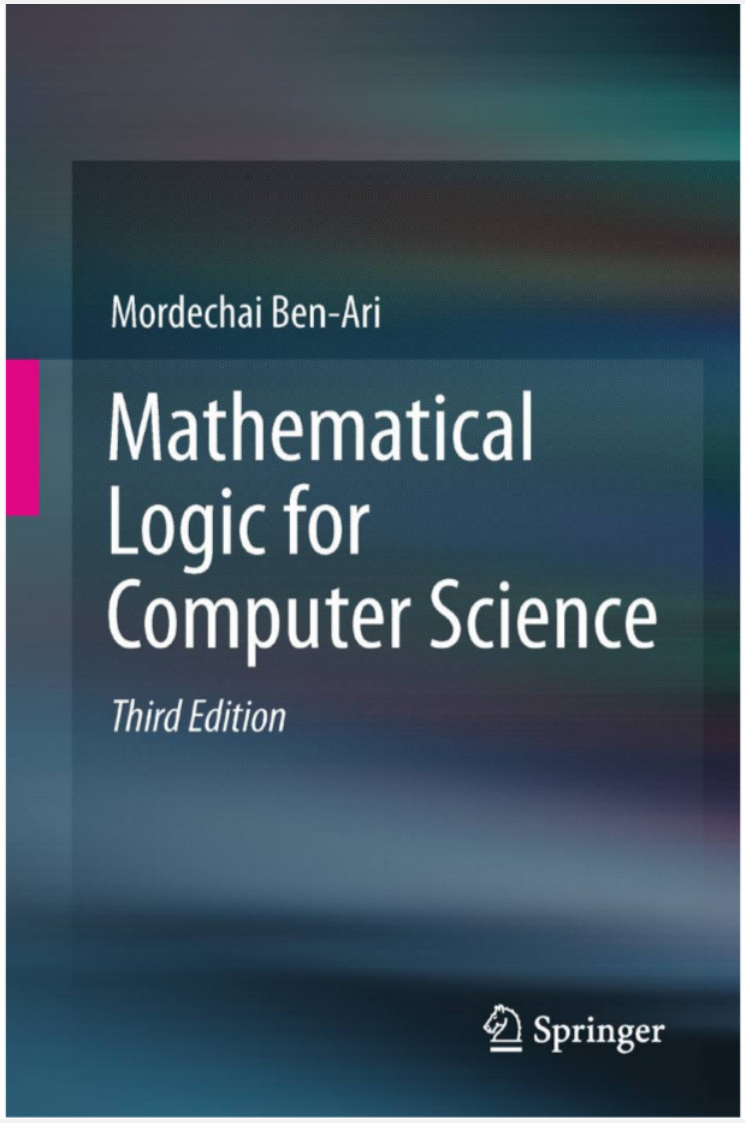
Logic and theory of computation

1st lecture

17-09-2018

Book


Mordechai Ben-Ari
Mathematical Logic for Computer Science
Third Edition
Springer

Introduction

History of logic (quotes from the introduction of the book):

- "The study of logic was begun by the *ancient Greeks* whose educational system stressed competence in reasoning and in the use of language. Along with rhetoric and grammar, logic formed part of the trivium, the first subjects taught to young people. Rules of logic were classified and named."
- "Logic must be formalized because reasoning expressed in informal natural language can be flawed."

Introduction

- "The formalization of logic began in the nineteenth century as mathematicians attempted to clarify the foundations of mathematics. One trigger was the discovery of non-Euclidean geometries: replacing Euclid's parallel axiom with another axiom resulted in a different theory of geometry that was just as consistent as that of *Euclid*."
- "During the first half of the twentieth century, logic became a full-fledged topic of modern mathematics. The framework for research into the foundations of mathematics was called Hilbert's program, (named after the great mathematician *David Hilbert*). His central goal was to prove that mathematics, starting with arithmetic, could be axiomatized in a system that was both consistent and complete."

Introduction

- "*Kurt Gödel* showed that this goal cannot be achieved: any consistent axiomatic system for arithmetic is incomplete since it contains true statements that cannot be proved within the system (1931)."
- "In the second half of the twentieth century, mathematical logic was applied in computer science and has become one of its most important theoretical foundations. Problems in computer science have led to the development of many new systems of logic that did not exist before or that existed only at the margins of the classical systems."

Introduction

Two models

We shall study two models.

I. Propositional logic (other names: propositional calculus, zeroth-order logic)

The formulas of the logic are built from atomic propositions, which are statements that have *no internal structure*. Formulas can be combined using Boolean operators (and, or, etc.). Atomic propositions can be either true or false.

Advantage: simple, close to natural languages

Disadvantage: not sufficiently expressive for formalizing mathematical theories

An application: digital circuits

Introduction

Two models

II. First-order logic (other names: predicate logic, predicate calculus)

Statements have inner structure. Allows the use of sentences that contain variables and symbols that can be interpreted as functions and relations. Variables range over a domain. Formulas can be combined using Boolean operators (and, or, etc.) and quantifiers.

Advantage: more expressive power can describe the world more accurately

Disadvantage: more difficult

An application: automated theorem proving (Prolog language)

Syntax of propositional logic

Alphabet

PROPOSITIONAL LOGIC

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a (countably) infinite set, its elements are called **atoms** (also called: atomic propositions, variables, atomic variables).
We define an **alphabet** (the set of terminals) as follows:

- elements of $\mathcal{P}$,
- Boolean operators:

negation	$\neg$	conjunction	$\wedge$
disjunction	$\vee$	implication	$\rightarrow$

- $(\ ,\ )$ (in the string representation only).

Note: sometimes further Boolean operators are used as well, such as equivalence ($\leftrightarrow$), exclusive or ($\oplus$), nor ($\downarrow$), nand ($\uparrow$).

Syntax of propositional logic

The tree representation of formulas

Formula

A formula is one of the following rooted node-labelled binary trees

- a single node (root) labelled by an atom
- a node (root) labelled by $\neg$ with a single child that is a formula
- a node (root) labelled by one of the binary operations with two children both of which are formulas

Subformula

A (proper) subformula is a (proper) subtree.

Principal operator of a formula

The operator at the root of the tree.

(Formulas, that are atoms have no principal operator).

Syntax of propositional logic

The string representation of formulas

Input: a formula F given by its tree representation
Output: string representation of F

Algorithm INORDER(F)

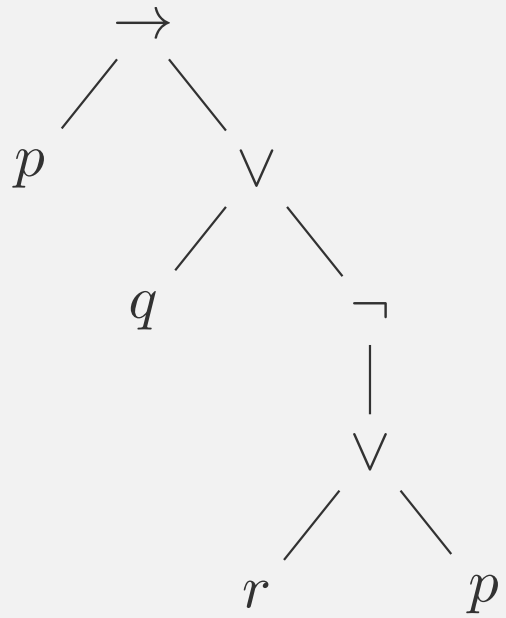
1: **if** F is a leaf **then**
2: write its label
3: return
4: let F_1 and F_2 be the left and right subtrees of F
5: write a left parenthesis '('
6: INORDER(F_1)
7: write the label of the root of F
8: INORDER(F_2)
9: write a right parenthesis ')'

If the root of F is labelled by $\neg$, the left subtree is considered to be empty and the step INORDER(F_1) is skipped.

Syntax of propositional logic

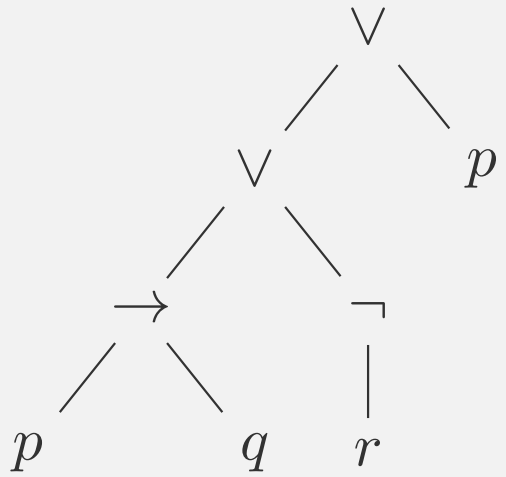
The need of parenthesis

Example:



String representation:

$(p \to (q \vee (\neg(r \vee p))))$



String representation:

$(((p \to q) \vee (\neg r)) \vee p)$

Without parenthesis there would be an ambiguity as both string representation were $p \to q \vee \neg r \vee p$.

Syntax of propositional logic

Leaving parenthesis

Another way to resolve ambiguous formulas is to define precedence and associativity conventions.

Analogy: arithmetic expression, e.g., $3 + 8 \cdot 7$.

Order of precedence from high to low: $\neg,\ \wedge,\ \vee,\ \rightarrow$.

Operators are assumed to associate to the right, that is, $A \vee B \vee C$ means $(A \vee (B \vee C))$.

Parentheses are used only if needed to indicate an order different from that imposed by the rules for precedence and associativity.

Examples, minimum number of parenthesis:

$$(p \rightarrow (q \vee (\neg(r \vee p)))) \quad \Rightarrow \quad p \rightarrow q \vee \neg(r \vee p)$$
$$(((p \rightarrow q) \vee (\neg r)) \vee p) \quad \Rightarrow \quad ((p \rightarrow q) \vee \neg r) \vee p$$

Semantics of propositional logic

Interpretation

Analogy: arithmetic expressions. E.g., $y = a + 2 \cdot b$. Assign values to a and b then evaluate y.

Interpretation

Let A be a formula and let P_A be the set of atoms appearing in A. An interpretation for A is a total function $I_A : P_A \to \{T, F\}$ that assigns one of the truth values T or F to every atom in P_A.

We will use the short notation I instead of I_A whenever the formula is clear from the context. Example:

$A = p \to q \vee \neg(r \vee p)$. $P_A = \{p, q, r\}$. One possibility is the following: $I(p) = T,\ I(q) = F,\ I(r) = F$.
There are 8 possibilities for I in this case.

Semantics of propositional logic

Evaluation of a formula

Truth value of a formula

Let I be an interpretation for a formula A. $v_I(A)$, the truth value of A under I is defined recursively as follows:

$v_I(A) = I(A)$	if A is an atom
$v_I(\neg A) = T$	if $v_I(A) = F$
$v_I(\neg A) = F$	if $v_I(A) = T$
$v_I(A_1 \wedge A_2) = T$	if $v_I(A_1) = T$ and $v_I(A_2) = T$
$v_I(A_1 \wedge A_2) = F$	in the other three cases
$v_I(A_1 \vee A_2) = F$	if $v_I(A_1) = F$ and $v_I(A_2) = F$
$v_I(A_1 \vee A_2) = T$	in the other three cases
$v_I(A_1 \rightarrow A_2) = F$	if $v_I(A_1) = T$ and $v_I(A_2) = F$
$v_I(A_1 \rightarrow A_2) = T$	in the other three cases

Semantics of propositional logic

Example for an evaluation

We will use the short notation v instead of v_I or v_{I_A} whenever the formula A and the interpretation I is clear from the context.

Example:

$A = p \to q \vee \neg(r \vee p)$.

$P_A = \{p, q, r\}$.

$I(p) = T,\ I(q) = F,\ I(r) = F$.

$v(r \vee p) = v(r) \vee v(p) = F \vee T = T$.

$v(\neg(r \vee p)) = \neg v(r \vee p) = \neg T = F$.

$v(q) = F$, $v(\neg(r \vee p)) = F$, so $v(q \vee \neg(r \vee p)) = F$

$v(p) = T$, $v(q \vee \neg(r \vee p)) = F$, so $v(A) = F$.

Semantics of propositional logic

Truth table

We may be interested in the truth value for every possible interpretation of a formula.

Let A be a formula and suppose, that $|P_A| = n$.

Since each of the n atoms can be assigned T or F independently, there are 2^n possible interpretations.

Truth table

A truth table for a formula A is a table with $n + 1$ columns and 2^n rows, where $n = |P_A|$. There is a column for each atom in P_A , plus a column for the formula A. The first n columns specify the interpretation I that maps atoms in P_A to $\{T, F\}$. The last column shows $v_I(A)$, the truth value of A for the interpretation I.

Semantics of propositional logic

Example for truth table

Example:

Let us make the truth table for $A = p \to q \vee \neg(r \vee p)$.

p	q	r	$r \vee p$	$\neg(r \vee p)$	$q \vee \neg(r \vee p)$	$p \to q \vee \neg(r \vee p)$
T	T	T	T	F	T	T
T	T	F	T	F	T	T
T	F	T	T	F	F	F
T	F	F	T	F	F	F
F	T	T	T	F	T	T
F	T	F	F	T	T	T
F	F	T	T	F	F	T
F	F	F	F	T	T	T

Semantics of propositional logic

Semantic properties of formulas

Let A be a formula

- A is **satisfiable** iff $v_I(A) = T$ for some interpretation I .
- A satisfying interpretation I is a **model** for A, denoted $I \vDash A$.
- A is **valid**, denoted $\vDash A$, iff $v_I(A) = T$ for all interpretations I . A valid propositional formula is also called a **tautology**.
- A is **unsatisfiable** iff it is not satisfiable, that is, if $v_I(A) = F$ for all interpretations I.
- A is **falsifiable**, denoted $\nvDash A$, iff it is not valid, that is, if $v_I(A) = F$ for some interpretation I.

Semantics of propositional logic

Semantic properties of formulas, examples

Let A_1, A_2 be formulas.

If $v_I(A_1) = v_I(A_2)$ for all interpretations I , then A_1 is **logically equivalent** to A_2 , denoted $A_1 \equiv A_2$.

Examples:

- Let $A = p \to q \vee \neg(r \vee p)$.

  Then A is satisfiable, falsifiable, interpretation TTT is a model for A. (See its truth table.)

  On the other hand A is not valid and not unsatisfiable.

- formula $p \vee \neg p$ is valid, on the other hand $p \wedge \neg p$ is unsatisfiable.

- $p \vee q$ and $q \vee p$ are logically equivalent formulas.