中国科学院大学
University of Chinese Academy of Sciences

CS101

Logic Thinking

Propositional Logic: Concepts

zxu@ict.ac.cn
zhangjialin@ict.ac.cn

Outline

- Foundation of logic
  - Propositional Logic
  - Predicative Logic
- Automata and Turing Machines
- Power and Limitation of Computing
  - Mechanical Theorem Proving
  - Church-Turing Hypothesis

1.1 PROPOSITIONAL LOGIC

Three motivating examples of Boolean logic

- Boolean logic
  - A logic system to reason about true-or-false statements
  - Manifests in propositional logic and predicate logic
  - A perfect match for computer science
    - CS uses binary values of 0 and 1 to represent digital symbols
- Three examples
  - The Congruent Triangles Problem
    - Boolean logic can be used to solve mathematic problems
      - without using mathematic domain knowledge
  - The Impatient Guide Problem
    - Application problems can be encoded as Boolean logic problems
  - The Adder Implementation Problem
    - Computer hardware can be implemented as Boolean expressions

The congruent triangle problem

- Consider the following propositions
  - *P*: “two triangles are congruent”; Q: “two triangles are similar”
  - Original: $P \to Q$.
  - If two are congruent triangles, they are similar.
  - Inverse: $\neg P \to \neg Q$

    If two are not congruent triangles, they are not similar.
  - Converse: $Q \to P$

    If two are similar triangles, they are congruent.
  - Contrapositive: $\neg Q \to \neg P$

    If two are not similar triangles, they are not congruent.
- A proposition and its contrapositive are equivalent.
- The inverse and the converse are equivalent.
- Why?

The impatient guide problem

- Problem context
  - A tourist is traveling in the land of Oz and wants to go to the Emerald City. The tourist reaches a crossroad with paths P and Q, one of which leads to the Emerald City. There is a guide G at the crossroad, who comes from either the Honest Village or the Lying Village. Anyone from the Honest Village always tells the truth, and anyone from the Lying Village always tells lies. The guide is impatient, in that G only answers one question from the tourist, and the answer is either "Yes" or "No".
- Q: What question should the tourist ask the guide, to determine the correct path?

The adder implementation problem

- Realizing addition of two *n*-bit numbers with Boolean logic operators
- Design an adder, which takes two *n*-bit numbers *X* and *Y* as inputs and produce an *n*-bit number *Z* as the output
  - $(x_n \dots x_1)_2 + (y_n \dots y_1)_2 + c_0 = (c_n z_n \dots z_1)_2$
- Here,
  - c_0 is the least significant carry bit
  - c_n is the most significant carry bit
- Only Boolean logic operators can be used
  - What are Boolean logic operators?

Proposition

- A proposition is a true-or-false statement
  - Can be either a primitive proposition or a combinational proposition
- A primitive proposition is a proposition without logic connectives, such as
  - “It rains today”; “Earth is flat”
- A combinational proposition is a proposition with one or more ***logic connectives***, such as
  - “If Earth is flat, then it rains today”
  - “Two triangles are either congruent or similar”
- Propositional logic
  - Does not care about the truthiness of primitive propositions
  - It cares about connectives, i.e., how connectives influence the truthiness of combinational propositions

Logic Connectives

- True (T, 1), False (F, 0)
  - Today is Friday
  - $a^2 \geq 0$
- Negation, Not: $\neg$ (also
  - $\neg x = 1$ (T) iff $x = 0$ (F)
  - $\bar{x} = \neg x$
- Conjunction, And: $\wedge$
  - $x \wedge y = 1$ (T) iff $x = y = 1$ (T)
- Disjunction, Or: $\vee$
  - $x \vee y = 0$ (F) iff $x = y = 0$ (F)

Logic connectives (2)

- Exclusive or: $\oplus$
  - $x \oplus y = 1$ (T) iff $x \neq y$
  - $x + y \; mod \; 2$
- Implication: $\rightarrow$
  - $(x \rightarrow y) = 1$ (T) iff $x = 0$ (F) or $y = 1$ (T)
  - If two are congruent triangles, they are similar.
    - $x =$ two triangles are congruent
    - $y =$ two triangles are similar
    - $x \rightarrow y$

Thinking problems

- Original: $P \rightarrow Q$
- Inverse: $\neg P \rightarrow \neg Q$
- Converse: ?
- Contrapositive: ?

Summary and comparison of the five logic connectives

- **Conjunction** ∧ (also called **AND**): $x \land y = 1$ if and only if $x = y = 1$.
  For example, the solution of $x^2 + 2\mathrm{x} < 0$ is $x > -2$ and $x < 0$. We can describe it as $(x > -2) \land (x < 0)$.
- **Disjunction** ∨ (also called **OR**): $x \lor y = 0$ if and only if $x = y = 0$.
  For example, the solution of $x^2 + 2\mathrm{x} \geq 0$ satisfies $x \leq -2$ or $x \geq 0$. We can describe it as $(x \leq -2) \lor (x \geq 0)$.
- **Negation** ¬ (also called **NOT**): $\neg x = 0$ if and only if $x = 1$.
  We also use $\bar{x}$ to represent the negation of x, that is, $\bar{x} = \neg x$.
- **Implication** →: $(x \to y) = 1$ if and only if $x = 0 \text{ or } y = 1$.
  We call x **premise** and y **conclusion**. Implication means $x \to y = 1$ if and only if the premise is false or the conclusion is true.
- **Exclusive-or** (also called **XOR**) ⨁: $x \oplus y = 1$ if and only if $x \neq y$.
  That is, $x \oplus y$ is true iff either *x* or *y* (but not both) is true. Note $x \oplus 1 = \neg x$, that is, we can use exclusive-or operation to realize negation.

Propositions can be precisely stated

- Any proposition can be precisely stated (represented) in one of three ways
  - By a Boolean function
    - $f: \{0,1\}^n \rightarrow \{0,1\}$
    - **Parity** of an *n*-bit number X: X's bits have an odd number of 1's
  - By a truth table
    - An exhaustive way to represent a Boolean function
    - When n = 3, shown by the right table
  - By a Boolean expression
    - Parity(X) = $X_0 \oplus X_1 \oplus X_2 \oplus \ldots \oplus X_{63}$
      - Much more compact than the truth table
- A key property: the above three representations are equivalent

X_2	X_1	X_0	Parity
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

We can also combine multiple truth tables in one table

- Four Boolean functions AND, OR, Imply, XOR in one truth table

x	y	$x \wedge y$	$x \vee y$	$x \rightarrow y$	$x \oplus y$
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	1	0

Boolean Function

- $f: \{0,1\}^n \to \{0,1\}$
  - An *n*-input-1-output Boolean function is a mapping from the domain $\{0,1\}^n$ to the range $\{0,1\}$
- Examples
  - $f(x_1, x_2) = x_1 \wedge x_2$
  - $g(x_1, x_2, \cdots, x_n) = (\vee_{i=1}^{n-1} x_i) \oplus x_n$
  - $h(x_1, x_2, \cdots, x_n) = x_1$
- Same Boolean function: same truth table
  - Similar to: $f(x, y) = x^2 - y^2$ and $g(x, y) = (x + y)(x - y)$ are the same polynomials
  - $f(x, y) = x \to y, g(x, y) = \neg x \vee y$
    - They are the same Boolean functions

Boolean expression

- When logic connectives = Boolean logic operators, propositions become Boolean expressions
  - Both primitive and combinational
- The set *L* of all Boolean expressions
  - Recursively defined
    - Initially, let **$0, 1, x_1, \ldots, x_n \in L$**
      - Here, $x_1, \ldots, x_n$ are primitive propositions for some natural number *n*
      - 0 and 1 are *Boolean constants*
      - $x_1, \ldots, x_n$ are *Boolean variables*
    - Recursively apply negation, conjunction, disjunction, implication, and exclusive-or operations to *L*:
      If $x, y \in L$, then $\neg x, x \wedge y, x \vee y, x \rightarrow y, x \oplus y \in L$
      - Combine equivalent expressions (discussed later)
    - Repeat until *L* does not change any more

Logic connectives viewed as operators

- Negation, Not: $\neg$ (also denoted as $\overline{\boldsymbol{x}}$)
  - $\neg x = 1$ (T) iff $x = 0$ (F)
  - $\neg x = \bar{x}$
- Conjunction, And: ∧ (logic multiply)
  - $x \wedge y = 1$ (T) iff $x = y = 1$ (T)
  - $x \wedge y = \boldsymbol{x \cdot y}$
- Disjunction, Or: ∨ (logic add)
  - $x \vee y = 0$ (F) iff $x = y = 0$ (F)
  - $x \vee y = \boldsymbol{x + y}$