Digital Logic Design: a rigorous approach ©

Chapter 6: Propositional Logic

(Part 1)

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

March 23, 2020

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

Building Blocks of Boolean Formulas

The building blocks of a Boolean formula are constants, variables, and connectives.

1. A constant is either 0 or 1. As in the case of bits, we interpret a 1 as "true" and a 0 as a "false". The terms constant and bit are synonyms; the term bit is used in Boolean functions and in circuits while the term constants is used in Boolean formulas.
2. A variable is an element in a set of variables. We denote the set of variables by U. The set U does not contain constants. Variables are usually denoted by upper case letters.
3. Connectives are used to build longer formulas from shorter ones. We denote the set of connectives by $\mathcal{C}$.

Logical Connectives

We consider unary, binary, and higher arity connectives.

1. There is only one unary connective called negation. Negation of a variable A is denoted by $\text{NOT}(A)$, $\neg A$, or $\bar{A}$.
2. There are several binary connectives, the most common are AND (denoted also by $\wedge$ or $\cdot$) and OR (denoted also by $\vee$ or $+$). A binary connective is applied to two formulas. We later show the relation between binary connectives and Boolean functions $B : \{0, 1\}^2 \to \{0, 1\}$.
3. A connective has arity j if it is applied to j formulas. The arity of negation is 1, the arity of AND is 2, etc.

Example: parse tree

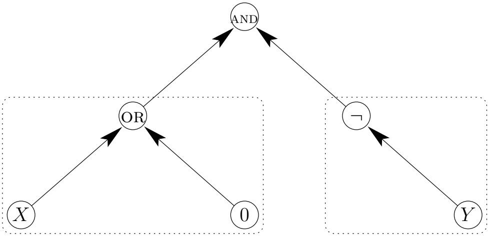


Figure: A parse tree that corresponds to the Boolean formula $((X \text{ OR } 0) \text{ AND } (\neg Y))$. The rooted trees that are hanging from the root of the parse tree (the AND connective) are bordered by dashed rectangles.

Parse Trees

We use parse trees to define Boolean formulas.

Definition

A parse tree is a pair (G, π), where $G = (V, E)$ is a rooted tree and $\pi : V \to \{0, 1\} \cup U \cup \mathcal{C}$ is a labeling function that satisfies:

1. A leaf is labeled by a constant or a variable. Formally, if $v \in V$ is a leaf, then $\pi(v) \in \{0, 1\} \cup U$.
2. An interior vertex v is labeled by a connective whose arity equals the in-degree of v. Formally, if $v \in V$ is an interior vertex, then $\pi(v) \in \mathcal{C}$ is a connective with arity $deg_{in}(v)$.

We usually use only unary and binary connectives. Thus, unless stated otherwise, a parse tree has an in-degree of at most two.

Boolean formulas

- We use strings that contain constants, variables, connectives, and parenthesis to construct Boolean formulas.
- We use parse trees to define Boolean formulas.
- This definition is constructive (inorder traversal of the parse tree).

Examples of Good and Bad Formulas

- $(A \text{ AND } B)$
- $(A \text{ OR } B)$
- $A \text{ OR } \text{ OR } B)$ not a Boolean formula!
- $((A \text{ AND } B) \text{ OR } (A \text{ AND } C) \text{ OR } 1)$.
- If φ and ψ are Boolean formulas, then $(\varphi \text{ OR } \psi)$ is a Boolean formula.
- If φ is a Boolean formula, then $(\neg\varphi)$ is a Boolean formula.

We will stick to parse trees, and now show how they are parsed to generate valid Boolean formulas.

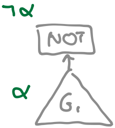
Algorithm 1 INORDER(G, π) - An algorithm for generating the Boolean formula corresponding to a parse tree (G, π), where $G = (V, E)$ is a rooted tree with in-degree at most 2 and $\pi : V \to \{0, 1\} \cup U \cup \mathcal{C}$ is a labeling function.

1. Base Case: If $|V| = 1$ then return $\pi(v)$ *(where $v \in V$ is the only node in V)*

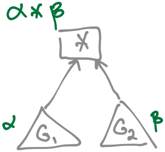
   0 0 $\times$ $\times$

   1 1

2. Reduction Rule:



   1. If $deg_{in}(r(G)) = 1$, then
      1. Let $G_1 = (V_1, E_1)$ denote the rooted tree hanging from $r(G)$.
      2. Let π_1 denote the restriction of π to V_1.
      3. $\alpha \leftarrow$ INORDER(G_1, π_1).
      4. Return $(\neg\alpha)$.



   2. If $deg_{in}(r(G)) = 2$, then
      1. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ denote the rooted subtrees hanging from $r(G)$.
      2. Let π_i denote the restriction of π to V_i.
      3. $\alpha \leftarrow$ INORDER(G_1, π_1).
      4. $\beta \leftarrow$ INORDER(G_2, π_2).
      5. Return $(\alpha \underbrace{\pi(r(G))}_{*} \beta)$.

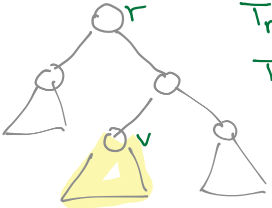
Boolean Formula

Definition

Let (G, π) denote a parse tree and let T_v denote the subtree hanging from v.

- The output φ of INORDER(G, π) is a Boolean formula.
- The output of INORDER(T_v, π) is a subformula of φ.

We say that Boolean formula φ is defined by the parse tree (G, π).



T_r defines φ

T_v defines φ'

φ' subformula of φ

Notation

- Consider all the parse trees over the set of variables U and the set of connectives $\mathcal{C}$.
- The set of all Boolean formulas defined by these parse trees is denoted by $\mathcal{BF}(U, \mathcal{C})$.
- To simplify notation, we abbreviate $\mathcal{BF}(U, \mathcal{C})$ by $\mathcal{BF}$ when the sets of variables and connectives are known.

Examples

Some of the connectives have several notations. The following formulas are the same, i.e. string equality.

$+ = \vee = \text{OR}$

$$
\begin{aligned}
(A + B) &= (A \vee B) = (A \text{ OR } B)\,,\\
(A \cdot B) &= (A \wedge B) = (A \text{ AND } B)\,,\\
(\neg B) &= (\text{NOT}(B)) = (\bar{B})\,,\\
(A \text{ XOR } B) &= (A \oplus B)\,,\\
((A \vee C) \wedge (\neg B)) &= ((A + C) \cdot (\bar{B}))\,.
\end{aligned}
$$

We sometimes omit parentheses from formulas if their parse tree is obvious. When parenthesis are omitted, one should use precedence rules as in arithmetic, e.g., $a \cdot b + c \cdot d = ((a \cdot b) + (c \cdot d))$.

The Implication Connective

The implication connective is denoted by $\rightarrow$.

X	Y	$X \rightarrow Y$
0	0	1
1	0	0
0	1	1
1	1	1

$\rightarrow$	0	1
0	1	1
1	0	1

commutative?
associative?

Table: The truth table representation and the multiplication table of the implication connective.

Lemma

$A \rightarrow B$ is true iff $A \leq B$.

more on the implication connective

$\to$	0	1
0	1	1
1	0	1

- The implication connective is not commutative, namely, $(0 \to 1) \neq (1 \to 0)$.
- This connective is called implication since it models the natural language templates "Y if X" and "if X then Y".
- Note that $X \to Y$ is always 1 if $X = 0$.

Connectives NAND NOR

$$\begin{aligned} \text{NAND}(A, B) &\triangleq \text{NOT}(\text{AND}(A, B))\,, \\ \text{NOR}(A, B) &\triangleq \text{NOT}(\text{OR}(A, B))\,. \end{aligned}$$

Truth Tables

X	Y	X NAND Y
0	0	1
1	0	1
0	1	1
1	1	0

X	Y	X NOR Y
0	0	1
1	0	0
0	1	0
1	1	0

NAND	0	1
0	1	1
1	1	0

NOR	0	1
0	1	0
1	0	0

commutative?
associative?

The Equivalence Connective

The equivalence connective is denoted by $\leftrightarrow$.

$$(p \leftrightarrow q) \text{ abbreviates } ((p \to q) \text{ AND } (q \to p)).$$

X	Y	$X \leftrightarrow Y$
0	0	1
1	0	0
0	1	0
1	1	1

$\leftrightarrow$	0	1
0	1	0
1	0	1

$$(X \leftrightarrow Y) = \begin{cases} 1 & \text{if } X = Y \\ 0 & \text{if } X \neq Y. \end{cases}$$

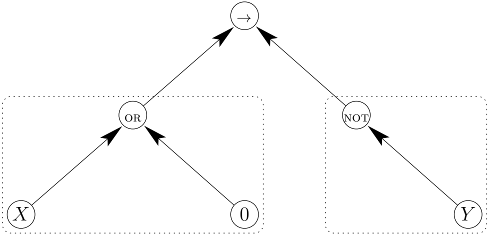


Figure: The parse tree of the Boolean formula $((X \text{ OR } 0) \to (\neg Y))$. The root is labeled by an implication connective. The rooted trees hanging from the root are encapsulated by dashed rectangles.

$(X \text{ OR } 0) \to (\neg Y)$

not "equiv" $(\neg Y) \to (X \text{ OR } 0)$

Recapping

- Variables: $X, Y, Z, \ldots$
- Logical connectives:
  - unary: NOT
  - binary: $\text{AND}, \text{OR}, \text{NOR}, \text{NAND}, \rightarrow, \leftrightarrow$
- Parse Trees: rooted tree labeled by variables and connectives.
- Boolean Formula: defined by inorder traversal of parse tree.
- Attach Boolean operators to logical connectives.