# Digital Logic Design: a rigorous approach ©

## Chapter 9: Representation of Boolean Functions by Formulas

Guy Even  Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

April 21, 2020

Book Homepage:
http://www.eng.tau.ac.il/~guy/Even-Medina

# Normal Forms of Boolean Functions

- A normal form is a restricted syntax for Boolean Formulas.
- For example, Negation Normal Form (NNF) allows negations only of variables or constants.
- We now consider two more normal forms:
  - Disjunctive Normal Form (DNF) also called Sum of Products (SoP)
  - Conjunctive Normal Form (CNF) also called Product of Sums (PoS)
- We will also consider polynomials over a finite field!

# Literals

**Definition (literal)**

A variable or a negation of a variable is called a literal.

**Example**

- $X$
- $\text{NOT}(X)$

# Product / Conjunction

Recall that:

- $\text{AND}, \cdot, \wedge$ denote the same logical connective.
- Associativity of $\text{AND}$ function allows us to omit parenthesis.

**Definition (product/conjunction)**

A Boolean formula $\varphi$ is a conjunction (or a product) if

$$\varphi = \ell_1 \text{ AND } \cdots \text{ AND } \ell_k,$$

for $k \geq 1$ and every $\ell_i$ is a literal.

**Example**

$$\begin{aligned} X \cdot \bar{Y} \cdot Z &= (X \text{ AND } \bar{Y} \text{ AND } Z) \\ &= (X \wedge \bar{Y} \wedge Z) \end{aligned}$$

# Notation

- With each product $p$, we associate the set of variables that appear in $p$.
- The set of variables that appear in $p$ is denoted by $vars(p)$.
- Let $vars^+(p)$ denote the set of variables that appear in $p$ that appear without negation.
- Let $vars^-(p)$ denote the set of variables that appear in $p$ that with negation.
- Let $literals(p)$ denote the set of literals that appear in $p$.
- $p = \bigwedge_{\ell \in literals(p)} \ell = (\bigwedge_{X_i \in vars^+(p)} X_i)$ AND $(\bigwedge_{X_i \in vars^-(p)} \bar{X}_i)$.

**Example**

Let $p = X_1 \cdot \bar{X}_2 \cdot X_3$, then $vars(p) = \{X_1, X_2, X_3\}$, $vars^+(p) = \{X_1, X_3\}$ and $vars^-(p) = \{X_2\}$, and $literals(p) = \{X_1, \bar{X}_2, X_3\}$.

# Simple Products

**Definition (simple product)**

A product term $p$ is simple if every variable appears at most once in $p$.

a simple product: $X_1 \cdot X_2 \cdot \bar{X}_3$
not simple: $X \cdot X$, $X_1 \cdot X_2 \cdot \bar{X}_1$
Recall that:

1. $X \cdot \bar{X}$ is a contradiction
2. $X \cdot X$ is logically equivalent to $X$
3. $\bar{X} \cdot \bar{X}$ is logically equivalent to $\bar{X}$.

**Claim**

*Every product is a contradiction or logically equivalent to a simple product.*

# Minterms

**Definition (minterm)**

A simple product term $p$ is a minterm with respect to a set $U$ of variables if $vars(p) = U$.

**Example**

$U = \{X, Y, Z\}$. Minterms: $X \cdot Y \cdot Z$, $\bar{X} \cdot \bar{Y} \cdot Z$.

**question**

How many different minterms are there with respect to $U$?

**lemma**

A minterm $p$ attains the truth value 1 for exactly one truth assignment.

# Sum-of-Products (SOP) / Disjunctive Normal Form (DNF)

**Definition (SoP/DNF)**

A Boolean formula $\varphi$ is called a sum-of-products (SOP) (or in Disjunctive Normal Form (DNF)) if satisfies one of the following conditions:

1. $\varphi = p_1 + \cdots + p_k$, where $k \geq 2$ and each $p_i$ is a product
2. $\varphi$ is a product

(the case of a product is a degenerate case for $k = 1$ and includes the case of a single literal.)

# Examples

Each of the following formulas is a sum-of-products.

1. $\varphi_1 = X \cdot Y + X \cdot Y$,
2. $\varphi_2 = (\bar{A} \text{ AND } B \text{ AND } C) \text{ OR } (A \text{ AND } \bar{B} \text{ AND } C) \text{ OR } \bar{D}$,
3. $\varphi_3 = L$.

Each of the following formulas is not a sum-of-products.

1. $(X + Y) \cdot Z$,
2. $(A \text{ OR } B) \text{ AND } (C \text{ OR } D)$.

# SoP representation

## Definition

For a $v \in \{0,1\}^n$, define the minterm $p_v$ to be $p_v \stackrel{\triangle}{=} (\ell_1^v \cdot \ell_2^v \cdots \ell_n^v)$, where:

$$\ell_i^v \stackrel{\triangle}{=} \begin{cases} X_i & \text{if } v_i = 1 \\ \bar{X}_i & \text{if } v_i = 0. \end{cases}$$

## Question

What is the truth assignment that satisfies $p_v$?

## Question

Prove that the mapping $v \mapsto p_v$ is a bijection from $\{0,1\}^n$ to the set of all minterms over $\{X_1, \ldots, X_n\}$.

# SoP representation - cont (i)

**Definition (preimage)**

Let $f : \{0,1\}^n \to \{0,1\}$. Let $f^{-1}(1)$ denote the set

$$f^{-1}(1) \stackrel{\triangle}{=} \{v \in \{0,1\}^n \mid f(v) = 1\}.$$

**Definition**

The set of minterms of $f$ is defined by

$$Min(f) \stackrel{\triangle}{=} \{p_v \mid v \in f^{-1}(1)\}.$$

# SoP Representation - cont (ii)

**Theorem**

*Every Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *that is not a constant zero is expressed by the sum of the minterms in* $Min(f)$.

**Question**

Let $\varphi$ be the sum of the minterms in $Min(f)$ and let $\tau$ denote a truth assignment that satisfies $\varphi$ (i.e., $\hat{\tau}(\varphi) = 1$). How many products in $\varphi$ are satisfied by $\tau$?

# sum-of-minterms: a "bad" example

We are interested in "short" formulas that express a given Boolean function.

- Consider the constant Boolean function $f : \{0,1\}^n \to \{0,1\}$ that is defined by $f(v) = 1$, for every $v$.
- The sum-of-minterms that represents $f$ is the sum of all the possible minterms over $n$ variables. This sum contains $2^n$ minterms.
- On the other hand, $f$ can be represented by the constant 1.
- The question of finding the shortest sum-of-products that represents a given Boolean formula is discussed in more detail in our book.

# Product of Sums (PoS)/ Conjunctive Normal Form (CNF)

The second normal form we consider is called conjunctive normal form (CNF) or product of sums (POS).

# Sum / Disjunction

Recall that:

- $\text{OR}, +, \vee$ denote the same logical connective.
- Associativity of $\text{OR}$ function allows us to omit parenthesis.

**Definition**

A Boolean formula $s$ is a disjunction (or a sum) if

$$s = \ell_1 + \cdots + \ell_k,$$

for $k \geq 1$ and every $\ell_i$ is a literal.

**Example**

$$\begin{aligned} X + \bar{Y} + Z &= (X \text{ OR } \bar{Y} \text{ OR } Z) \\ &= (X \vee \bar{Y} \vee Z) \end{aligned}$$

Define $\mathit{vars}(s), \mathit{vars}^+(s), \mathit{vars}^-(s), \mathit{literals}(s)$ as in products.

# Maxterms

**Definition (simple sum)**

A sum $s$ is simple if every variable appears at most once in $s$.

**Definition (maxterm)**

A simple sum term $s$ is a maxterm with respect to a set $U$ of variables if $vars(s) = U$.

**Question**

How many maxterms are there with respect to $U$?

**Lemma**

*A maxterm $s$ is satisfied by all but one truth assignment ($s$ attains the truth value 0 for exactly one truth assignment).*

# Product-of-Sums (PoS) / Conjunctive Normal Form (CNF)

**Definition (SoP/DNF)**

A Boolean formula $\varphi$ is called a product-of-sums (POS) (or in Conjunctive Normal Form (CNF)) if satisfies one of the following conditions:

1. $\varphi = s_1 \cdot \cdots \cdot s_k$, where $k \geq 2$ and each $s_i$ is a sum
2. $\varphi$ is a sum

(the case of a sum is a degenerate case for $k = 1$ and includes the case of a single literal.)

# relation to de Morgan duality

Recall that $DM(\varphi)$ is the De Morgan dual of the formula $\varphi$.

## observation

1. If $p$ is a product, then $DM(p)$ is a sum.
2. If $s$ is a sum, then $DM(s)$ is a product.
3. If $p$ is a minterm, then $DM(p)$ is a maxterm.
4. If $s$ is a maxterm, then $DM(s)$ is a minterm.
5. If $p$ is a sum-of-products, then the formula $DM(p)$ is a product-of-sums.
6. If $p$ is a product-of-sums, then the formula $DM(p)$ is a sum-of-products.

# Maxterms of a Boolean Function

## Definition

For a $v \in \{0,1\}^n$, define the maxterm $s_v$ to be $s_v \triangleq (m_1^v + \cdots + m_n^v)$, where:

$$m_i^v \triangleq \begin{cases} X_i & \text{if } v_i = 0 \\ \bar{X}_i & \text{if } v_i = 1. \end{cases}$$

Note that $\ell_i^v$ is logically equivalent to $\text{NOT}(m_i^v)$.

## Claim

*For every $v \in \{0,1\}^n$, the following formulas are tautologies:*

$$s_v \leftrightarrow DM(p_v)$$
$$p_v \leftrightarrow DM(s_v)\ .$$

## Question

Which truth assignment does not satisfy $s_v$?

# PoS representation of Boolean Functions

**Definition (Maxterms of a function $f : \{0,1\}^n \to \{0,1\}$)**

$$Max(f) \triangleq \{s_v \mid v \in f^{-1}(0)\} .$$

**Theorem**

*Every Boolean function $f : \{0,1\}^n \to \{0,1\}$ that is not a constant one is expressed by the product of the maxterms in $Max(f)$.*

# De Morgan Duality and CNF representation

**Question**

What is the relation between $Min(f)$ and $Max(\text{NOT}(f))$?

Let $U = \{X_1, \ldots, X_n\}$ and $f : \{0,1\}^n \to \{0,1\}$.

**Lemma**

*Let $p$ denote a minterm wrt $U$. Then,*

$$p \in Min(f) \iff DM(p) \in Max(\text{NOT}(f))$$

*Let $s$ denote a maxterm wrt $U$. Then,*

$$s \in Max(f) \iff DM(s) \in Min(\text{NOT}(f))$$

**Theorem**

$CNF(f) = DM(DNF(\text{NOT}(f)))$

# Representation by polynomials

**Definition**

The Galois Field $GF(2)$ is defined as follows.

1. Elements: the elements of $GF(2)$ are $\{0, 1\}$. The zero is called the additive unity and one is called the multiplicative unity.
2. Operations:
   1. addition which is simply the XOR function, and
   2. multiplication which is simply the AND function.

In the context of $GF(2)$ we denote multiplication by $\cdot$ and addition by $\oplus$.

# $GF(2)$ properties

We are used to infinite fields like the rationals (or reals) with regular addition and multiplication. In these fields, $1 + 1 \neq 0$. However, in $GF(2)$, $1 \oplus 1 = 0$.

**Observation**

$X \oplus X = 0$, *for every* $X \in \{0, 1\}$.

A minus sign in a field means the additive inverse.

**Definition**

The element $-X$ stands for the element $Y$ such that $X \oplus Y = 0$.

# $GF(2)$ properties - more

**Observation**

*In $GF(2)$, the additive inverse of $X$ is $X$ itself, namely $-X = X$, for every $X \in \{0, 1\}$.*

Thus, we need not write minus signs, and adding an $X$ is equivalent to subtracting an $X$.

The distributive law holds in $GF(2)$, namely:

**Observation**

$(X \oplus Y) \cdot Z = (X \cdot Z) \oplus (Y \cdot Z)$, *for every* $X, Y, Z \in \{0, 1\}$.

# $GF(2)$ properties - even more

Let $X^k$ denote the product (AND of literals)

$$X^k \triangleq \overbrace{X \cdot \ldots \cdot X}^{k \text{ times}}.$$

We define $X^0 = 1$, for every $X \in \{0, 1\}$. The following observation proves that multiplication is idempotent.

**Observation**

*$X^k = X$, for every $k \in \mathbb{N}^+$ and $X \in \{0, 1\}$.*

# $GF(w)$ is a field like the reals

The structure of a field allows us to solve systems of equations. In fact, Gauss elimination works over any field. The definition of a vector space over $GF(2)$ is just like the definition of vector spaces over the reals. Definitions such as linear dependence, dimension of vector spaces, and even determinants apply also to vector spaces over $GF(2)$.

# Examples

- $$X_1 \oplus X_2 = 0 \quad \Leftrightarrow \quad X_1 = X_2.$$
- We show how to solve a simple systems of equalities over $GF(2)$ using Gauss elimination. Consider the following system of equations

$$\begin{array}{rcrcrl} X_1 & \oplus & X_2 & \oplus & X_3 & = 0\,, \\ X_1 & & & \oplus & X_3 & = 0\,, \\ & & X_2 & \oplus & X_3 & = 1\,. \end{array}$$

# Polynomials over $GF(2)$

**Definition**

A monomial in $GF(2)$ over the variables in the set $U$ is a finite product of the elements in $U$ or a constant in $\{0, 1\}$.

**Observation**

*Every monomial $p$ in $GF(2)$ over the variables in $U$ equals a constant or a simple product of variables in $p$.*

- By commutativity: $X_1 \cdot X_2 \cdot X_3 \cdot X_1 = X_1^2 \cdot X_2 \cdot X_3$.
- Positive exponents can be reduced to one. For example, $X_1^2 \cdot X_2 \cdot X_3$ equals $X_1 \cdot X_2 \cdot X_3$.

# Polynomials

**Definition**

A polynomial in $GF(2)$ over the variables in the set $U$ is a finite sum of monomials.

Example: $X_1 \cdot X_2 \oplus X_1 \cdot X_3 \oplus X_2 \cdot X_3 \oplus 1$.
We denote the set of all polynomials in $GF(2)$ over the variables in $U$ by $GF(2)[U]$. Just as multivariate polynomials over the reals can be added and multiplied, so can polynomials in $GF(2)[U]$.

# representation by polynomials in $GF(2)[U]$

Every polynomial $p \in GF(2)[U]$ is a Boolean function $f_p : \{0,1\}^{|U|} \to \{0,1\}$. The converse is also true.

**Theorem**

*Every Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be represented by a polynomial in $GF(2)[U]$, where $U = \{X_1, \ldots, X_n\}$.*

**proof outline**

- easy: if $f$ is constant.
- For every $v \in \{0,1\}^n$, there exists a polynomial $p'_v$ that is expressed by the minterm $p_v$ (why?).

$$f = \bigoplus_{v \in f^{-1}(1)} p'_v.$$

# Side effect

**Corollary**

*The set of connectives* $\{\text{XOR}, \text{AND}\}$ *is complete.*

# Satisfiability

The problem of satisfiability of Boolean formulas is defined as follows.

Input: A Boolean formula $\varphi$.

Output: The output should equal "yes" if $\varphi$ is satisfiable. If $\varphi$ is not satisfiable, then the output should equal "no".

Note that the problem of satisfiability is quite different if the input is a truth table of a Boolean function. In this case, we simply need to check if there is an entry in which the function attains the value 1.

# Relation to P vs. NP

The main open problem in Computer Science since 1971 is whether $P = NP$. We will not define the classes $P$ and $NP$, but we will phrase an equivalent question in this section.

Consider a Boolean formula $\varphi$. Given a truth assignment $\tau$, it is easy to check if $\hat{\tau}(\varphi) = 1$. We showed how this can be done in Algorithm EVAL. In fact, the running time of the EVAL algorithm is linear in the length of $\varphi$.

On the other hand, can we find a satisfying truth assignment by ourselves (rather than check if $\tau$ is a satisfying assignment)? Clearly, we could try all possible truth assignments. However, if $n$ variables appear in $\varphi$, then the number of truth assignments is $2^n$.

# Satisfiability and P vs. NP

We are ready to formulate a question that is equivalent to the question $P = NP$.

**Satisfiability in polynomial time**

Does there exist a constant $c > 0$ and an algorithm $Alg$ such that:

1. Given a Boolean formula $\varphi$, algorithm $Alg$ decides correctly whether $\varphi$ is satisfiable.
2. The running time of $Alg$ is $O(|\varphi|^c)$, where $|\varphi|$ denotes the length of $\varphi$.

This seemingly simple question turns out to be a very deep problem about what can be easily computed versus what can be easily proved. It is related to the question whether there is a real gap between checking that a proof is correct and finding a proof.