CS3231 - The Theory of Computing

Yadunand Prem

1. Chapter 1

1.1. Deductive Proofs

Sequence of statements, whose truth leads us from some initial statement, the *hypothesis* to *conclusion*. Each step in the proof must follow by some accepted logical principle, either from facts or some of the previous statements in the deductive proof.

Theorem 1.1.1: If $x \geq 4$ then $2^x \geq x^2$

Proof: ■

Theorem 1.1.2: If x is the sum of the squares of 4 positive integers, then $2^x \geq x^2$

Proof:
1. $x = a^2 + b^2 + c^2 + d^2$ (Given)
2. $a \geq 1, b \geq 1, c \geq 1, d \geq 1$ (Given)
3. $a^2 \geq 1, b^2 \geq 1, c^2 \geq 1, d^2 \geq 1$ ((2) and properties of arithmetic)
4. $x \geq 4$ ((1), (3), and properties of arithmetic
5. $2^x \geq x^2$ ((4) and Theorem 1.3) ■

1.2. Proof by Contradiction

Another way to prove statements of the form "if H then C" is to prove "H and not C implies falsehood".

We can start by assuming both hypothesis H and negation of conclusion C. Complete proof by showing that something known to be false follows logically from H and $\text{not } C$

1.3. Proofs about sets

1.3.1. Equivalences

To prove the equality of to sets, E and F, $E = F$, we need to prove the following.
1. Proof that if x is in E, then x is in F.
2. Proof that if x is in F, then x is in E.

Theorem 1.3.1.1: $R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$

Proof: The two set expressions involved are $E = R \cup (S \cap T)$ and $F = (R \cup S) \cap (R \cup T)$. In the *if* part, we assume element x is in E and show it is in F.

1. x is in $R \cup (S \cap T)$ (given)
2. x is in R or x is in $S \cap T$ (defn of union)

3. x is in R or x is in both S and T (defn of intersection)
4. x is in $R \cup S$ (defn of union)
5. x is in $R \cup T$ (defn of union)
6. x is in $(R \cup S) \cap (R \cup T)$ (4, 5, defn of intersection)

In the *only if* part, we assume element x is in F and show it is in E.

1. x is in $(R \cup S) \cap (R \cup T)$ (Given)
2. x is in $(R \cup S)$ (defn of intersection)
3. x is in $(R \cup T)$ (defn of intersection)
4. x is in R or x is in both S and T (2, 3, reasoning about unions)
5. x is in R or x is in $S \cap T$ (defn of intersection)
6. x is in $R \cup (S \cap T)$ (defn of union)

■

1.4. Inductive Proofs

Suppose we are given statement $S(n)$ about an integer n to prove. We need to prove 2 things.

1. The *basis*, where we show $S(i)$ for a particular integer i. Usually $i = 0$ or $i = 1$.
2. The *inductive* step, where we assume $n \geq i$, where i is the basis integer, and we show that "if $S(n)$ then $S(n + 1)$"

These 2 parts should convince us that $S(n)$ is true for every integer n that is equal to or greater than basis integer i.

2. Central Concept of Automata Theory

2.1. Alphabet

Alphabet is a finite, nonempty set of symbols. We use the symbol Σ for alphabet. Common alphabets include

1. $\Sigma = \{0, 1\}$ the binary alphabet
2. $\Sigma = \{a, b, ..., z\}$ the set of all lowercase letters

2.2. String

String is a finite sequence of symbols chosen from some alphabet. 01101 is a string from the binary alphabet.

Empty String is the string with 0 occurances of symbols, denoted by ϵ. This string is a string that may be chosen from any alphabet whatsoever

Length of String is denoted by $|w|$, where w is a string

Powers of an Alphabet If Σ is an alphabet, we can express the set of all strings of a certain length. We define Σ^k to be the set of strings of length k, whose embols is in Σ. Note that $\Sigma^0 = \{\epsilon\}$

The set of all strings over an alphabet Σ is denoted Σ^*. For instance, $\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, ...\}$ The set of nonempty strings is denoted by Σ^+ The set of nonempty strings is denoted by Σ^+

- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup ...$
- $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$

Concat of Strings Let x and y be strings. xy denotes concatenation of x and y. If x is a string composed of i symbols $x = a_1a_2...a_i$ and y is the string composed of j symbols $y = b_1b_2...b_j$ then xy is the string of length $i + j : xy = a_1...a_ib_1...b_j$.

2.3. Languages

A set of strings all of which are chosen from some Σ^*, where Σ is a particular alphabet, is called a language. If Σ is an alphabet, and $L \subseteq \Sigma^*$, then L is a language over Σ. Language over Σ need not include strings with all the symbols of Σ, so once we have established than L is a language over Σ, we also know it is a language over any alphabet that is a superset of Σ.

Example languages:

- Σ^* is a language for any alphabet Σ.
- $\emptyset$,the empty language, is a language over any alphabet
- $\{\epsilon\}$, the language consisting of only empty string.

Only constraint on what can be a language is that all alphabets are finite. Languages can have an infinite number of strings, but are restricted to consist of strings drawn from one fixed, finite alphabet.

3. Deterministic Finite Automata

DFA consists of

1. Finite set of states, often denoted Q
2. Finite set of input symbols,often denoted Σ
3. A transition function that takes as arguments a state, and an input symbol, and returns a state. Commonly denoted by δ
4. A start state, one of the states in Q
5. A set of final or accepting states F. $F \subset Q$

In proofs we often talk about DFA in "5 tuple" notation:

$$A = (Q, \Sigma, \delta, q_0, F)$$

where A is the name of the DFA.

3.1. DFA Processes Strings

The Language of a DFA: The set of all strings that result in a sequence of state transitions from the *start* state to an *accepting* state.

We define an *extended transition function* to describe what happens when we start in any state, and follow a *sequence* of inputs, denoted as $\hat{\delta}$. The extended transition function is a function that takes state q and a string w and returns state p - the state automaton reaches when starting in state q and processing the sequence of input w. We define $\hat{\delta}$ by induction on the length of input string, as follows:

Definition 3.1.1 () : **Extended Transition Function**

Basis: $\hat{\delta}(q, \epsilon) = q$. That is if we are in state q and read no inputs, we are still in state q.

Induction: Suppose w is a string of the form xa, that is a is the last symbol of w and x is the string consisting of all but the last symbol. Then

$$\hat{\delta}(q, w) = \delta\Big(\hat{\delta}(q, x), a\Big)$$

3.2. Language of DFA

The language of a DFA $A = (Q, \Sigma, \delta, q_0, F)$, denoted by $L(A)$ is defined by

$$L(A) = \Big\{w \mid \hat{\delta}(q_0, w) \in F\Big\}$$

That is, the language of A is the set of strings w that take the start state q_0 to one of the accepting states.

If L is $L(A)$ for some DFA A, then L is a **regular language**

Theorem 3.2.1: For any state q and string x and y, $\hat{\delta}(q, xy) = \hat{\delta}\Big(\hat{\delta}(q, x), y\Big)$

Proof: By inducting on $|y|$

Base case: $(y = \epsilon)$:

$\hat{\delta}(q, x\epsilon) = \hat{\delta}(q, x)$ and $\hat{\delta}\Big(\hat{\delta}(q, x), \epsilon\Big) = \hat{\delta}(q, x)$

Inductive Step: Assume the statement holds for some $y = w \in \Sigma^*$, i.e. $\hat{\delta}(q, xw) = \hat{\delta}\Big(\hat{\delta}(q, x), w\Big)$

Let $y = wa, a \in \Sigma$, we have

1. $\hat{\delta}(q, xwa) = \delta\Big(\hat{\delta}(q, xw)a\Big)$ (defn of $\hat{\delta}$)
2. $= \delta\Big(\hat{\delta}\Big(\hat{\delta}(q, x), w\Big), a\Big)$ (Apply IH)
3. $= \hat{\delta}\Big(\hat{\delta}(q, x), wa\Big)$ (defn of $\hat{\delta}$)

So the statement holds true for wa. ■

Theorem 3.2.2: For any state q, string x and symbol a, $\hat{\delta}(q, ax) = \hat{\delta}(\delta(q, a), x)$

Proof:

1. Let x = a and y = x. Then, $\hat{\delta}(q, ax) = \hat{\delta}\Big(\hat{\delta}(q, a), x\Big)$.
2. $= \hat{\delta}(\delta(q, a), x)$ (by defn of $\hat{\delta}$)

■

4. Nondeterministic Finite Automata

NFA has a set of finite states, finite input symbols, 1 start and a set of accepting states. NFA's transition function takes a state and input symbols but returns a **set** of 0, 1, or more states.

Definition 4.1 (Nondeterministic Finite Automata) :

$$A = (Q, \Sigma, \delta, q_0, F)$$

, where

1. Q is a finite set of states
2. Σ is a finite set of symbols
3. $q_0 \in Q$, is the start state
4. $F \subset Q$, set of final states
5. δ, the transition function, takes in a state in Q and an input symbol in Σ and returns a subset of Q.

4.1. Extended Transition function

function $\hat{\delta}$ takes a state q, and a string of input symbols w, and returns the set of states that the NFA is in if it starts in state q and processes string w.

Definition 4.1.1 (Extended Transition Function for NFA) :

Basis: $\hat{\delta}(q, \epsilon) = \{q\}$. That is, without reading any input symbols, we are only in the state we began in.

Induction: Suppose w is of the form $w = xa$, where a is the final symbol of w and x is the rest of w. Also suppose that $\hat{\delta}(q, x) = \{p_1, ..., p_k\}$. Let

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, ..., r_m\}$$

Then $\hat{\delta}(q, w) = \{r_1, r_2, ..., r_m\}$

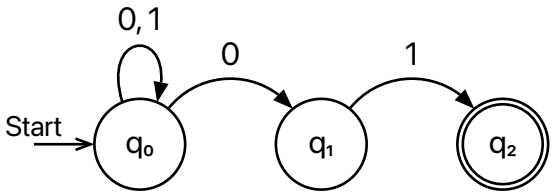
4.2. Language of NFA

NFA accepts string w if it is possible to make any sequence of choices of next state, while reading characters of w, and go from start state to any accepting state. If $A = (Q, \Sigma, \delta, q_0, F)$ is an NFA, then

$$L(A) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset$$

That is, $L(A)$ is the set of strings w in Σ^* such that $\hat{\delta}(q_0, w)$ contains at least 1 accepting state

Example 4.2.1:



Prove formally that this NFA accepts language $L = \{w \mid w \text{ ends in } 01\}$

Proof: The following 3 statements characterisze the 3 states:

1. $\hat{\delta}(q_0, w)$ contains q_0 for every w
2. $\hat{\delta}(q_0, w)$ contains q_1 if and only if w ends in 0.
3. $\hat{\delta}(q_0, w)$ contains q_2 if and only if w ends in 01.

We prove by induction on $|w|$.

Basis: If $|w| = 0, \text{then } w = \epsilon$.

- Statement (1) says that $\hat{\delta}(q_0, \epsilon)$ contains q_0, by defn of $\hat{\delta}$.
- Statement (2), we know that ϵ does not end in 0, and $\hat{\delta}(q_0, \epsilon)$ does not contain q_1 by defn of $\hat{\delta}$
- Statement (3), same as statement 2.

Induction: Assume $w = xa$, where a is a symbol either in 0 or 1. We assume statements 1-3 hold for x, and we need to prove them for w, that is, we assume $|w| = n + 1, |x| = n$.

1. $\hat{\delta}(q_0, x)$ contains q_0. Since there are transitions from 0/1 from q_0 to itself, it follows that $\hat{\delta}(q_0, w)$ also contains q_0, so statement 1 is proved for w
2. (If) Assume w ends in 0, i.e. $a = 0$. By statement (1) applied to x, we know that $q_0 \in \hat{\delta}(q_0, x)$. Since there are transitions from q_0 to q_1 on input 0, we know that $q_1 \in \hat{\delta}(q_0, w)$.
   (Only-if) Assume $q_1 \in \hat{\delta}(q_0, w)$. Only way to get to q_1 is if $w = x0$.
3. (If) Assume w ends in 01. If $w = xa$, then $a = 1$ and x ends in 0. By statement 2 applied to x, we know that $q_1 \in \hat{\delta}(q_0, x)$. Since there is a transition from q_1 to q_2 on input 1, we conclude that $q_2 \in \hat{\delta}(q_0, w)$
   (Only-if) Suppose $q_2 \in \hat{\delta}(q_0, w)$. Only way to get to q_2 is for w to be of the form $x1$, where $q_1 \in \hat{\delta}(q_0, w)$. By (2) applied to x, we know that x ends in 0. Thus, w ends in 01.

■

4.3. Equivalence of DFA and NFA

We prove this using subset construction. We start with NFA $N = (Q_N, \Sigma, \delta_N, Q_0, F_N)$. The goal is the description of a DFA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L(D) = L(N)$. The input alphabets are the same, and the start of D is the set containing only the start state of N.

- Q_D is the set of subsets of Q_N, that is Q_D is the power set of Q_N. If Q_N has n states, Q_D has 2^n states.

- F_D is the set of subsets S of Q_N such that $S \cap F_N \neq \emptyset$. F_D is all sets of N's states that include at least 1 accepting state of N.
- For each set $S \subseteq Q_N$, and for each input symbol a in Σ,

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

Theorem 4.3.1: If $D = (Q_d, \Sigma, \delta_D, \{q_0\}, F_D)$ is the DFA constructed from NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ by subset construction, then $L(D) = L(N)$.

Proof: We prove by induction on $|w|$ that

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w) \tag{1}$$

Notice that each of the $\hat{\delta}$ function returns a set of states from Q_N, but $\hat{\delta}_D$ interprets this set as one of the states of Q_D, while $\hat{\delta}_N$ interprets this set as a subset of Q_N.

Basis: Let $|w| = 0, w = \epsilon$. By basis definition of $\hat{\delta}$ for DFA and NFA, both $\hat{\delta}_D(\{q_0\}, \epsilon)$ and $\hat{\delta}_N(q_0, \epsilon)$ are $\{q_0\}$

Induction: Let w be of length $n + 1$, assume statement for length n. $w = xa$, where a is final symbol of w. By inductive hypothesis, $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x) = \{p_1, ..., p_k\}$

Definition 4.1.1 tells us that

$$\hat{\delta}_{N(q_0, w)} = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2}$$

and subset construction tells us

$$\delta_D(\{p_1, ..., p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{3}$$

. We can use this to construct

$$\hat{\delta}_D(\{q_0\}, w) = \delta_D\Big(\hat{\delta}_D(\{q_0\}, x), a\Big) = \delta_D(\{p_1, ..., p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{4}$$

Thus, (2) and (4) demonstrate that $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$. When we observe that D and N both accept w if and only if $\hat{\delta}_D(\{q_0\}, w)$ or $\hat{\delta}_N(q_0, w)$, respectively, contain a state F_N, we have a completed proof that $L(D) = L(N)$ ■

Theorem 4.3.2: Language L is accepted by some DFA if and only if L is accepted by some NFA

Proof: (If) The if part is subset construction and Theorem 4.3.1

(Only if) Convert a DFA into an identical NFA. Let $D = (Q, \Sigma, \delta_D, q_0, F)$ be a DFA. Define $N = (Q, \Sigma, \delta_N, q_0, F)$ to be the equivalent NFA, where δ_N is defined by "if $\delta_D(q, a) = p$, then $\delta_N(q, a) = \{p\}$

We can induct on $|w|$, that

$$\hat{\delta}_N(q_0, w) = \left\{\hat{\delta}_D(q_0, w)\right\}$$

Basis: $(w = \epsilon)$

$$\hat{\delta}_N(q_0, \epsilon) = \{q_0\} = \left\{\hat{\delta}_D(q_0, w)\right\}$$

Inductive Step: Let w be of length $n + 1$, $w = xa$, where a is final symbol of w.

$$\begin{aligned}
\hat{\delta}_N(q_0, xa) &= \bigcup_{p \in \hat{\delta}_N(q_0, x)} \delta_N(p, a) \quad \left(\text{defn of } \hat{\delta}_N\right) \\
&= \bigcup_{p \in \hat{\delta}_d(\{q_0\}, x)} \{\delta_D(p, a)\} (\text{I.H}) \\
&= \left\{\hat{\delta}_D(q_0, xa)\right\} \qquad \left(\text{def of } \hat{\delta}_D\right)
\end{aligned}$$

■

5. Reference

- Alphabet - Finite Non empty set of symbols, denoted by Σ
  - Powers of Alphabet
    - $\Sigma^2 = \{00, 01, 10, 11\}$
    - $\Sigma^0 = \{\epsilon\}$
    - $\Sigma^{\leq} 2 = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2$
- String - Sequence of Symbols from Σ, denoted by ϵ
  - Concat - $x = 00, y = 10, x \cdot y = xy = 0010$
  - Substring - ab is a substring of $babaa$, bb is not.
  - Subsequence - bba is not subseq of $babaa$, abb is
- Length of String - Number of symbols in a string, denoted by $|w|$
- Language - Set of strings over an alphabet
  - $L = \{00, 11, 01, 110\}$
  - $L = \emptyset$
  - $L_1 \cdot L_2 = L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$
  - $L^* = \{x_1 x_2 ... x_n : x_1, x_2, ... x_n \in L, n \in \mathbb{N}\}$
  - $L^+ = \{x_1 x_2 ... x_n : x_1, x_2, ... x_n \in L, n \geq 1\}$

5.0.1. Proof Structure

5.0.1.1. Set A = Set B

1. Show $A \subseteq B$
   - Take **arbitrary** element $x \in A$
   - Use definition of A to show $x \in B$
   - Therefore, $A \subseteq B$
2. Show $B \subseteq A$
   - Take **arbitrary** element $x \in B$
   - Use definition of B to show $x \in A$
   - Therefore, $B \subseteq A$
3. Since $A \subseteq B$ and $B \subseteq A$, $A = B$