$x_1 \in B$**
  - **$x_1, x_2 ...$ are variable**
  - **$x_1, x'_2 ...$ are literals**
- **Basically**
  - **f maps each vertex of B^n to 0 or 1**

- **Definitions**
  - **The onset of f is**
    - $\{x \mid f(x) = 1\} = f^{-1}(1) = f^1$
  - **The offset of f is**
    - $\{x \mid f(x) = 0\} = f^{-1}(0) = f^0$
  - **f is a tautology iff ALL assignments are models, i.e.,**
    - $f^1 = B^n$, i.e., $f \equiv 1$
  - **f is contradictory (not satisfiable) iff NONE is, i.e.,**
    - $f^0 = B^n$, i.e., $f^0 = \phi$, i.e., $f \equiv 0$
  - **If f(x)=g(x) for all $x \in B^n$, then f and g are equivalent**
  - **A satisfying assignment is a set of input values in the onset of the function**

Boolean Operations: AND, OR, NOT

- **Given two Boolean functions**

  $f : B^n \rightarrow B$ $\quad$ $g : B^n \rightarrow B$

  we define
  - **The AND operation**

    $h = f \wedge g$
    - $h = \{x \mid f(x)=1 \wedge g(x)=1\}$
  - **The OR operation**

    $h = f \vee g$
    - $h = \{x \mid f(x)=1 \vee g(x)=1\}$
  - **The COMPLEMENT operation**

    $h = \neg f$
    - $h = \{x \mid f(x) = 0\}$

Cofactor and Quantification

- **Given a Boolean function**

  $f : B^n \rightarrow B$

  with the input variable $(x_1,x_2,\ldots,x_i,\ldots,x_n)$, we define
  - **The positive cofactor**

    $h = f_{xi}$
    - $h = \{x \mid f(x_1,x_2,\ldots,1,\ldots,x_n)=1\}$
  - **The negative cofactor**

    $h = f_{xi}$
    - $h = \{x \mid f(x_1,x_2,\ldots,0,\ldots,x_n)=1\}$
  - **The existential quantification of variable x_i**

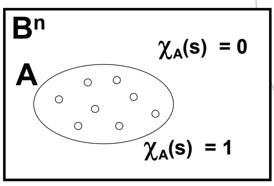
    $h = \exists x_i . F$
    - $h = \{x \mid f(x_1,x_2,\ldots,0,\ldots,x_n)=1 \vee f(x_1,x_2,\ldots,1,\ldots,x_n)=1\}$
  - **The universal quantification of variable x_i**

    $h = \forall x_i . F$
    - $h = \{x \mid f(x_1,x_2,\ldots,0,\ldots,x_n)=1 \wedge f(x_1,x_2,\ldots,1,\ldots,x_n)=1\}$

Characteristic function

- ❖ Given a set A
- ❖ We define the Characteristic Function $\chi_A(s)$ of the set A as

$$\chi_A(s) = \begin{cases} 1 & \text{IFF} \quad s \in A \\ 0 & \text{IFF} \quad s \notin A \end{cases}$$



Representation of Boolean Functions

- ❖ What do we need?

  A good data structure for Boolean formulas !!!
- ❖ We need representations for Boolean Functions for two reasons
  - ◆ A mechanism to build a data structure that represents the problem
  - ◆ A set of algorithm to manipulate the representation used
  - ◆ A decision procedure to decide about SAT or UNSAT, i.e., to perform Boolean reasoning

---

- ❖ Classical Methods
  - ◆ Canonical Forms
    - ✧ Canonical: one and only one representation for each function, i.e., data structure uniquely represents function
    - ✧ Decision procedure is trivial (e.g., just pointer comparison)
    - ✧ Example: Reduced Ordered Binary Decision Diagrams
    - ✧ Problem: Size of data structure is in general exponential
  - ◆ NON Canonical Forms
    - ✧ Systematic search for satisfying assignment
    - ✧ Size of data structure is linear
    - ✧ Problem: decision may take an exponential amount of time
- ❖ Non-Classical Methods

Classical Canonical Methods

- ❖ Truth Table

  F = Graphical/Tabular Representation
- ❖ Canonical Disjunctive Normal Form (cDNF)

  $F = (x_1^* \wedge x_2^* \wedge \dots \wedge x_n^*) \vee \dots \vee (x_1^* \wedge x_2^* \wedge \dots \wedge x_n^*)$
- ❖ Canonical Conjunctive Normal Form (cCNF)

  $F = (x_1^* \vee x_2^* \vee \dots \vee x_n^*) \wedge \dots \wedge (x_1^* \vee x_2^* \vee \dots \vee x_n^*)$

Two-Level Normal Forms

- ❖ Automata

  F = Graphical/Graph Representation

  (Reduced Automatas are a Canonical Representation)

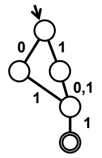
Example

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- ❖ Truth Table
- ❖ DNF

  $F = (\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3)$
- ❖ CNF

  $F = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \dots$
- ❖ Automata



- ❖ Pros
  - ◆ Unique representation (one and only for each function)
  - ◆ Constant Time Comparison (same representation)
- ❖ Cons
  - ◆ Exponential Size
  - ◆ Complex Resolution Algorithms
  - ◆ Satisfiability is NP-complete (Cook) (i.e., resolution algorithms require exponential time)
  - ◆ Examples
    - ✧ DNF ➔ satisfiability requires polynomial time, tautology is co-NP complete
    - ✧ CNF ➔ … vice-versa …
    - ✧ Conversion CNF ⟷ DNF is exponential
  - ◆ Example
    - ✧ $F_{CNF} = (x_{01} \vee x_{11}) \wedge (x_{02} \vee x_{12}) \wedge \dots \wedge (x_{0n} \vee x_{1n})$ size n
    - ✧ $F_{DNF} = \dots$ size $n \cdot 2^n$

Classical NON Canonical Methods

- Disjunctive Normal Form (DNF)
  - $F = (x_1^* \wedge \ldots \text{<some i missing>} \ldots \wedge x_n^*) \vee \ldots \vee (x_1^* \wedge \ldots \wedge x_n^*)$
- Conjunctive Normal Form (CNF)
  - $F = (x_1^* \vee \ldots \text{<some i missing>} \ldots \vee x_n^*) \wedge \ldots \wedge (x_1^* \vee \ldots \vee x_n^*)$
- Automata
  - F = Graphical/Graph Representation
  - (Not-Reduced Automatas)

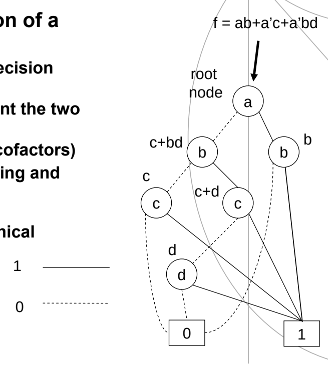
- Pros
  - Non-Exponential Representation's Size
- Cons
  - Non-Unique representation (more representations for each function)
  - Complex Algorithms for Comparison
  - Complex Algorithms for Conversions

Non Classical Methods

- Decision Diagrams
  - BDDs - Binary Decision Diagrams
  - ZBDDs - Zero Suppressed Binary Decision Diagrams
  - Etc.
- Boolean Circuits
  - RBCs – Reduced Boolean Circuits
  - BEDs – Boolean Expression Diagrams
  - AIGs – And Inverter Graphs

Binary Decision Diagram (BDD)

- Graph representation of a Boolean function f
  - vertices represent decision nodes for variables
  - two children represent the two subfunctions
  - f(x = 0) and f(x = 1) (cofactors)
  - restrictions on ordering and reduction rules
  - can make a BDD representation canonical



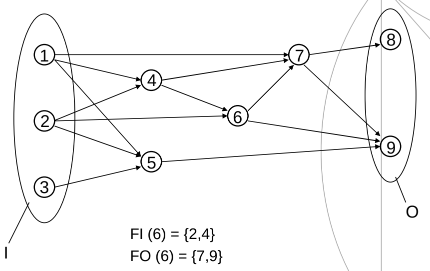
Boolean Circuits

- Used for two main purposes
  - As representation for Boolean reasoning engine
  - As target structure for logic implementation which gets restructured in a series of logic synthesis steps until result is acceptable
- Efficient representation for most Boolean problems we have in CAD
  - Memory complexity is same as the size of circuits we are actually building
- Close to input representation and output representation in logic synthesis

---

- Definition
  - A Boolean circuit is a directed graph C(G,N) where G are the gates and $N \subseteq G \times G$ is the set of directed edges (nets) connecting the gates
  - Some of the vertices are designated
    - Inputs: $I \subseteq G$
    - Outputs: $O \subseteq G, I \cap O = \emptyset$
  - Each gate g is assigned a Boolean function f_g which computes the output of the gate in terms of its inputs
  - The fanin FI(g) of a gate g are all predecessor vertices of g
    - $FI(g) = \{g' \mid (g',g) \in N\}$
  - The fanout FO(g) of a gate g are all successor vertices of g
    - $FO(g) = \{g' \mid (g,g') \in N\}$
  - The cone CONE(g) of a gate g is the transitive fanin of g and g itself.
  - The support SUPPORT(g) of a gate g are all inputs in its cone
    - $SUPPORT(g) = CONE(g) \cap I$

Example



FI (6) = {2,4}
FO (6) = {7,9}
CONE (6) = {1,2,4,6}
SUPPORT (6) = {1,2}

And Inverter Graphs (AIGs)

- **Base data structure uses two-input AND function for vertices and INVERTER attributes at the edges (individual bit)**
  - **use De'Morgan's law to convert OR operation etc.**
- **Hash table to identify and reuse structurally isomorphic circuits**