계산이론

2022년 1학기

이은주

1장 계산 이론 개요

증명기법, 언어, 문법, 오토마타

증명이란?

- 논리적인 법칙 이용
  - 주어진 가정으로부터 결론을 유도, 추론의 한 방법
- 명제나 논증이 적절하고 타당한지를 입증하는 작업
- 참(T)인 전제들이 주어짐
- **유효 추론(valid argument)**
  - 참인 전제들의 결론 - 참(T)이 되는 것
- 정확한 증명 : 유효 추론 성립

증명에 필요한 개념

- 공리(Axiom) : 항상 참(T)으로 이용되는 명제(proposition)
- 정리(Theorem) : 공리와 정의를 통해 참(T)으로 확인된 명제
- 정의(Definition) : 논의 대상의 의미와 내용을 뚜렷이 규정한 문장이나 식
- **증명(Proof)** : 특정한 공리들을 가정하고, 어떤 명제가 참이라는 것을 보여주는 것

증명에 필요한 개념 : 공리(Axiom)

- 가장 기초적인 근거가 되는 명제
- 다른 명제들을 증명하기 위한 전제로 이용되는 가장 기본적인 가정
- 예
  - **명제 P가 성립한다**면, 명제 ‘ P 또는 Q ‘도 성립한다.
  - **두 점이 주어졌을** 때, 그 두 점을 통과하는 직선을 그을 수 있다. (유클리드 기하학)
  - $a = b$이면, $a + c = b + c$이다.

증명에 필요한 개념 : 정리(Theorem)

- 수학에서 가정(assumption)으로부터 증명된 명제
  - 보조정리(lemma) : 정리를 증명하기 위해 사용, 보조적인 명제
  - 따름정리(corollary) : 정리로부터 쉽게 도출, 부가적인 명제

- 예
  - 직각삼각형에서 빗변 길이의 제곱은 다른 두 변의 길이의 제곱의 합과 같다. (피타고라스의 정리)
  - 평면을 유한개의 부분으로 나누어 각 부분에 색을 칠할 때, 서로 맞닿은 부분을 다른 색으로 칠한다면 네 가지 색으로 충분하다. (4색 정리)

증명에 필요한 개념 : 정의(Definition)

- 용어(낱말, 구, 기타 상징의 집합)의 의미를 설명하는 문장
- 사용하는 용어, 기호의 의미 규정

- 예
  - 네 변의 길이가 같은 사각형을 **정사각형**이라 한다.
  - **직각삼각형**은 한 내각의 크기가 직각인 삼각형이다.

증명(Proof)

- 증명(Proof) : 하나의 명제가 참(T)임을 확인하는 과정

- 종류
  - 직접 증명법
  - 존재 증명법
  - **수학적 귀납법**
  - **모순 증명법**

- 증명(Proof) : 하나의 명제가 참(T)임을 확인하는 과정

- 종류
  - **직접 증명법**
  - 존재 증명법
  - 수학적 귀납법
  - 모순 증명법

- 공리와 정의, 이미 증명된 정리

- 명제 $p \rightarrow q$의 직접 증명
  - ✓ 논리적으로 p의 진릿값이 참일 때 q도 참이 됨을 보임

- 증명(Proof) : 하나의 명제가 참(T)임을 확인하는 과정

- 종류
  - 직접 증명법
  - **존재 증명법**
  - 수학적 귀납법
  - 모순 증명법

- 주어진 명제가 참이 되는 예

- $\mathrm{P}(x)$: 변수 x를 가지는 명제
  - ✓ $\mathrm{P}(x)$가 참인 x 가 적어도 하나 존재함
    ($\exists x$ *such that* $\mathrm{P}(x)$) 을 보임으로 증명

- 증명(Proof) : 하나의 명제가 참(T)임을 확인하는 과정

- 종류
  - 직접 증명법
  - 존재 증명법
  - **수학적 귀납법**
  - 모순 증명법

- 기저 명제(base case) 참, 귀납 규칙(induction rule)을 증명, 무한히 많은 다른 명제들도 참이라는 것을 보임

- 수학적 귀납법의 3단계
  - ✓ **기저 단계(basis)** : 출발점이 되는 n의 값을 대입하여 초깃값 계산
  - ✓ **귀납 가정(inductive assumption)** : P_1, P_2, $P_3, \cdots, P_n$이 사실이라고 가정
  - ✓ **귀납 단계(inductive step)** : 기저 단계와 귀납 가정을 이용하여 P_{n+1}의 경우에 성립됨을 보임

- 증명(Proof) : 하나의 명제가 참(T)임을 확인하는 과정

- 종류
  - 직접 증명법
  - 존재 증명법
  - 수학적 귀납법
  - **모순 증명법**

- 어떤 명제가 거짓이라고 가정, 모순이 발생한다는 것을 증명, 그 명제가 참이어야 함
  - ✓ 기존의 전통적인 방법으로 쉽게 증명할 수 없을 경우

증명 기법 : 귀납법(proof by induction)

- 귀납법(proof by induction)
  - 특정 사례(instance)가 참(true)이라는 사실들로부터 여러 문장들이 참임을 추론

- 참임을 증명하고자 하는 문장들의 순서열 $P_1, P_2, \cdots$ 에 대하여 다음이 성립한다고 가정
  - 어떤 $k(k \geq 1)$에 대하여 $\boxed{P_1, P_2, \cdots, P_k}$는 참 — 귀납법의 기저(base)
  - $n \geq k$인 모든 n에 대해서 $\boxed{P_1, \cdots, P_k, \cdots, P_n}$이 참인 사실이 $\boxed{P_{n+1}}$가 참임을 의미

귀납 가정
(inductive assumption)

귀납 단계(inductive step)

증명 기법 : 귀납법(proof by induction)

- 어떤 명제 P(n)

(i) $n = 1$일때 참이고,

귀납법의 기저(base)

(ii) $n = k$일때 참이면 $n = k + 1$일때도 참

귀납 가정(inductive assumption)

이면 주어진 n에 관한 명제는 모든 $n \geq 1$에 대해 참이다.

귀납 단계(inductive step)

증명 방법 : 귀납법(proof by induction)

- [ex 1] 귀납법(Proof by Induction)

  $S_n = \sum i = \frac{n(n+1)}{2}$ 임을 증명하라.

  [증명] (기저 단계) n = 0 인 경우 $S_0 = 0$

  (귀납 가정) 만약 $S_n = \frac{n(n+1)}{2}$라고 가정하면

  (귀납 단계) $S_{n+1} = S_n + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{(n+1)(n+2)}{2}$

  그러므로 위의 식은 성립한다.

증명 기법 : 귀납법(proof by induction)

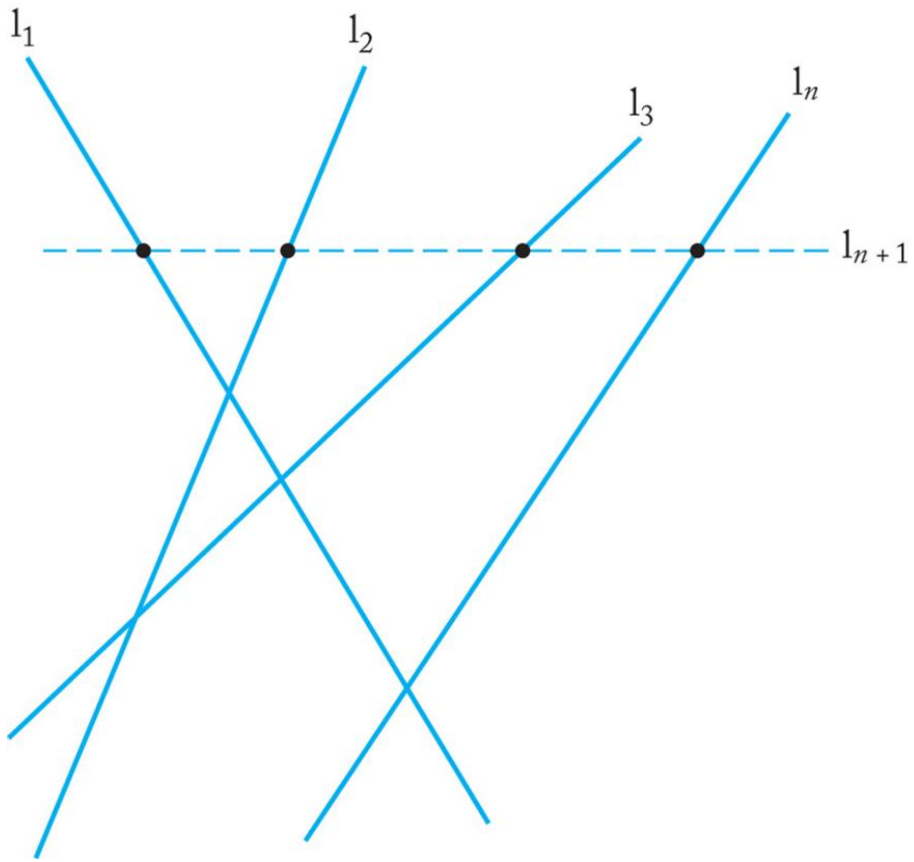
- 예제 1-5 (12 page) 높이가 n인 이진 트리의 잎 노드의 총 개수가 최대 2^n임을 증명
  - **증명** : 높이가 n인 이진 트리의 잎 노드의 최대 개수 $l(n)$, $\boldsymbol{l(n) \le 2^n}$을 증명
  - **기저 단계** : 높이가 0인 이진 트리 루트만 존재- 최대 1개의 잎 노드, $\boldsymbol{l(0) = 1}$이고 $\boldsymbol{2^0 = 1}$이므로 $\boldsymbol{l(0) = 2^0}$ 성립
  - **귀납 가정** : $\boldsymbol{l(i) \le 2^i, for\ i = 0, 1, \dots, n}$
  - **귀납 단계** : 높이가 n 인 이진 트리로 부터 높이가 $n+1$인 이진 트리를 만들기 - 높이가 n인 트리의 잎 노드들 각각에 대해 최대 두개의 잎 노드를 추가 생성, $\boldsymbol{l(n+1) \le 2^{n+1}}$
  - 귀납 가정을 적용 : $\boldsymbol{l(n+1) \le 2 \times 2^n = 2^{n+1}}$
    - $\boldsymbol{n}$은 임의의 수일 수 있으므로 모든 $\boldsymbol{n}$에 대해 참

증명 기법 : 귀납법(proof by induction)

- 귀납적 추론(inductive reasoning) : 귀납법과 프로그래밍에서 사용되는 재귀(recursion) 개념간의 연관성
- 예 : n을 양의 정수라 할 때 임의의 함수 $f(n)$에 대한 재귀적 정의는 다음의 두 부분으로 나뉨
  - $f(n+1)$의 정의를 $f(n), f(n-1), \ldots, f(1)$로 나타냄
    - 귀납법의 귀납 단계에 해당
  - 재귀로부터 벗어나는 부분 : $f(1), f(2), \ldots, f(k)$를 비재귀적으로 정의
    - 귀납법의 기저 단계에 해당
- 재귀 : 몇몇 초기값들과 문제 자체의 재귀적 속성만 주어지면 문제의 모든 사례(instance)들에 대한 결론을 유도

증명 기법 : 귀납법(proof by induction)

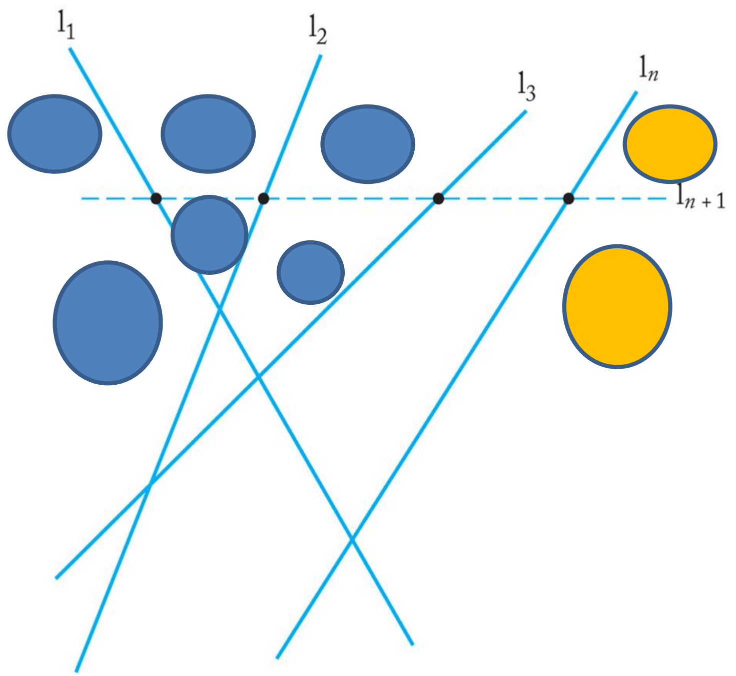
- 예제 1-6 (13 page) $l_1, l_2, \ldots, l_n$은 평면을 여러 개의 분리된 영역으로 나누는 서로 교차하는 직선들
  - 직선 하나 : 두 부분
  - 두 개의 직선 : 4개의 영역
  - 세 개의 직선 : 7개의 영역
  - 직선의 개수 증가 : 대입해서 구하기 어려움



재귀적으로 해결

증명 기법 : 귀납법(proof by induction)

- 예제 1-6 (13 page) 새로운 직선 l_{n+1}을 추가하였을 경우 평면의 개수는?
  - 직선 l_1, l_2, l_3의 왼쪽 영역 :
    - 두 개의 새로운 영역으로 분할
  - 마지막 직선 l_n의 왼쪽 영역 :
    - 두 개의 새로운 영역으로 분할
  - n개 교차점 각각에 대하여
    - 새로운 영역+ 맨 뒤에 새로운 영역 하나 더 추가



증명 기법 : 귀납법(proof by induction)

- 예제 1-6 (13 page) 재귀식으로 풀기
  - $A(n)$: n 개의 직선으로 분할된 영역의 수

$$A(n+1) = A(n) + n + 1, \qquad n = 1,2,\ldots,$$
$$A(1) = 2$$

  - 위의 재귀식으로 계산 : $A(2) = 4,\ A(3) = 7, A(4) = 11$
  - 귀납법 사용 : $A(n)$에 대한 식을 얻고 그 식이 참임을 증명
    - 예측 : $A(n) = \frac{n(n+1)}{2} + 1$
    - 귀납단계

$$A(n+1) = \frac{n(n+1)}{2} + 1 + n + 1 = \frac{(n+1)(n+2)}{2} + 1$$

증명 기법 : 귀류법(모순 증명법, proof by contradiction)

- 귀류법(모순 증명법, proof by contradiction)
  - 주어진 문장 : 거짓이라 가정, 틀린 결론 유도 : 주어진 문장은 참
  - 다른 모든 증명 방법 실패할 때 사용

증명 기법 : 귀류법(모순 증명법, proof by contradiction)

- 예제 1-7 (15 page) **모순 증명법(Proof by Contradiction)**

  √2가 유리수(rational number)가 아님을 증명하라.

  [증명]

  √2 : 유리수라고 가정

  √2 = n/m (<u>n, m은 서로 소(disjoint)인 정수</u>)

  양변을 제곱하여 정리 : $2m^2 = n^2$

  $2m^2$: 짝수, n^2 : 짝수 → n = 2k

  식에 대입하면 $2m^2 = 4k^2$, $m^2 = 2k^2$

  $2k^2$: 짝수 → m^2 도 반드시 짝수 (<u> m도 짝수)</u>

  m과 n이 모두 짝수 : <u>서로 소라는 가정에 모순</u>

  √2는 유리수가 아니다.

세 가지 기초 개념

- 언어(Language) : 특별한 성질을 가진 스트링들의 집합
- 문법(Grammar) : 언어를 정의하는 수단
- 오토마타(Automata) : 언어를 인식하는 추상의 기계(모델)

언어(Language)

- **기호**(symbol, character)
- **알파벳**(alphabet) Σ : 기호들의 집합
- **문자열**(string) : 기호의 나열
  - 널(null) 스트링 λ
  - 스트링 w의 길이 : $|w|$

언어(Language)

- [ex 2] 알파벳 $\Sigma = \{a, b\}$

  $\lambda, a, b, abaa, ab$: Σ로부터 만든 스트링

  $w = abaa$: 값 $abaa$를 갖는 스트링 (명칭) w

  - <u>묵시적인 표기법</u>

  영문 소문자 앞부분 $a, b, c, \cdots$: 알파벳 기호를 표시

  영문 소문자 뒷부분 $u, v, w, \cdots$: 스트링 명칭을 표시

언어(Language)

- 스트링 u와 v의 연결(concatenation) : uv
- 역 스트링(reverse string) : w^R of string w

- [ex 3] $\Sigma = \{a, b, c\}$

  스트링 $u = abc, v = bba, w = cc$

  - uv 의 연결 : $abcbba$

  - wuw의 연결 : $(wu)w = \ (ccabc)cc = ccabccc$

  - 역 스트링 $v^R = abb$

언어(Language)

- u, v, w : 스트링
  - $u(vw) = (uv)w$
  - $\lambda w = w\lambda = w$
  - $uv \neq vu$
  - $|uv| = |u| + |v|$

  - 접두사(prefix) **u** of $w = uv$
  - 접미사(postfix) **v** of $w = uv$
  - 부분 스트링(substring) **u** of $w = xuy$

- $\mathbf{u}^{\mathbf{n}}$ = uu ⋯ u
- $\mathbf{u}^{\mathbf{o}}$ = λ

언어(Language)

- L_1, L_2 : 스트링의 집합
  집합 L_1과 L_2의 **연결**(concatenation) : $L_1 L_2$
  $L_1 L_2 = \{ xy \mid x \in L_1 \text{ and } y \in L_2 \}$

- $L^n = LL \cdots L$
  $L^0 = \{\lambda\}$

- $L_1 \cup L_2 = L_2 \cup L_1$
  $L_1 L_2 \neq L_2 L_1$
  $LL^0 = L^0 L = L$

언어(Language)

- [ex 4] $\Sigma = \{a, b, c\}$, $L_1 = \{a, ab\}, L_2 = \{c, bc\}$
  - $L_1 \cup L_2 = L_1 \cup L_2 = \{a, ab, c, bc\}$
  - $L_1 L_2 = \{ac, abc, abbc\}$
  - $L_2 L_1 = \{ca, cab, bca, bcab\}$
    - $\boldsymbol{{L_1}^2 = L_1 L_1 = \{a, ab\}\{a, ab\} = \{aa, aab, aba, abab\}}$
    - $\boldsymbol{{L_1}^3 = L_1 L_1 L_1 = \{a, ab\}\{a, ab\}\{a, ab\} = \{aaa, \ldots\}}$

- [ex 5] $L_a = \{ a^n, n \geq 0 \}$, $L_b = \{ b^n, n \geq 0 \}$
  - $L_a L_b = \{ a^n, n \geq 0 \} \{ b^n, n \geq 0 \}$

    $= \{ \lambda, a, aa, aaa, \cdots \} \{ \lambda, b, bb, bbb, \cdots \}$

    $= \{ a^n b^m, n \geq 0, m \geq 0 \}$

    $= \{ \cancel{a^n b^n, n \geq 0} \}$ (X)

언어(Language)

- Σ^* : closure of Σ (시그마의 클로져)
  - Σ 의 기호를 0번 이상 연결하여 만든 스트링의 집합
  - $\Sigma^* = \Sigma^i = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$

- Σ^+ **: positive closure of** Σ (시그마의 포지티브 클로져)
  - Σ 의 기호를 1번 이상 연결하여 만든 스트링의 집합
  - $\Sigma^+ = \Sigma^i = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$
  - 그러므로 $\Sigma^+ = \Sigma^* - \{\lambda\}$

- 언어(language) : Σ^*의 부분집합
- 문장(sentence) : 언어에 속하는 스트링

언어(Language)

- [ex 6] $\Sigma = \{a, b\}$

  $\sum^* = \{ \lambda, a, b, aa, ab, ba, bb, aaa, \cdots \} = \{ a, b \}^*$

  $\sum^+ = \{ a, b, aa, ab, \cdots \} = \{ a, b \}^+$

  언어 $L_1 = \{ a, aa, aab \}$: 3개의 문장

  언어 $L_2 = \{ w \mid |w| = 2, w \in \{a,b\}^* \} = \{ aa, ab, ba, bb \}$

  언어 $L_3 = \{ a^n b^n, n \geq 0 \}$: 무한개의 문장

  언어 $L_4 = \{ abw \mid w \in \{a,b\}^* \}$

  언어 $L_5 = L_3 L_3 = L_3^2 = \{ a^n b^n, n \geq 0 \} \{ a^n b^n, n \geq 0 \}$

  $= \{ a^n b^n a^m b^m, n \geq 0, m \geq 0 \}$

  $\cancel{= \{ a^n b^n a^n b^n, n \geq 0 \}}$

  $\therefore L_1, L_2, L_3, L_4, L_5 \subseteq \sum^*$

세 가지 기초 개념

- 언어(Language) : 특별한 성질을 가진 스트링들의 집합
- **문법(Grammar) : 언어를 정의하는 수단**
- 오토마타(Automata) : 언어를 인식하는 추상의 기계(모델)

문법(Grammar)

- [정의] 문법 $G = (N, \Sigma, P, S), G = (N, \Sigma, S, P)$
  - N : nonterminal 기호의 집합(variable)
  - Σ : terminal 기호의 집합
  - P : $\boldsymbol{\alpha \rightarrow \beta}$ 형태의 **생성규칙**(production)의 집합

    (단, $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ and $\beta \in (N \cup \Sigma)^*$)
  - S : **시작 기호**(start symbol) $\in$ N

- $\alpha \rightarrow \beta_1$ ⇨ $\boldsymbol{\alpha \rightarrow \beta_1 \mid \beta_2}$

  $\alpha \rightarrow \beta_2$

문법(Grammar)

- [ex 7] 문법의 여러 가지 표현방법
  - $G_1 = (\{A, S\}, \{0, 1\}, P, S)$

    $P : S \rightarrow 0A1$

    $\quad 0A \rightarrow 00A1$

    $\quad A \rightarrow \lambda$

  - $G_2 = (\{<digit>\}, \{0,1,\cdots,9\}, \{<digit> \rightarrow 0|1|\cdots|9\}, <digit>)$

  - $G_3 = (\{S\}, \{a, b\}, \{ S \rightarrow aSb \mid ab \}, S)$

    또는

    $G_3 = (\{S\}, \{a, b\}, P, S)$

    $P : S \rightarrow aSb \mid ab$

문법(Grammar)

- 묵시적 선언(implicit declaration)
  - 영문자 대문자 ⇨ 논터미날

- 최초의 nonterminal ⇨ 시작 기호
  - 소문자 및 기타 특수기호 ⇨ 터미널

$$
\begin{aligned}
G_1 : S &\rightarrow 0A1 \\
0A &\rightarrow 00A1 \\
A &\rightarrow \lambda
\end{aligned}
$$

문법(Grammar)

- 생성규칙 $\alpha \rightarrow \beta$ 의 의미 :
  - **α를 β 로 바꿀 수 있다.**

- $x\alpha y \Rightarrow x\beta y$ 의 의미 :
  - 생성규칙 $\alpha \rightarrow \beta$ 를 이용하여 **$x\,\alpha\,y$ 로부터 $x\,\beta\,y$ 를 유도**(derive)

$\Rightarrow$: 직접(1번에) 유도

$\overset{*}{\Rightarrow}$: 여러 번(0번 이상)에 걸쳐 유도

문법(Grammar)

- **문장형**(sentential form) of $G = (N, \Sigma, P, S)$
  - 시작기호 S로부터 시작하여 유도과정에 나타나는 모든 스트링

- **문장**(sentence) : 터미널 기호만으로 구성된 스트링

- **언어**(language) L(G)
  - 문법 G로부터 만들어진 문장(터미날 스트링)들의 집합
  - **L(G) = { w | w $\in \Sigma^*$ and S $\stackrel{*}{\Rightarrow}$ w }**

문법(Grammar)

- [ex 8] $G_1 = (\{A, S\}, \{0, 1\}, P, S)$

  $P : S \rightarrow 0A1$

  $0A \rightarrow 00A1$

  $A \rightarrow \lambda$

  $S \Longrightarrow 0A1 \Longrightarrow 01\,S \overset{*}{\Rightarrow} \mathbf{01}$

  $S \Longrightarrow 0A1 \Longrightarrow 00A11 \Longrightarrow 0011\,S \overset{*}{\Rightarrow} \mathbf{0011}$

  $S \Longrightarrow 0A1 \Longrightarrow 00A11 \Longrightarrow 000A111 \Longrightarrow 000111\,S \overset{*}{\Rightarrow} \mathbf{000111}$

  $\vdots$

  $\therefore L(G_1) = \{ \mathbf{0^n1^n} \mid n \geq 1 \}$

  - 문장형: S, 0A1, 00A11, 000A111, 01, 0011, 000111, ⋯
  - 문장: 01, 0011, 000111, ⋯
  - 언어 $L(G_1) = \{ \mathbf{0^n1^n} \mid n \geq 1 \}$

문법(Grammar)

- G_{11} : $S \rightarrow 0A1$

  $A \rightarrow 0A1$

  $A \rightarrow \lambda$ $\therefore L(G_{11}) = \{ \mathbf{0^n1^n} \mid n \geq 1 \}$

- G_{12} : $S \rightarrow 1A0$

  $0A \rightarrow 00A1$

  $A \rightarrow \lambda$

  $S \Longrightarrow 1A0 \Longrightarrow 10$ $\therefore L(G_{12}) = \{ \mathbf{10} \}$

- G_{13} : $S \rightarrow 1A0$

  $A \rightarrow 0A1 \mid \lambda$

  $S \Longrightarrow 1A0 \Longrightarrow 10 \quad S \overset{*}{\Rightarrow} 10$

  $S \Longrightarrow 1A0 \Longrightarrow 10A10 \Longrightarrow 1010 \quad S \overset{*}{\Rightarrow} 1010$

  $S \Longrightarrow 1A0 \Longrightarrow 10A10 \Longrightarrow 100A110 \Longrightarrow 100110 \quad S \overset{*}{\Rightarrow} 100110$

  ⋮

$\therefore L(G_{13}) = \{ \mathbf{10^n1^n0} \mid n \geq 0 \}$

문법(Grammar)

- [ex 9] $G_2 : S \rightarrow Ab$

  $A \rightarrow aAb \mid \lambda$

  $L(G_2) = \{ \mathbf{a^n b^{n+1}} \mid n \geq 0 \}$

- [ex 10] $G_3 = (\{S\}, \{a, b\}, \{ S \rightarrow aSb \mid ab \}, S)$

  $S \Rightarrow \mathbf{ab}$

  $S \Rightarrow aSb \Rightarrow \mathbf{aabb}$

  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \mathbf{aaabbb}$

  $\vdots$

  $\therefore \mathbf{L(G_3) = \{ a^n b^n \mid n \geq 1 \}}$

문법(Grammar)

- [ex 11] $G_4 = (\{E, T, F\}, \{a, +, *, (,)\}, P, E)$

  $P : E \rightarrow E + T \mid T$

  $T \rightarrow T * F \mid F$

  $F \rightarrow (E) \mid a$

  $E \overset{*}{\Rightarrow} a + a * a$

  ∴ $L(G_4)$ = {피연산자 a와 연산자 +, * 그리고 괄호 ( ) 로 이루어진 산술식}

문법(Grammar)

- [ex 12] $G_5 : S \rightarrow aSBC \mid abC$

  $CB \rightarrow BC$

  $bB \rightarrow bb$

  $bC \rightarrow bc$

  $cC \rightarrow cc$

  $S \Rightarrow aSBC \Rightarrow aabCBC \Rightarrow aabBCC \Rightarrow aabbCC \Rightarrow aabbcC \Rightarrow \mathbf{aabbcc}$

  $\therefore L(G_5) = \{ \mathbf{a^n b^n c^n} \mid n \geq 1 \}$

문법(Grammar)

- [ex 13] G_6 : $A \rightarrow aABC \mid abC$

  $CB \rightarrow BC$

  $bB \rightarrow bb$

  $bC \rightarrow b$

  $L(G_6) = \{ a^n b^n \mid n \geq 1 \}$

- [ex 1.12] G_7 : $S \rightarrow 0S \mid 1S \mid \lambda$

  $L(G_7) = \{ w \mid \mathbf{w} \in \{\mathbf{0}, \mathbf{1}\}^* \}$

문법(Grammar)

- [ex 14] $G_8 : S \rightarrow aAb$

  $aA \rightarrow aaAb$

  $A \rightarrow \lambda$

  $L(G_8) = \{ a^n b^n \mid n \geq 1 \}$

  ※ $G_3 : S \rightarrow aSb \mid ab$

  $L(G_3) = \{ a^n b^n \mid n \geq 1 \}$

- $L(G_3) = L(G_6) = L(G_8)$

  $\therefore G_3, G_6, G_8$: 동치 문법(equivalent grammar)

문법(Grammar)

- 예제 1.11 (교재 26page): $G = (V, T, S, P)$

$V = \{ S \}$

$T = \{ a, b \}$

$P = \{ S \rightarrow aSb, S \rightarrow \lambda \}$

$S \Rightarrow aSb$ (applying first production)

$\Rightarrow aaSbb$ (applying first production)

$\Rightarrow aabb$ (applying second production)

문법(Grammar)

- 예제 1.14 (교재 30page): $G = (V, T, S, P)$

  V = { A, S }, T = { a, b }, and productions

  $S \rightarrow aAb \mid \lambda$

  $A \rightarrow aAb \mid \lambda$

- The grammars in examples 1.11 and 1.14 can be shown to be equivalent

Chomsky Hierarchy(촘스키의 분류)

- [정의] 문법 $G = (V, T, S, P)$ 의 분류
  - $(\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ and $\beta \in (N \cup \Sigma)^*$)

- **Type 3** : **정규(regular) 문법**
  a) **우선형**(right-linear) 문법 : 생성규칙: **A → x B** 또는 **A → x** (A,B∈N and x∈Σ*)
  b) **좌선형**(left-linear) 문법 : 생성규칙: A → B x 또는 A → x

- **Type 2** : **문맥자유(context-free) 문법**
  생성규칙: **A →** ***β*** (A∈N)

- **Type 1** : **문맥연관(context-sensitive) 문법**
  생성규칙: ***α*** **→** ***β***
  (단, |α |≤|β|, non-contracting(줄어들지 않는) 문법, S → λ 은 허용)

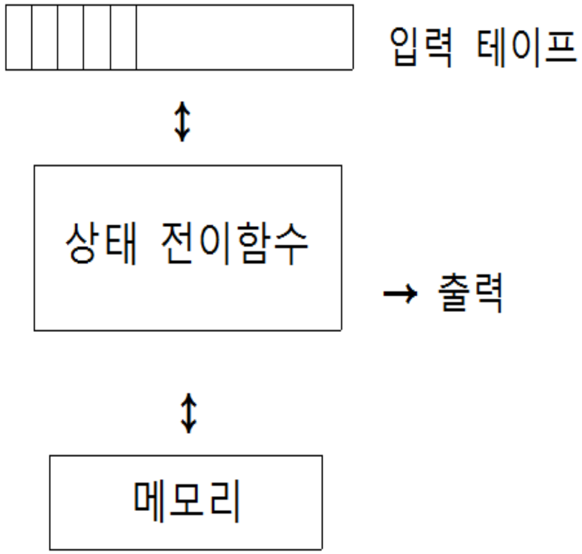
- **Type 0** : **무제한(unrestricted) 문법**
  생성규칙: ***α*** **→** ***β***

Chomsky Hierarchy(촘스키의 분류)

Type	문법(Grammar)	오토마타(Automata)
0	무제한 문법	튜링 기계
1	문맥연관 문법(CSG)	선형한계 오토마타
2	**문맥무관 문법(CFG)**	**푸시다운 오토마타**
3	**정규 문법** (우선형문법(RLG), 좌선형문법(LLG))	**유한 오토마타**

- Li : type-i 문법이 만들어 내는 언어
  L0 ⊇ L1 ⊇ L2 ⊇ L3

오토마타(Automata)



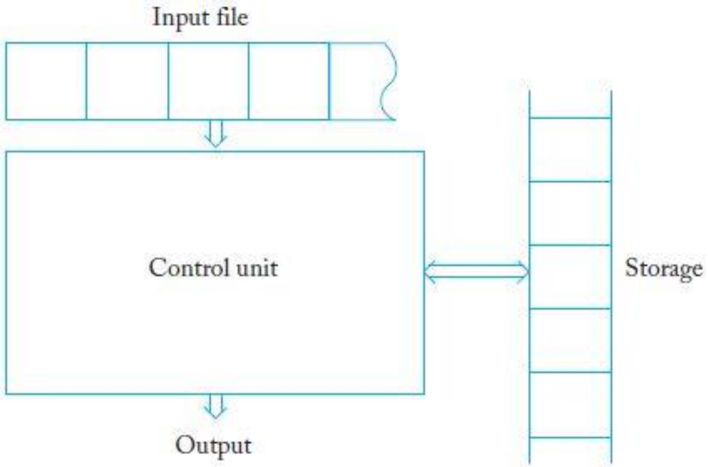
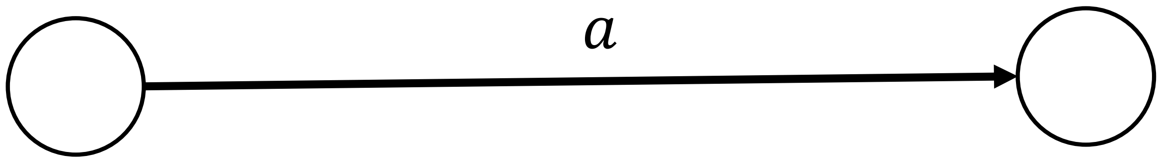


FIGURE 1.4

- **입력 테이프** : 입력 스트링
  입력시 기호를 하나씩 읽어 들임
- **메모리(stack)** : 특정한 기호를 기억(푸시다운 오토마타)
- **상태 전이함수**
- **출력** : accept/reject 또는 string

오토마타 예

- 상태 1에서 상태 2로의 전이를 나타내는 오토마타
  - 전이는 입력 심벌이 a일 때 행해 짐



오토마타

- 이산시간(discrete time) 단위로 운영
- 형상(configuration) : 제어 장치의 입력 파일, 임시기억장소의 상태 종합
- 이동(move) : 오토마타가 한 영상에서 다음 형상으로 전이하는 것

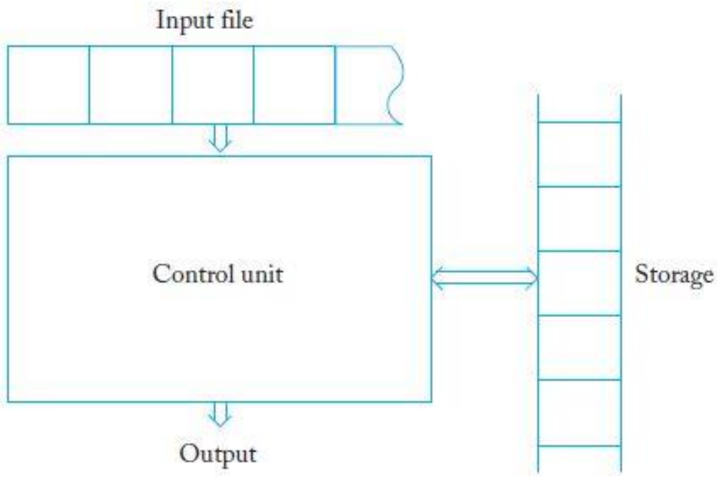
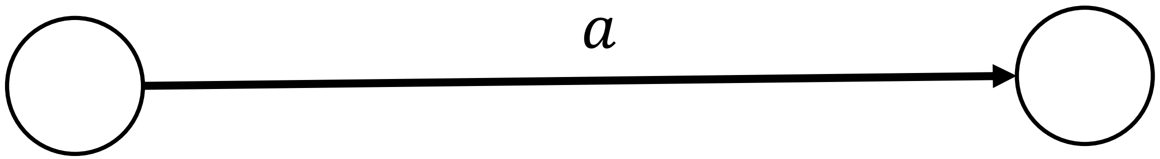


FIGURE 1.4



오토마타(Automata)

- 비결정적(non-deterministic) : 0개 이상의 상태로 전이.
- 결정적(deterministic) : 많아야 1개의 상태로 전이

- **인식기(Acceptor)**
  - 유한 오토마타, 푸쉬다운 오토마타
  - 출력: accept/reject
- **변환기(Transducer)**
  - 밀리기계, 무어기계
  - 출력: string

응용

• 형식언어와 문법 : 프로그래밍 언어 분야

예제 1.15(교재 35page)

1. 식별자 : 영문자(letters), 숫자문자(digits)와 밑줄(underscores)들의 순서열
2. 식별자 : 영문자나 밑줄로 시작
3. 식별자 : 대문자와 소문자 모두 허용

Productions for a sample grammar:

<id> → <letter><rest>|<undrscr><rest>

<rest> → <letter><rest>|<digit><rest>|<undrscr><rest>|λ

<letter> → a|b| … |z|A|B| … |Z

<digit> → 0|1|…|9

<undrscr> → -

응용

- **ao를 유도하는 과정**

<id> ⇒ <letter><rest>

⇒ a <rest>

⇒ a <digit><rest>

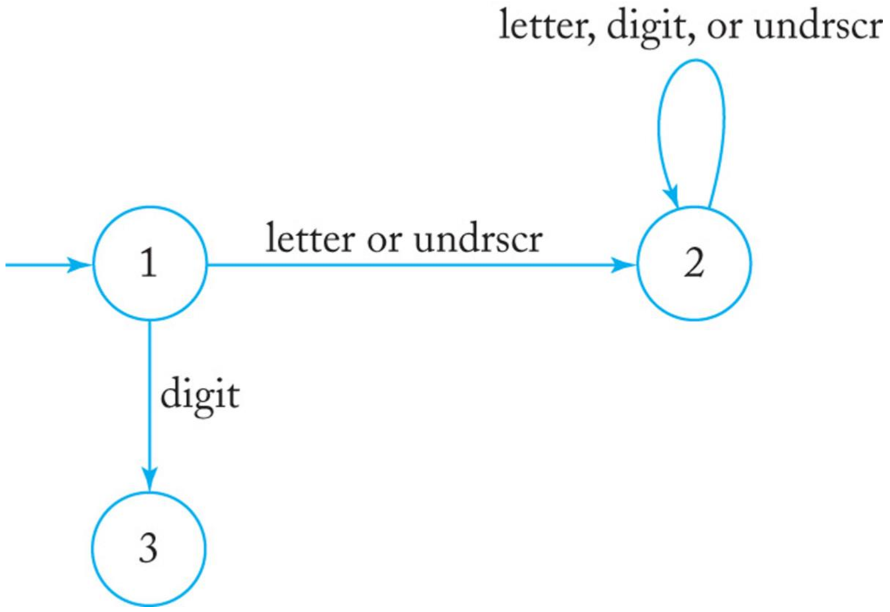
⇒ a0 <rest>

⇒ ao.

<id> → <letter><rest>|<undrscr><rest>
<rest> →
<letter><rest>|<digit><rest>|<undrscr><rest>|λ
<letter> → a|b| … |z|A|B| … |Z
<digit> → 0|1|…|9
<undrscr> → -

오토마타 예

- 예제 1.16(교재 36page)
  - 모든 적법한 C 식별자들을 승인하는 오토마타
    - 초기 : 오토마타는 상태 1에 놓인다고 가정
    - 영문자, 숫자문자와 밑줄 외의 어떤 다른 심벌도 입력되지 않음을 가정



오토마타 예

- 예제 1.17(교재 37page) 변환기 예
  - 이진 가산기(binary adder) : 범용 컴퓨터 시스템의 주요한 부분들 중의 하나
    - 가산기 : 수를 표현하는 두 개의 비트열(bit string)을 받아서 그들의 합을 출력으로 생성
    - 문제를 간단히 하기 위해서 : 양의 정수만을 대상으로 한다고 가정

$$x = a_0\ a_1, \cdots, a_n$$

정수값

$$a(x) = \sum_{i=0}^{n} a_i 2^i$$

을 나타낸다고 가정 : 일반적인 이진수 표현의 역순

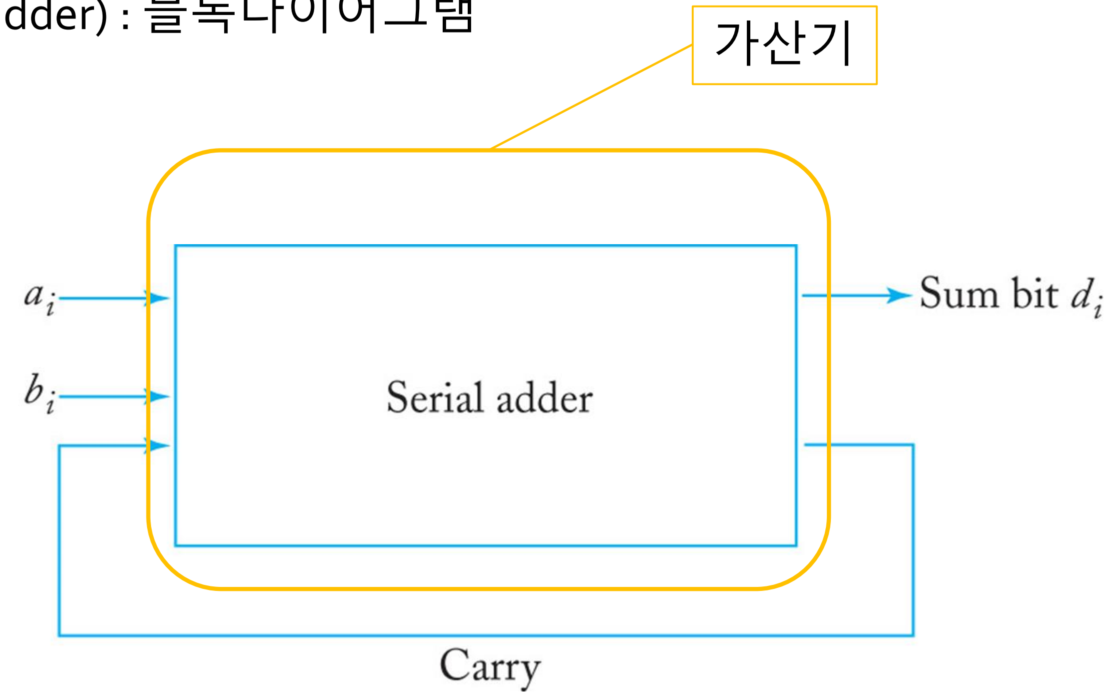
오토마타 예

- 순차 가산기(serial adder)
  - 두 개의 수 $x = a_0 a_1 \dots a_n$와 y $= b_0 b_1 \dots b_n$을 왼쪽에서부터 비트 단위로 처리
  - 각 비트의 합산은 합에 대한 비트와 다음 위치로 전해질 캐리 비트를 생성

		b_i	
		0	1
a_i	0	0 No carry	1 No carry
	1	1 No carry	0 Carry

오토마타 예

- 순차 가산기(serial adder) : 블록다이어그램

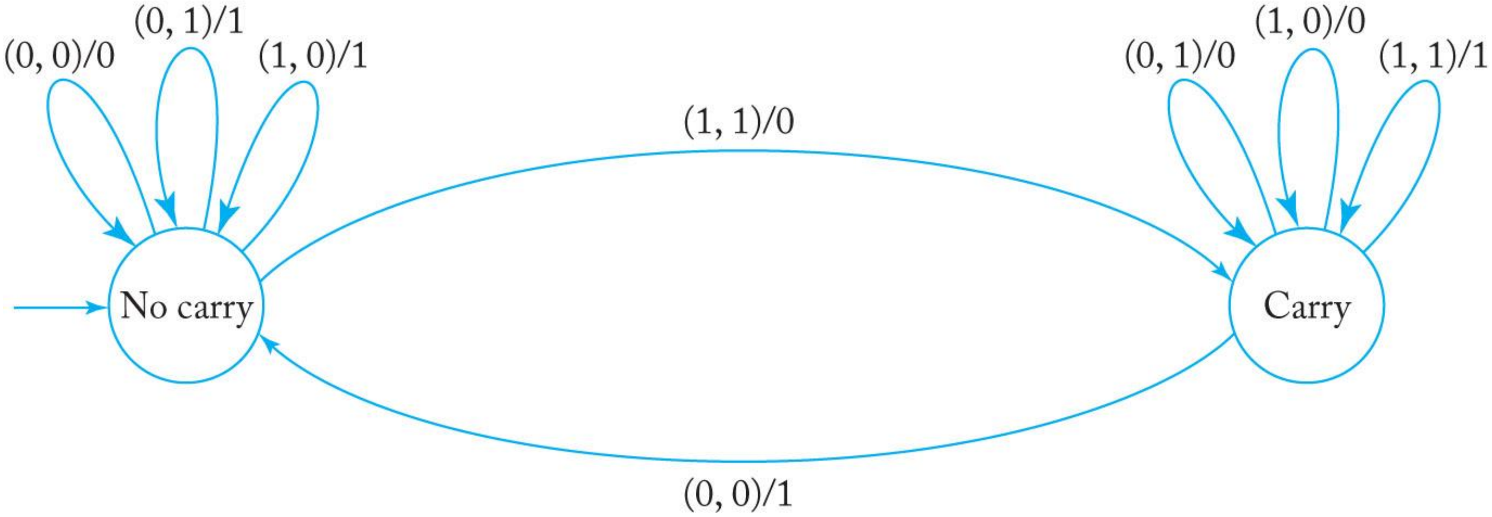


- 입력 : 두 비트, 출력 : 합산 비트, 캐리비트
- 오토마타(여기서는 변환기) : 내부 처리 과정을 명확하게 함

오토마타 예

- 가산기의 오토마타
  - 변환기로의 입력 : 비트 쌍(a_i, b_i)
  - 출력 : 합산 비트 d_i
  - 간선 : $(a_i, b_i)d_i$ 형태의 라벨
  - 캐리가 존재하면 다음 입력 비트 쌍이 읽혀질 때 합산에 고려됨

a_i \ b_i	0	1
0	0 No carry	1 No carry
1	1 No carry	0 Carry



유한상태 기계

- 오토마타
  - 입력 데이터를 읽을 수 있는 입력 기능을 가지고 있음
  - 입력 데이터 : 입력 파일에 쓰여져 있는 알파벳상의 문자열(string)들
  - 입력 파일 : 네모꼴의 셀(cell)들로 이루어져 있으며 각 셀에는 오직 하나의 기호(symbol)만 존재
  - 특정 형태의 출력 기능을 가지고 있음
    - 0(no)이나 1(yes)의 출력을 생성

유한상태 기계

- 인식기(accepter)로서의 오토마타
  - 출력이 단순히 0(no)이나 1(yes)로 제한되어 있는 오토마타
  - 입력 문자열이 주어졌을 때 단지 그 문자열을 인식(accept)하거나 거부(reject)하는 역할만 수행

유한상태 기계

- 변환기(transducer)로서의 오토마타
  - 임의의 문자열을 출력으로 생성할 수 있는 오토마타
  - 입력이 출력에 관여를 하는 가의 여부에 따라서 분류
    - 무어 기계(Moore machine) : 출력이 현재의 상태에서만 결정됨
    - 밀리 기계(Mealy machine) : 출력이 현재의 상태와 입력 모두에 의해서 결정됨
  - 일반적으로 대부분의 시스템에서는 무어 기계와 같은 것을 선호

유한상태 기계

- 유한상태 기계(유한오토마타)
  - 유한한 상태들의 집합과 전이 함수(transition function)들의 집합으로 구성
  - 임시 저장장치가 없음
  - 여기서의 ‘전이’ : 알파벳 Q로부터 선택된 입력 기호(input symbol)에 의해 생기는, 상태에서 상태로의 변화
  - 입력 기호에 따라 상태는 항상 변할 수 있으며, 원래의 상태로 다시 돌아가는 전이도 있을 수 있음

유한상태 기계

- 유한상태 기계(유한 오토마타)의 구조