COSE215: Theory of Computation

Lecture 1 — Mathematical Preliminaries

Hakjoo Oh
2019 Spring

Contents

- Logical notations
- Basic set theory
- Language
- Inductive proofs

Notations in Logic

- A, B: arbitrary statements.
- $P(x)$: a statement that involves variable x.
- $A \wedge B$: the conjunction of A and B
- $A \vee B$: the disjunction of A and B
- $A \implies B$: if A then B
- $A \iff B$: A if and only if (iff) B, i.e., $A \implies B \wedge B \implies A$
- $\forall x.P(x)$: for all x, $P(x)$
- $\forall x \in X.P(x)$: $\forall x.x \in X \implies P(x)$
- $\exists x.P(x)$: there exists x such that $P(x)$
- $\exists x \in X.P(x)$: $\exists x.x \in X \wedge P(x)$

Sets

- A set is a collection of elements, e.g.,
  - $\mathbb{N} = \{0, 1, 2, \ldots\}$
  - $\{x \mid P(x)\}$: a set determined by a property P
  - $\{x \in X \mid P(x)\}$: $\{x \mid x \in X \wedge P(x)\}$
  - $S = \{0, 1, 2\} = \{x \in \mathbb{N} \mid 0 \leq x \leq 2\}$
  - $S = \{2, 4, 6, \ldots\} = \{x \in \mathbb{N} \mid x \text{ is even}\}$
- Notations:
  - $\emptyset$: the empty set
  - $S_1 \subseteq S_2$ iff $\forall x \in S_1.\ x \in S_2$
  - $S_1 \subset S_2$ if $S_1 \subseteq S_2$ and $S_1 \neq S_2$
    - e.g., $\{1, 2\} \subset \{1, 2, 3\}$, $\{1, 2\} \not\subset \{1, 2\}$
  - $|S|$: the number of elements in set S
  - S_1 and S_2 are disjoint iff $S_1 \cap S_2 = \emptyset$.

Construction of Sets

- Union, intersection, and difference:

$$\begin{aligned} S_1 \cup S_2 &= \{x \mid x \in S_1 \vee x \in S_2\} \\ S_1 \cap S_2 &= \{x \mid x \in S_1 \wedge x \in S_2\} \\ S_1 - S_2 &= \{x \mid x \in S_1 \wedge x \notin S_2\} \end{aligned}$$

- Let X be a set of sets ($X = \{A_1, A_2, \ldots, A_n\}$).

$$\begin{aligned} \bigcup X &= A_1 \cup A_2 \cup \cdots \cup A_n = \{a \mid \exists A \in X. a \in A\} \\ \bigcap X &= A_1 \cap A_2 \cap \cdots \cap A_n = \{a \mid \forall A \in X. a \in A\} \end{aligned}$$

- Let $A_1, A_2, \ldots, A_n$ be sets.

$$\bigcup_{1 \leq i \leq n} A_i = A_1 \cup \cdots \cup A_n, \qquad \bigcap_{1 \leq i \leq n} A_i = A_1 \cap \cdots \cap A_n$$

- $\overline{S} = \{x \mid x \in U \wedge x \notin S\}$ (U: universe)
- Powerset: $2^S = \mathcal{P}(S) = \{x \mid x \subseteq S\}$
- Cartesian product: $S_1 \times S_2 = \{(x, y) \mid x \in S_1 \wedge y \in S_2\}$. In general, $S_1 \times S_2 \times \cdots \times S_n = \{(x_1, x_2, \ldots, x_n) \mid x_i \in S_i\}$

Partition

When $S_1, S_2, \ldots, S_n$ are subsets of a given set S, $S_1, S_2, \ldots, S_n$ forms a partition of S iff:

1. $S_1, S_2, \ldots, S_n$ are mutually disjoint:

$$\forall i, j.\ i \neq j \implies S_i \cap S_j = \emptyset$$

2. $S_1, S_2, \ldots, S_n$ cover S:

$$\bigcup_{1 \leq i \leq n} S_i = S$$

3. none of S_i is empty: $\forall i. S_i \neq \emptyset$.

Alphabet

A finite, non-empty set of symbols, e.g.,

1. $\mathbf{\Sigma = \{0, 1\}}$: the binary alphabet.
2. $\boldsymbol{\Sigma = \{a, b, \ldots, z\}}$: the set of all lowercase letters.
3. The set of all ASCII characters.

String

A finite sequence of symbols chosen from an alphabet, e.g.,

1. $\Sigma = \{0, 1\}$: 0, 1, 00, 01, $\ldots$
2. $\Sigma = \{a, b, c\}$: a, b, c, ab, bc, $\ldots$

Notations:

- ϵ: the empty string
- wv: the concatenation of w and v
- w^R: the reverse of w
- $|w|$: the length of string w
- $w = vu$: v is a prefix and u a suffix of w.
- Σ^k: the set of strings (over Σ) of length k
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots = \bigcup_{k \geq 0} \Sigma^k$
- $\Sigma^+ = \Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \cdots = \bigcup_{k \geq 1} \Sigma^k$

Language

A language L is a set of strings, i.e., $L \subseteq \Sigma^*$ ($L \in 2^{\Sigma^*}$)

When $\Sigma = \{0, 1\}$,

- $L_1 = \{0, 00, 001\}$
- $L_2 = \{0^n 1^n \mid n \geq 0\}$
- $L_3 = \{\epsilon, 01, 10, 0011, 0101, 1001, \ldots\}$
- $L_3 = \{10, 11, 101, 111, 1011, \ldots\}$

Language Operations

- union, intersection, difference: $L_1 \cup L_2, \quad L_1 \cap L_2, \quad L_1 - L_2$
- reverse: $L^R = \{w^R \mid w \in L\}$
- complement:$\overline{L} = \Sigma^* - L$
- concatenation of L_1 and L_2:

$$L_1 L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$$

- power:

$$\begin{array}{rcl} L^0 & = & \{\epsilon\} \\ L^n & = & L^{n-1}L \end{array}$$

- closures:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{i \geq 0} L^i$$

$$L^+ = L^1 \cup L^2 \cup L^3 \cup \cdots = \bigcup_{i \geq 1} L^i$$

Exercises

1. Consider $L = \{a^n b^n \mid n \geq 0\}$.
   1. $L^2 =$
   2. $L^R =$
2. Prove that $(uv)^R = v^R u^R$ for all $u, v \in \Sigma^+$.

Inductive proofs

In CS, set is usually defined inductively.

Example (Inductive Definition of Trees)

A set of trees is defined as follows:

1. (Basis) A single node (called root) is a tree.
2. (Induction) If $T_1, T_2, \ldots, T_k$ are trees, then the following is also a tree:
    1. Begin with a new node N, which is the root of the tree.
    2. Add edges from N to the roots of each of the trees $T_1, T_2, \ldots, T_k$.

Example (Inductive Definition of Arithmetic Expressions)

A set of arithmetic expressions is defined as follows:

- (Basis) Any number or letter (i.e., a variable) is an expression.
- (Induction) If E and F are expressions, then so are $E + F$, $E * F$, and (E).

Inductive Proofs

Induction is used to prove properties about inductively defined sets. Let S be an inductively-defined set. Let $P(x)$ be a property of x. To show that, for all $x \in S.P(x)$, it suffices to show that:

1. (Base case): Show $P(x)$ for all basis elements $x \in S$.
2. (Inductive case): For each inductive rule using elements $x_1, \ldots, x_k$ of S to construct an element x, show that

$$\text{if } P(x_1), \ldots, P(x_k) \text{ then } P(x)$$

$P(x_1), \ldots, P(x_k)$: induction hypotheses.

Inductive Proofs: Example

Prove that every tree has one more node than it has edges.

Proof.

Formally, what we prove is $P(T) =$ "if T is a tree, and T has n nodes and e edges, then $n = e + 1$".

1. Base case: The base case is when T is a single node. Then, $n = 1$ and $e = 0$, so the relationship $n = e + 1$ holds.
2. Inductive case: The inductive case is when T is built with root node N and k smaller trees $T_1, T_2, \ldots, T_k$.
   1. **Induction hypothesis:** The statements $P(T_i)$ holds for $i = 1, 2, \ldots, k$. That is T_i have n_i nodes and e_i edges; then $n_i = e_i + 1$.
   2. **To Show:** $P(T)$ holds: if T has n nodes and e edges, then $n = e + 1$. The nodes of T are node N and all the nodes of the T_i's, i.e., $n = 1 + n_1 + \cdots + n_k$ The edges of T are the k edges we added explicitly in the inductive definition step, plus the edges of the T_i's. Hence, T has $e = k + e_1 + \cdots + e_k$ edges.

$$\begin{aligned} n &= 1 + n_1 + \cdots + n_k && \text{def. of } n \\ &= 1 + (e_1 + 1) + \cdots + (e_k + 1) && \text{induction hypothesis} \\ &= 1 + k + e_1 + \cdots + e_k \\ &= 1 + e && \text{def. of } e \end{aligned}$$

□

Inductive Proofs: Example

Prove that every expression has an equal number of left and right parentheses.

Proof.

Formally, the formal statement $P(G)$ we need to prove is: "if G has l left parentheses and r right parentheses, then $l = r$."

1. **Base case:** The base case is when G is a number or a variable, in which cases $l = r = 0$.
2. **Inductive case:** There are three cases, where G is constructed recursively from smaller expressions:
   - $G = E + F$:
     1. **Induction hypothesis:** The statement holds for all smaller expressions: for E, $l_E = r_E$, and for F, $l_F = r_F$.
     2. **To Show**: $P(G)$ holds: $l_G = r_G$:

$$\begin{aligned} l_G &= l_E + l_F \\ &= r_E + r_F \quad \text{I.H.} \\ &= r_G \end{aligned}$$

   - $G = E * F$: similar
   - $G = (E)$: similar

□

Summary

- Sets: definition, notations, constructions
- Alphabet, String, Language
- Inductive definitions and proofs.