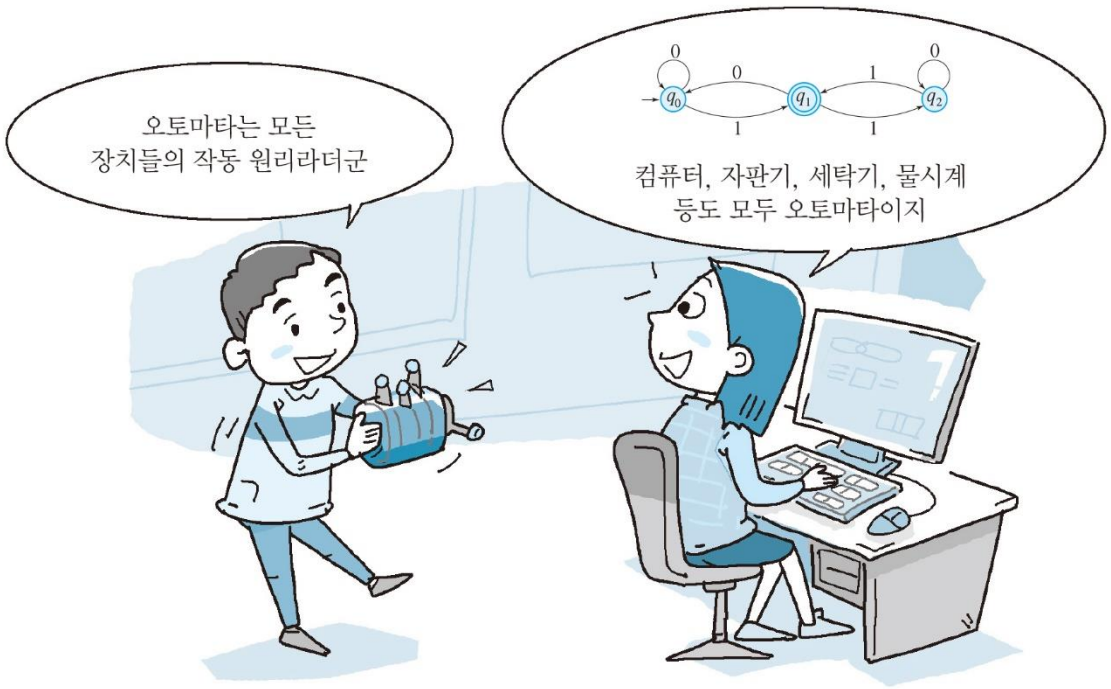
13 CHAPTER

오토마타, 형식 언어, 문법

Automata, Formal Language & Grammar



단원의 주요 목표

오토마타, 형식언어, 문법과 관련된 다양한 논제들을 고찰한다.

- 오토마타의 정의와 오토마타의 특성과 구조를 알아본다.
- 오토마타 학습의 필요성과 유한 상태 시스템의 응용을 탐구한다.
- 유한 오토마타의 개념, 작동, 응용을 이해한다.
- 형식언어와 문법 사이의 관련성을 파악한다.
- 튜링머신 모델을 정의하고 촘스키 포함 관계를 학습한다.
- 오토마타의 응용 분야와 4차 산업혁명과의 관계를 살펴본다.

13 CHAPTER

오토마타, 형식 언어, 문법

Automata, Formal Language & Grammar

CONTENTS

오토마타, 형식 언어, 문법

- **오토마타, 형식 언어, 문법**에 관한 연구는 매우 추상적인 특성을 가지고 있음
- 디지털 모델링, 컴파일러, 문서 편집기, 엘리베이터 등 다양한 분야에 응용이 됨
- 이론적인 계산 모델인 오토마타 중에서 유한 오토마타는 컴파일러의 어휘분석을 수행하는 데 있어서 결정적인 역할을 함
- 오토마타, 형식 언어, 문법은 상호 밀접한 관계에 있는데, 각종 컴퓨터 프로그램 언어들이 정해진 문법에 따른 형식 언어에 기반을 두고 만들어졌기 때문임
- 한편 **튜링머신**은 현재의 디지털 컴퓨터의 역량과 대등한 계산 모델임

13.1 오토마타란 무엇인가?

오토마타(Automata)

- 디지털 컴퓨터의 수학적 모델인 **오토마톤(automaton)**의 복수형으로서 **'자동기계 장치'**란 뜻을 가짐
- 일반적으로 **입력 장치, 출력 장치, 저장 장치, 제어 장치를** 가지고 있으므로 현대적인 디지털 컴퓨터가 작동하는 이론적인 메커니즘임
- 단순한 형태의 오토마타는 기원전 3천 년 무렵부터 만들어졌는데, 고대 이집트인들이 사용했던 모래시계나 물시계 등도 넓은 의미의 오토마타임
- 요즈음 벽에 걸려 있는 뻐꾸기 시계도 오토마타에 속함
- 일반적인 뻐꾸기 시계는 정해진 시각이 되면 뻐꾸기가 안에서 튀어나와 울지만, 요즘의 뻐꾸기 시계는 빛을 감지할 수 있는 센서를 통하여 밤이 되면 울지 않는 기능도 가지고 있는 여러 가지 형태의 시계 오토마타를 나타냄

13.1 오토마타란 무엇인가?

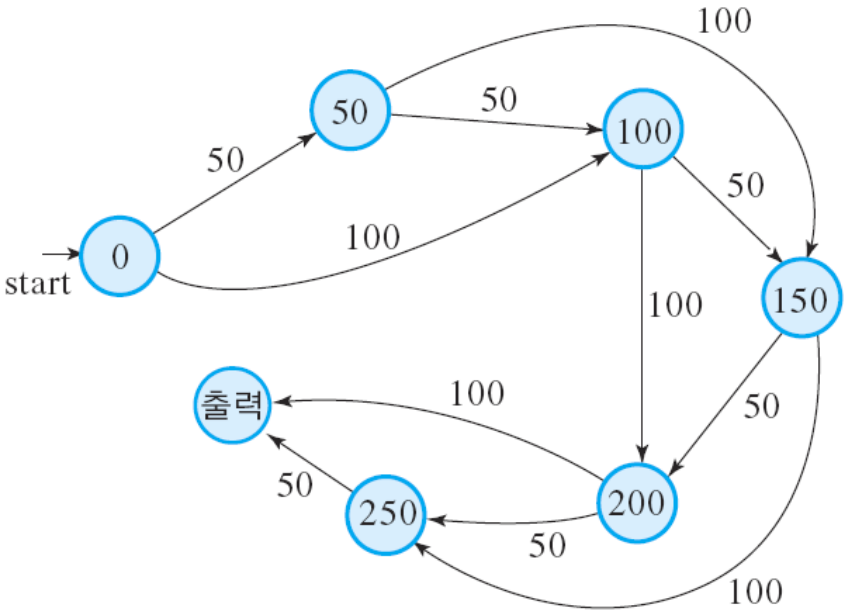
기본적이고 간단한 오토마타



〈그림 13.1〉 물시계, 모래시계, 뻐꾸기 시계

13.1 오토마타란 무엇인가?

간단한 자판기 오토마타의 다이어그램



〈그림 13.2〉 간단한 자판기 오토마타

13.1 오토마타란 무엇인가?

일상생활에서 흔히 만날 수 있는 오토마타의 예

- 이론적인 **자판기 오토마타**를 보여주는데, 50원짜리와 100원짜리 동전을 넣을 수 있음
- 투입한 돈이 300원 또는 그 이상일 때 커피나 음료수를 내주고 거스름돈을 돌려주지 않는다고 가정함
- 이러한 자판기 오토마타에서는 투입되는 액수에 따라 **상태**가 변함
- 먼저 **시작 상태**에서는 액수가 0원이며, 50원이 투입되면 50원 상태로 가고, 100원이 투입되면 100원 상태임
- 100원 상태에서 50원이 투입되면 150원상태로 이동하고, 이와 같은 과정을 계속하여 300원이나 그 이상의 액수 상태가 되면 음료수를 출력함

13.1 오토마타란 무엇인가?

오토마타의 특성

1) 오토마타는 입력 데이터를 읽을 수 있는 **입력 기능**을 가지고 있다. 입력 데이터는 입력 파일에 쓰여져 있는 알파벳상의 스트링들로 이루어져 있는데, 입력 파일은 네모꼴의 셀(cell)들로 이루어져 있으며 각 셀에는 오직 하나의 심볼만 존재함

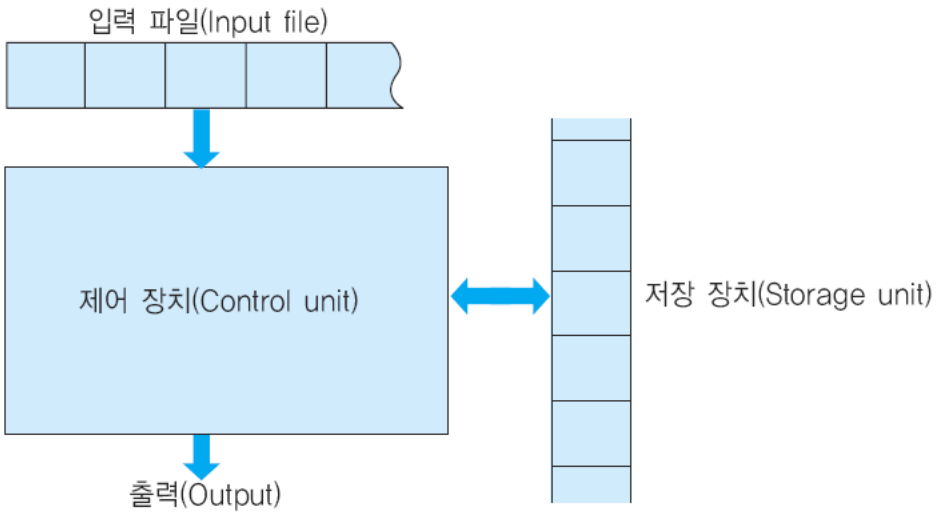
2) 오토마타는 특정 형태의 **출력 기능**을 가지고 있다. 0이나 1의 출력을 생성 할 수 있으며 인식(accept) 또는 기각(reject)의 출력도 생성함

3) 오토마타는 무한개의 셀들로 이루어진 임시 **저장 장치**를 가질 수 있다. 각 셀은 하나의 심볼만을 가질 수 있는데, 오토마타는 작동에 따라 셀들의 내용을 읽어내거나 변경할 수 있음

4) 오토마타는 유한개의 내부 상태를 제어할 수 있는 **제어 장치**를 가지고 있다. 이것의 제어에 따라 상태가 변화될 수 있음

13.1 오토마타란 무엇인가?

오토마타의 일반적인 구조



〈그림 13.3〉 오토마타의 일반적인 구조

13.1 오토마타란 무엇인가?

오토마타의 기본 개념

- 오토마타는 **이산적인(discrete)** 시간 단위로 작동됨을 기본 가정으로 함
- 어느 특정 시각에 제어 장치는 특정의 상태에 놓여 있음
- 입력 기능은 입력 파일상의 어떤 특정한 심볼을 읽음
- 제어 장치의 다음 상태는 **전이 함수(transition function)**에 의해 결정됨
- 전이 함수는 현재의 제어 상태, 입력 심볼, 임시 저장 장치 내의 정보들에 의해 결정됨

13.1 오토마타란 무엇인가?

결정적 오토마타와 비결정적 오토마타

정의 13-1 결정적 오토마타(deterministic automata)는 전이에 의한 다음 상태가 현재의 형상에 따라 유일하게 결정되는 오토마타이다. 즉, 현재의 내부 상태, 입력 심볼, 임시 기억 장치에 관한 정보를 알면 오토마타의 다음 행동을 정확히 예측할 수 있다.

정의 13-2 비결정적 오토마타(nondeterministic automata)는 현재의 형상에서 2가지 이상의 이동도 가능한 오토마타로서, 우리는 가능한 이동의 집합을 예측할 수 있을 뿐이다.

13.1 오토마타란 무엇인가?

출력 여부에 따른 오토마타

오토마타는 출력 여부에 따라 인식기와 트랜스듀서로 나누어짐

- **인식기 (accepter)**

  주어진 입력에 대해 인식하거나 기각할 수 있는 기능만을 가짐

- **트랜스듀서 (transducer)**
  - 밀리 기계와 같이 인식이나 기각의 기능 외에 출력도 할 수 있는 오토마타 모델임
  - 앞에서 설명한 음료 자판기의 경우 300원 이상이 입력되면 음료수를 내주고(즉, 출력을 하고) 상태 전이를 계속하므로 트랜스듀서에 해당됨

13.2 오토마타 학습의 필요성과 유한 상태 시스템

오토마타를 학습하는 주된 이유

1) 오토마타 이론이 전산학이나 전자공학 등의 학문을 이해하는 데 있어서 필요한 기본적인 개념과 원리를 제공하고, 여러 가지 응용 분야에 적용할 수 있기 때문임
2) 오토마타 이론을 통하여 컴퓨터와 관련된 이론적인 바탕과 작동의 원리를 보다 정확하게 이해함으로써 하드웨어나 소프트웨어 각 분야에 대한 직관을 넓힐 수 있으며, 보다 포괄적인 통찰력을 제공해줌
3) 오토마타 이론의 직접적인 응용이다. 이를 통하여 디지털 컴퓨터의 디자인, 프로그래밍 언어, 컴파일러, 신경생리학, 통신, 신경망, 언어론 등 다양한 분야에 직접 활용할 수 있다. 특히 컴파일러나 프로그래밍 언어를 정확하게 이해하기 위해서는 오토마타 이론의 이해가 필수적임
4) 추상적 모델의 개념적 이해이다. 복잡한 현상들을 간략하고 정확하게 추상화시킴으로써 연구의 폭을 넓히고 정교한 학문적 탐구가 가능해짐

13.2 오토마타 학습의 필요성과 유한 상태 시스템

정의 ⑬-3 유한 상태 시스템(Finite State Machine : FSM)은 유한개의 상태를 가진 오토마타를 말한다.

유한 상태 시스템의 응용

(1) 엘리베이터의 제어

엘리베이터의 제어 방식은 탑승자들이 과거에 요청했던 모든 요구들을 기억하지 않고, 단지 현재의 층수와 엘리베이터의 운동 방향 및 탑승자들이 요청한 것들 중에서 아직까지 이루어지지 않은 요구들만을 기억하고 있는 유한 상태 시스템임

13.2 오토마타 학습의 필요성과 유한 상태 시스템

(2) 컴퓨터의 제어 장치와 같은 논리 회로

논리 회로는 통상 0과 1로 표시되는 2가지 조건 중 하나의 상태인 유한개의 게이트들의 집합으로 구성된다. 따라서 n개의 **게이트**를 가진 논리 네트워크의 상태는 2^n개 중 하나이다. 이런 점에서 논리 회로를 유한 상태 시스템으로 생각할 수 있음

(3) 문서 편집기와 어휘분석기에도 적용됨

문서 편집기(text editor)는 입력을 추가할 때마다 상태가 변화되고, **어휘분석기(lexical analyzer)**는 컴퓨터 프로그램에서 심볼들을 차례로 읽어서 문자 스트링들을 변수, 상수, 예약어(reserved word) 등으로 대응시킨다. 이 과정에서는 단지 유한개의 정보를 기억하기만 하면 된다. 이와 같이 유한 상태 시스템은 오토마타 이론을 적용하여 여러 가지 형태의 스트링들을 효과적으로 처리하는 데 있어서 매우 유용함

13.2 오토마타 학습의 필요성과 유한 상태 시스템

(4) 컴퓨터도 일종의 유한 상태 시스템임

컴퓨터는 중앙처리장치, 주기억장치, 보조기억장치 등 상태의 개수는 매우 많지만 여전히 유한개의 상태로 이루어져 있다. 여기에서 우리는 한정된 수의 디스크, 드럼, 테이프 등이 사용된다고 가정함

여기서 잠깐!!

인간의 두뇌도 역시 유한 상태 시스템으로 간주될 수 있다. 우리 두뇌 속에 있는 신경 세포라고 불리는 뉴런(neuron)은 약 10조 개 정도로 추산된다. 만약 각 뉴런의 상태를 몇 개의 비트(bit)로써 표현할 수 있다면 인간 두뇌의 모델링도 가능할 것이다. 그러나 실제 인간 두뇌에 있어서는 상태의 개수가 너무나도 많고 작동 메커니즘이 정확하게 규명되지 않아 인간 두뇌의 모델링은 현재의 수준으로는 아직도 요원하다. 하지만 그에 대한 도전은 지속되고 있다.

13.3 유한 오토마타

(1) 유한 오토마타의 기본 개념

유한 오토마타(Finite Automata : FA)

- 이산적인 입력과 출력을 가지는 시스템의 수학적 모형임
- 이 시스템은 유한한 개수의 내부의 상태들을 가지고 있는데, 시스템의 상태는 다음에 들어올 입력에 대한 시스템의 행동을 결정하는데 필요한 과거의 정보들을 요약함
- **인식기(recognizer)**로서의 유한 오토마타를 고찰함
- 유한 오토마타의 주요 특징으로는 임시 저장 장치를 가지지 않는다는 점과 입력 테이프는 단지 읽힐 수만 있으며 그 내용이 변경될 수 없다는 점임
- 유한 오토마타의 기능은 작동 과정에서 정보를 기억하는 데 있어서 상당한 제약을 받게 됨

13.3 유한 오토마타

유한 제어(finite control)

- 유한 오토마타의 입력과 상태들을 제어함
- 유한 오토마타는 여러 종류의 오토마타들 중에서 가장 단순한 형태로서 입력과 유한 제어를 가진 유한 오토마타를 나타냄

〈그림 13.4〉 유한 오토마타

13.3 유한 오토마타

유한 오토마타의 전이함수

- 유한 오토마타는 유한한 상태들의 집합과 **전이 함수(transition function)**들의 집합으로 구성됨
- **전이**란 알파벳 ∑로부터 선택된 입력 심볼에 의해 생기는, 상태에서 상태로의 변화임
- 입력에 따라 상태는 항상 변할 수 있으며, 원래의 상태로 다시 돌아가는 전이도 있을 수 있음

13.3 유한 오토마타

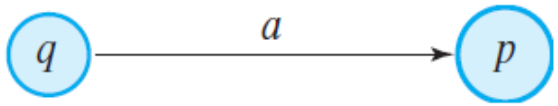
유한 오토마타의 시작 상태, 최종 상태

- q_0(q zero)로 나타내는 상태를 **시작 상태(start state)**라 하는데, 이 시작상태에서 오토마타가 시작됨
- 어떤 상태들은 **최종 상태(final state)** 또는 **인식 상태(accepting state)**들로 지정되는데, 그래프에서는 이중의 서클로 표시함
- 각 유한 오토마타마다 **전이 다이어그램(transition diagram)**이라고 불리는 방향이 있는 그래프는 전이 다이어 그램과 같이 그려짐

13.3 유한 오토마타

전이 다이어그램

- 각 유한 오토마타마다 **전이 다이어그램(transition diagram)**이라 불리는 방향이 있는 그래프는 전이 다이어 그램과 같이 그려짐
- 전이 다이어그램의 각 노드들은 유한 오토마타의 상태에 해당
- 어떤 상태 q에서 입력 a를 받아서 어느 상태 p로 가는 변환이 있다면 전이 다이어그램에서는 상태 q에서 상태 p로 가는 a라는 **연결선**이 존재함
- 입력 스트링 x에 대해 시작 상태에서 최종 상태로 가는 **경로**가 존재한다면 유한 오토마타는 스트링 x를 **인식**하게 됨



〈그림 13.5〉 전이 다이어그램

13.3 유한 오토마타

정규 언어(regular language)

- 유한 오토마타는 전이 방법에 따라 결정적 유한 오토마타와 비결정적 유한 오토마타로 구분됨
- 2개가 기능에 있어서는 서로 동치이며 유한 오토마타에 대응하는 언어임

결정적 유한 오토마타(DFA)

정의 13-4 결정적 유한 오토마타(Deterministic Finite Automata : DFA)는 다음과 같이 5개의 순서쌍으로 이루어진다.

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

Q : 상태들의 유한 집합(finite set of states)
Σ : 입력 알파벳(input alphabet)이라고 불리는 유한개의 심볼들의 집합
δ : $Q \times \Sigma \rightarrow Q$인 전이 함수(transition function)
q_0 : $q_0 \in Q$인 시작 상태(start state)
F : $F \subseteq Q$인 최종 상태의 집합(set of final states)

13.3 유한 오토마타

DFA의 작동 개요

- 처음에는 시작 상태가 q_0 에 있고 유한 제어는 입력 스트링의 가장 왼쪽에 있는 심볼을 가리킴
- 오토마타의 작동에 따라 입력 장치에서 한 심볼씩 오른쪽으로 이동하면서 상태가 바뀜
- 스트링을 모두 읽고 난 후 DFA가 최종 상태에 있으면 그 스트링이 **'인식되고'(accepted)** 그렇지 않으면 **'기각된다'(rejected)**
- 입력 메커니즘은 왼쪽에서 오른쪽으로의 방향으로만 이동이 가능하며 각 단계에서 하나씩의 심볼만을 읽을 수 있다는 점에 유의

여기서 잠깐!!

참고로 스트링이 인식된다는 것은 다른 말로 '받아들여진다' 라고도 표현하며, 기각된다는 것은 '받아들여지지 않는다' 로 표현하기도 한다.

13.3 유한 오토마타

(2) 출력이 없는 유한 오토마타

M이 다음과 같이 주어졌을 때 이것을 수식으로만 본다면 어떤 입력이 인식되는지 또는 기각되는지를 알기 어려움

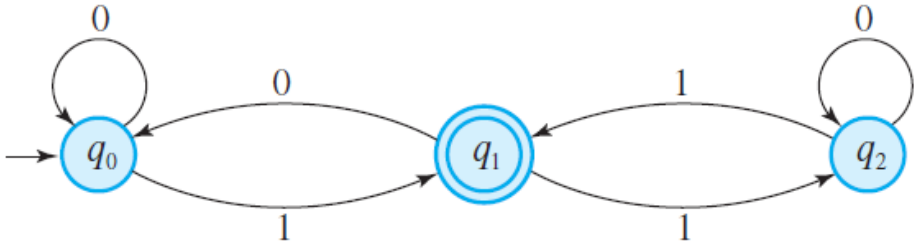
$$\text{DFA } M=(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

$$\begin{aligned} \delta : \delta(q_0, 0) &= q_0 \qquad \delta(q_0, 1) = q_1 \\ \delta(q_1, 0) &= q_0 \qquad \delta(q_1, 1) = q_2 \\ \delta(q_2, 0) &= q_2 \qquad \delta(q_2, 1) = q_1 \end{aligned}$$

- 위의 식을 아래의 그래프로 나타내면 훨씬 편리함
- q_0는 시작 상태이므로 통상 앞에 작은 화살표를 그려서 표시하고 q_1은 이중의 원으로 최종 상태를 나타냄

13.3 유한 오토마타

DFA의 예



〈그림 13.6〉 DFA의 예

13.3 유한 오토마타

001이라는 스트링이 DFA M에 입력되었을 경우

1. 시작 상태 q_0에서 0이라는 최초의 심볼을 읽었을 때 상태는 다시 q_0에 머무르게 됨
2. q_0에서 두 번째 입력 0을 읽고 상태는 또 다시 q_0에 머무르게 됨
3. q_0에서 다시 1이라는 마지막 심볼을 읽고 상태는 q_1으로 바뀜(이동함)

   이때 1이 마지막 심볼이고 상태 q_1이 최종 상태이므로 스트링 001은 인식됨

- 스트링 01, 0101, 1101 등도 인식됨을 확인함
- 00, 110, 011 등은 입력 스트링을 모두 읽고 나서도 최종 상태에 도달하지 않으므로 기각됨

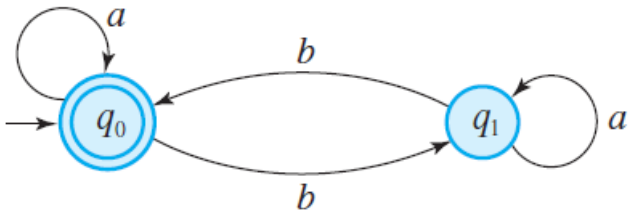
13.3 유한 오토마타

예제 13-1

짝수 개의 b를 가지는 $\Sigma = \{a, b\}$상의 모든 스트링들을 인식하는 DFA를 디자인해보자.

풀이 이 경우에는 2개의 상태 q_0, q_1을 이용하는데, q_0에서 출발하여 a가 나오면 q_0에 머물고, b가 나오면 q_1으로 이동한다. q_1에서 a가 나오면 q_1에 머물고, b가 나오면 q_0로 이동한다. 즉, 상태 q_0는 b의 개수가 짝수이고 q_1은 b의 개수가 홀수가 되는 상태이다.

따라서 이에 해당하는 DFA는 다음과 같다.



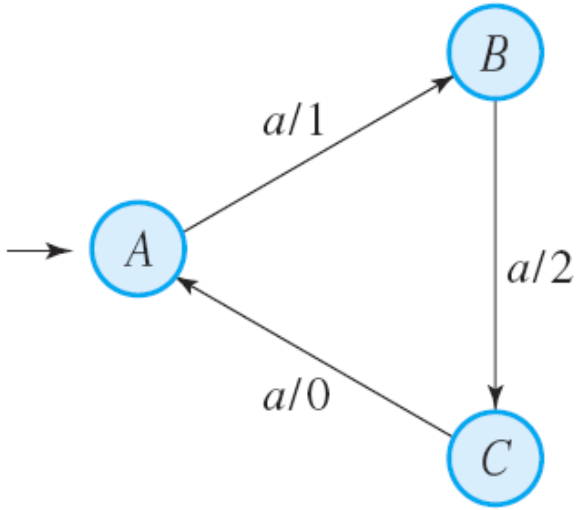
13.3 유한 오토마타

(3) 출력이 있는 유한 오토마타

- 어떤 입력의 개수를 읽는 순간마다 알 수 있는 것이 **카운터(counter)**란 장치임
- 어떤 수를 3으로 나누어서 나머지를 출력해내는 오토마타가 바로 **Mod-3 카운터**임
- 이것을 출력이 있는 유한 오토마타로 나타낸 것과 같음
- 상태 A에서 a가 들어오면 1을 출력하면서 상태 B로 이동하고, 상태 B에서 입력 a가 들어오면 2를 출력하면서 상태 C로 이동함
- 상태 C에서 a가 들어오면 0을 출력하면서 상태 A로 이동하게 되며 추가적인 입력이 있을 경우에는 위의 과정을 반복함

13.3 유한 오토마타

출력이 있는 유한 오토마타의 예(Mod 3 카운터)



〈그림 13.7〉 출력이 있는 유한 오토마타

13.3 유한 오토마타

(4) 유한 오토마타의 응용

- 유한 오토마타를 이용하여 변수를 만드는 방법에 대해 알아봄
- C언어에 있어서 모든 **변수(identifier)**들의 집합은 하나의 언어임
- 각 변수는 하나의 **문자(letter)**로 시작되어야 하며, 문자 다음에는 임의 개수의 문자나 **숫자(digit)**가 혼용하여 사용됨
- 다음의 문법은 변수에 대한 정확한 정의를 나타내는 규칙들임

```
<id>       →    <letter><rest>
<rest>     →    <letter><rest> | <digit><rest> | λ
<letter>   →    a | b | ··· | z | _
<digit>    →    0 | 1 | ··· | 9
```

13.3 유한 오토마타

변수의 생성

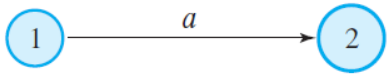
이 문법에서 변수들로는 <id>, <letter>, <rest>, <digit>이며 터미널 심볼들은 a, b, ..., z, _ , 0, 1, ..., 9인데, 여기서 '_'는 **underscore**임

예를 들어, 변수 x5의 유도 과정은 다음과 같음

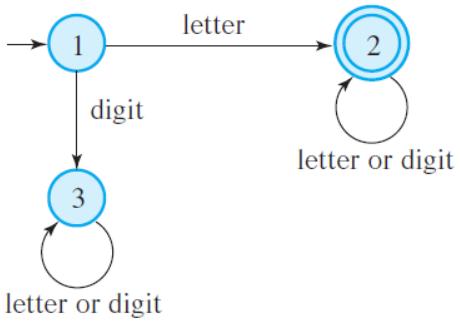
```
<id>      →    <letter><rest>
            →    x<rest>
            →    x<digit><rest>
            →    x5<rest>
            →    x5
```

13.3 유한 오토마타

- **오토마타는 그래프를 이용하여 표현하는 것이 매우 편리함**
- <그림 13.8>에서 상태 1로부터 상태 2로의 전이는 입력 심볼이 a인 경우 일어날 수 있음을 보여주는데, 이러한 그래프를 전이 다이어그램 또는 전이 그래프라고 함
- <그림 13.9>는 전이 그래프를 이용하여 C언어의 변수를 인식하는 오토마타를 나타냄



〈그림 13.8〉 입력 a에 의한 상태 전이



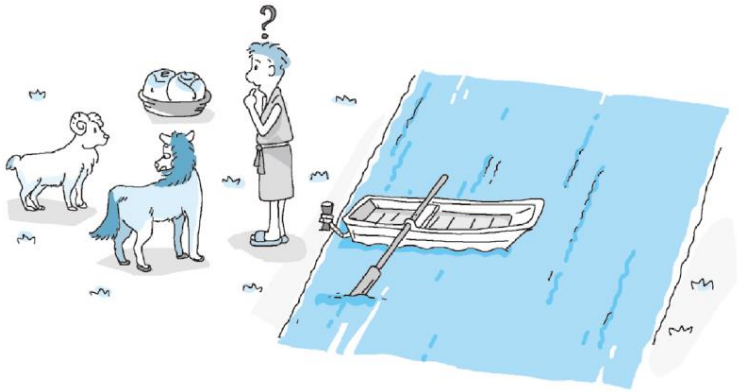
〈그림 13.9〉 C언어에서의 변수를 인식하는 오토마타

13.3 유한 오토마타

예제 13-2

문제 해결을 위한 유한 오토마타의 예로 다음과 같이 강을 건너는 문제를 살펴보자.

어떤 사람이 늑대, 염소 그리고 양배추와 더불어 강의 왼쪽 기슭에 있다고 생각하자. 사람은 한 번에 늑대나 염소 또는 양배추 중 하나만 선택하여 강의 왼쪽 기슭이나 오른쪽 기슭을 왕복할 수 있다. 물론 사람 혼자서 건널 수도 있다.



만약 사람이 늑대와 염소를 어느 한쪽 기슭에 같이 남겨 둔다면 늑대는 사람이 없는 틈을 타서 염소를 잡아먹게 될 것이며, 염소와 양배추만 한쪽 기슭에 남겨 둔다면 염소는 양배추를 먹어 치우게 될 것이다. 그렇다면 어떻게 염소나 양배추가 먹히지 않고 사람에 의해 강을 무사히 건너갈 수 있을까?

13.3 유한 오토마타

풀이 우리는 사람이 강을 한 번 건너간 후, 강의 왼쪽 기슭에 무엇이 남겨져 있는지에 대한 관련 정보를 살펴봄으로써 이 문제를 유한 오토마타로 표현할 수 있다. 이 경우 사람(M), 늑대(W), 염소(G), 양배추(C)의 4가지의 모든 가능한 결합인 16가지 경우의 부분 집합들이 있을 수 있다.
여기서 하나의 상태가 강의 왼쪽 기슭에 있는 것의 부분 집합에 해당되면 우리는 하이픈(hyphen)을 이용하여 그러한 상태들을 표시할 수 있다. 하이픈의 왼쪽에 있는 심볼들은 강의 왼쪽 기슭에 무엇이 있는지를 나타내고, 하이픈의 오른쪽에 있는 심볼들은 강의 오른쪽 기슭에 무엇이 있는지를 나타낸다.

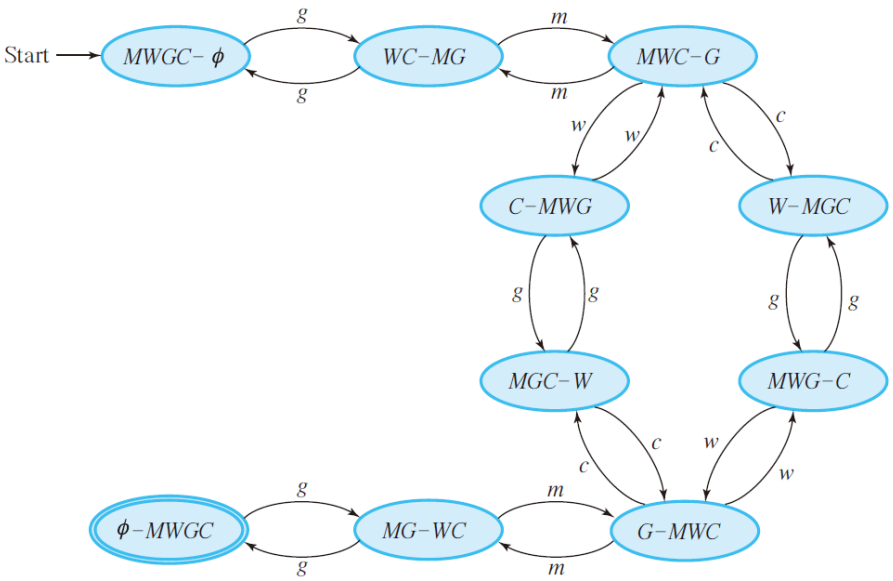
예를 들어, $MG-WC$로 표시된 상태는 강의 왼쪽 기슭에 사람(M)과 염소(G)가 있고, 강의 오른쪽 기슭에는 늑대(W)와 양배추(C)가 있음을 나타낸다. 16가지 상태들 중에서 $GC-MW$와 같은 몇몇 상태들은 실패하는 경우를 의미하므로 이 시스템에서 사용되지 않는다.

이 시스템에서의 입력은 사람이 취하는 행동이다. 즉, 사람은 혼자서만 강을 건너거나(입력 m), 늑대와 함께 건너거나(입력 w), 염소와 함께 건너거나(입력 g), 양배추와 함께 강을 건널 수 있다(입력 c).

따라서 시작 상태는 $MWGC-\phi$로 나타낼 수 있으며 최종 상태는 $\phi-MWGC$로

13.3 유한 오토마타

나타낼 수 있다. 다음의 〈그림 13.10〉은 이에 대한 유한 오토마타를 나타낸 것이다.



〈그림 13.10〉 사람, 늑대, 염소, 양배추 문제의 유한 오토마타

13.3 유한 오토마타

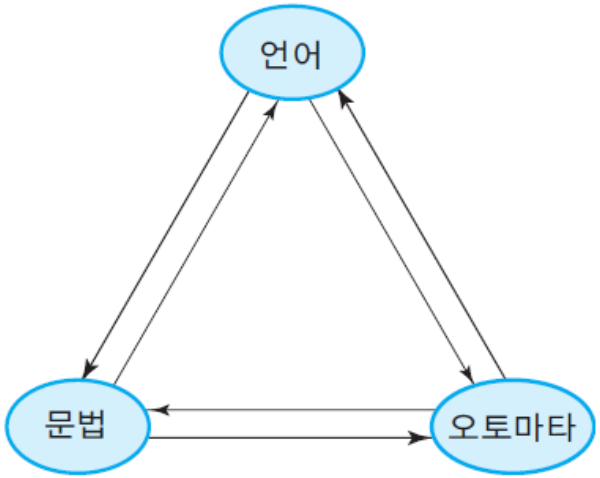
Start로 표시된 시작 상태에서 이중의 원으로 표시된 최종 상태에 이르는 방법은 2가지가 있는데, 똑같은 수의 단계를 거친 수많은 해답들 중에서 이 2개의 간결한 해답을 제외하면 다른 해답들은 불필요한 사이클을 포함하고 있다. 예를 들어, *G–MWC*에서 *MGC–W*나 *MWG–C*로 되돌아갈 수도 있다.

위와 같은 문제를 해결하는 데 있어 주목할 점은 추상적 모델링 단계와 수학적 모델링 단계에서 강을 건너는 상황을 문자로 표현하고, 그것을 상태의 유한 오토마타로 그렸다는 점이다. 이와 같은 방법론은 이산수학의 여러 가지 문제들에 적용될 수 있다.

13.4 형식언어와 문법

오토마타 이론에서 가장 중요한 3가지 개념

- **언어(language), 문법(grammar), 오토마타(automata)**를 들 수 있음
- 이 세 가지는 상호 간에 깊은 관련성을 가지고 있기 때문에 각각의 개념을 파악하고 그들 간의 관계를 탐구하는 것은 매우 중요함



〈그림 13.10〉 언어, 문법, 오토마타의 관계

13.4 형식언어와 문법

(1) 언어(language)

- 언어에는 한글을 비롯하여 영어 등 우리가 일상생활에서 자주 사용하는 **자연어(natural language)**와 오토마타를 이용하여 만들어지는 이론적인 언어인 **형식 언어(formal language)**의 2가지임
- 여기에서는 형식 언어와 관련된 기본적인 용어와 표현들을 살펴봄

정의 ⑬-5 알파벳(alphabet)은 유한개의 공집합이 아닌 심볼들(symbols)의 집합 Σ이며, 이들 심볼들로부터 스트링들(strings)이 만들어지는데, 알파벳으로부터 얻어진 심볼들의 유한개의 시퀀스(sequence)로 이루어진다.

정의 ⑬-6 팰린드롬(palindromes)은 aba나 abba와 같이 앞에서 읽으나 뒤에서 읽으나 똑같은 스트링들을 말한다.

13.4 형식언어와 문법

스트링에서의 연산

① 연결(concatenation)

두 개의 스트링 $w = a_1 a_2 a_3 \cdots a_n$와 $v = b_1 b_2 b_3 \cdots b_n$를 연결하면 w와 v의 연결 $wv = a_1 a_2 a_3 \cdots a_n b_1 b_2 b_3 \cdots b_n$이다.

② 역(reverse)

어떤 스트링의 역은 주어진 심볼들을 거꾸로 나열한 것이다. 즉, $w = a_1 a_2 a_3 \cdots a_n$이라면 그것의 역인 w^R은 $w^R = a_n \cdots a_3 a_2 a_1$이 된다. 예를 들면, $(\text{cat})^R = \text{tac}$가 된다.

③ 스트링의 길이

스트링의 길이는 스트링에 포함된 심볼의 개수를 말하는데 $|w|$와 같이 절대값을 써서 나타낸다. 만일 주어진 스트링이 어떠한 심볼도 가지고 있지 않을 때에는 공스트링(empty string)이라 하고 통상 λ로 나타낸다. 따라서 $|\lambda| = 0$이고 모든 w에 대해 $\lambda w = w\lambda = w$가 성립한다.

13.4 형식언어와 문법

④ 스트링의 반복

w가 스트링일 때 w^n이란 w를 n번 연결한 것이며 모든 w에 대해 $w^0 = \lambda$가 된다.

⑤ Σ^*와 Σ^+

Σ가 알파벳일 때 Σ^*는 Σ상에서의 심볼들의 결합으로부터 만들어지는 모든 스트링들의 집합인데, 이때 Σ^*는 λ를 항상 포함하게 된다. Σ^*에서 λ를 제외할 경우에는 Σ^+로 표현한다.

예를 들어, $\Sigma = \{a, b\}$일 때

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

$\Sigma^+ = \{a, b, aa, ab, ba, bb, aaa, aab, \cdots\}$가 된다.

⑥ 언어와 문장

언어란 일반적으로 Σ^*의 부분 집합으로 정의되는데 통상 대문자 L로 표현된다. 언어 L 안에 있는 어떤 스트링 w는 L의 단어(word) 또는 문장(sentence)으로 불린다.

$\Sigma = \{a, b\}$일 때 집합 $\{a, ab, aabb\}$는 Σ상에서의 언어가 된다.

⑦ 언어에서의 연산

언어는 단어들로 이루어진 집합이므로 합집합, 교집합, 차집합 등 집합의 일반적인 연산이 가능하다. 그 외에도 여집합, 연결 등의 연산이 가능하다.

13.4 형식언어와 문법

(2) 문법(grammar)

- 우리말이나 영어의 경우에는 부정확하고 애매한 경우가 상당히 많음
- 컴퓨터에서 쓰이는 문법에는 애매성이 배제되어야 하며 일정한 규칙을 따라 엄밀하게 정의되어야 함
- 컴퓨터에 사용되는 문법은 배커스와 나우어에 의해 체계화된 **배커스-나우어 표기법(Backus-Naur Form: BNF)**과 푸시다운 오토마타(Push-Down Automata : PDA)에 의한 **문맥자유 문법(Context-Free Grammer : CFG)**으로 구별될 수 있음
- 표현 방법은 다르지만 기능상에서는 차이가 없음

13.4 형식언어와 문법

- 문법은 어떤 문장이 제대로 작성되었는지의 여부를 판정하는 기준이 됨
- 한가지 전형적인 규칙의 예는 '문장에서는 명사절 다음에 서술어가 온다'의 경우를 들 수 있음
- 이것을 보다 구체적으로 표현하면

```
<sentence>      →      <noun_phrase><predicate>
```

- 세분화하면

```
<noun_phrase>  →      <article><noun>
<predicate>    →      <verb>
```

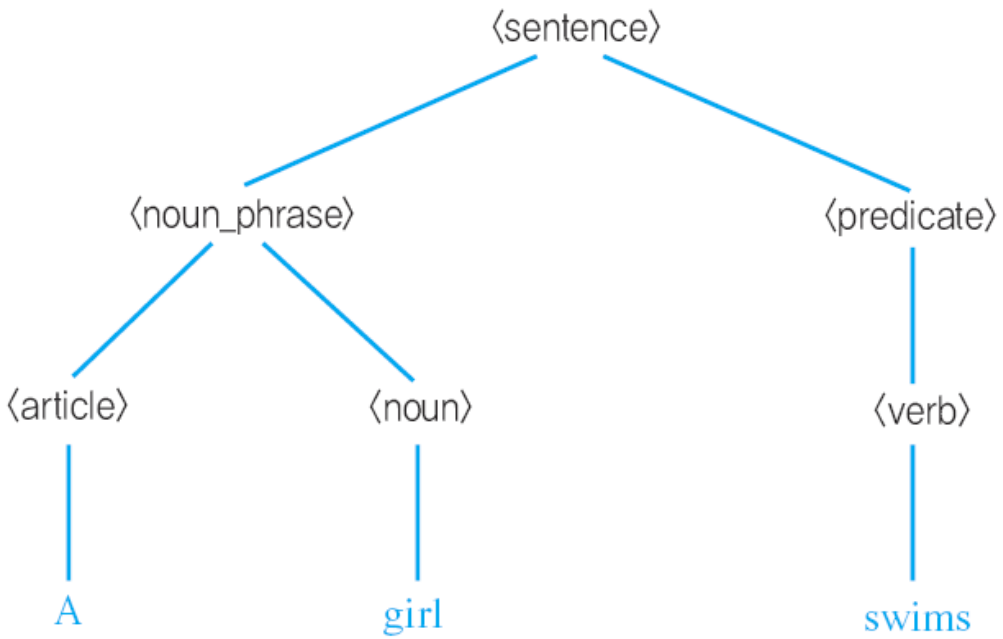
13.4 형식언어와 문법

예를 들어

```
<article>    →    A | The
<noun>       →    girl | cat
<verb>       →    swims | jumps
```

'A girl swims' 또는 'The cat jumps' 등은 문법에 맞는 문장이 됨

13.4 형식언어와 문법



⟨그림 13.11⟩ 영어 문장 구조의 예

13.4 형식언어와 문법

정의 13-7 문법(Grammar) G는 다음과 같은 4개의 순서쌍으로 정의된다.

$G = (N, T, P, S)$

1. N : 변수(variable)라고 불리는 넌터미널 심볼(nonterminal symbol)들의 유한 집합인데 통상 대문자 알파벳으로 표현된다.
2. T : 터미널 심볼(terminal symbol)의 유한 집합인데 통상 소문자 알파벳으로 표현된다.
3. P : 생성 규칙(production rule)의 유한 집합인데 집합 P는

   $A \rightarrow X_1 X_2 \cdots X_n$

   와 같이 표현된다. 여기서 A는 넌터미널이고 $X_1 X_2 \cdots X_n$은 유한한 길이의 터미널 또는 넌터미널 심볼이다.
4. S : N에 속하는 시작 심볼(start symbol)로서 문장의 시작을 나타내며 통상 S를 사용한다.

정의 13-8 $\Rightarrow$는 유도된다(derived)라고 말하는데, $w \Rightarrow z$일 때 'w는 z를 유도한다' 또는 'z는 w로부터 유도된다' 라고 표현한다.

13.4 형식언어와 문법

정의 13-9 $G = (N, T, P, S)$가 문법일 때 $L(G) = \{w \in T^* | S \overset{*}{\Rightarrow} w\}$는 문법 G에 의해 생성된 언어이다.

정의 13-10 $S \overset{*}{\Rightarrow} w$이고 w가 $(N \cup T)^*$에 속하면 w를 문장 형태(sentential form)라고 하며, w가 T^*에 속할 경우 w를 문장(sentence)이라고 한다. 즉,

$$L(G) = \{w \mid wT^*,\ S \overset{*}{\Rightarrow} w\}$$

에서 w는 $L(G)$의 문장이다.

13.4 형식언어와 문법

예제 13-2 $L = \{a^n b^{n+1} \mid n \geq 0\}$을 생성하는 문법을 만들어보자.

풀이 먼저 앞의 예에서 b가 하나 추가된 것에 힌트를 얻을 수 있다. 그러므로

$$S \rightarrow Ab$$
$$A \rightarrow aAb$$
$$A \rightarrow \lambda$$

는 위의 요구를 만족시킬 수 있다.

$$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow aaAbbb \Rightarrow a^n b^{n+1}$$

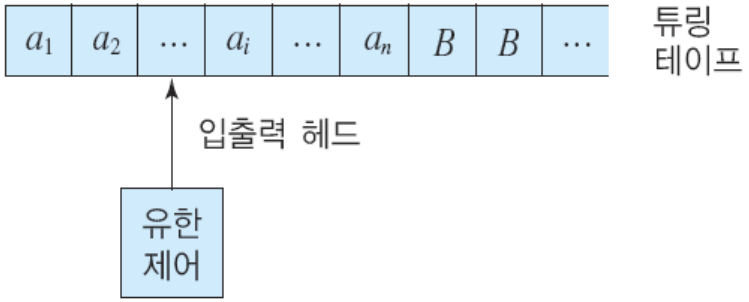
정의 13-11 두 개의 문법 G_1, G_2가 똑같은 언어 L을 생성할 때 G_1과 G_2는 동치(equivalent)라고 한다.

13.4 형식언어와 문법

- **효과적인 프로시저(procedure)를 수행할 수 있는 모델**이 되기 위해서는 다음과 같은 특성을 가짐

  첫째, 각 프로시저는 유한하게 기술될 수 있어야 함

  둘째, 프로시저는 기계적으로 수행되는 이산적인 단계들로 이루어짐
- 이러한 조건을 만족하는 최초의 모델은 1936년 영국의 수학자인 **앨런 튜링(Alan Turing)**에 의해 만들어진 **튜링머신(Turing Machine)**임



〈그림 13.12〉 기본적인 튜링머신

13.5 튜링머신

정의 ⑬-12 튜링머신 M은 다음과 같은 7개의 쌍으로 정의된다.

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Q는 상태들의 유한 집합이다.

Γ는 허용되는 테이프 심볼들의 유한 집합이다.

B는 Γ에 속하는 블랭크(blank)이다.

Σ는 입력 심볼의 집합으로서 B를 포함하지 않으며 Γ의 부분 집합이다.

δ는 다음 동작 함수(next move function)인데 $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$이다.
여기서 δ는 경우에 따라 정의되지 않을 수도 있다. 여기서 L과 R은 각각 왼쪽과 오른쪽으로의 이동을 의미한다.

q_0는 시작 상태로서 $q_0 \in Q$이다.

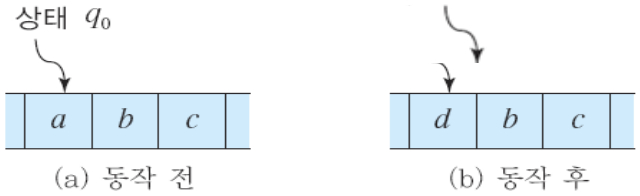
$F \subseteq Q$는 최종 상태의 집합이다.

13.5 튜링머신

튜링머신 (Turing Machine)

- **튜링머신**은 그것의 임시 기억 장치가 테이프인 오토마타인데, 이 테이프는 셀들로 나누어져 있는데 각 셀은 하나씩의 심볼을 가짐
- 테이프의 왼쪽이나 오른쪽으로 이동하면서 각 동작마다 하나의 심볼을 읽고 하나의 심볼을 쓰는 **입출력 헤드(read-write head)**가 존재하며 제어 유니트의 명령에 따라 동작하게 됨

다음의 전이에 의한 동작의 전과 후의 상황은 다음과 같음



〈그림 13.13〉 전이에 의한 튜링머신의 동작의 전후 상황

13.5 튜링머신

정의 13-13 튜링머신 M에 의해 인식되는 언어인 $L(M)$은 M의 테이프 헤드가 가장 왼쪽 셀에 위치하고 q_0 상태에서 시작하여 M이 최종 상태에 들어가게 하는 Σ^* 내의 단어들의 집합이다. 즉,

$$L(M) = \{w \in \Sigma^* | q_0 w \vdash a_1 p a_2 \text{ for some } p \in F \text{ and } a_1, a_2 \in \Gamma^*\}$$

이 된다.

여기서 잠깐!! 어떤 튜링머신 M이 언어 L을 인식하게 된다는 것은 튜링머신의 상태가 최종 상태에 들어가면서 정지하는 경우이다. 그러나 인식되지 않을 때에는 튜링머신이 정지하지 않을 수도 있다.

13.5 튜링머신

예제 13-3 다음과 같이 정의된 튜링머신의 작동 예를 살펴보자.

$$Q = \{q_0, q_1\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{a, b, B\}$$
$$F = \{q_1\}$$

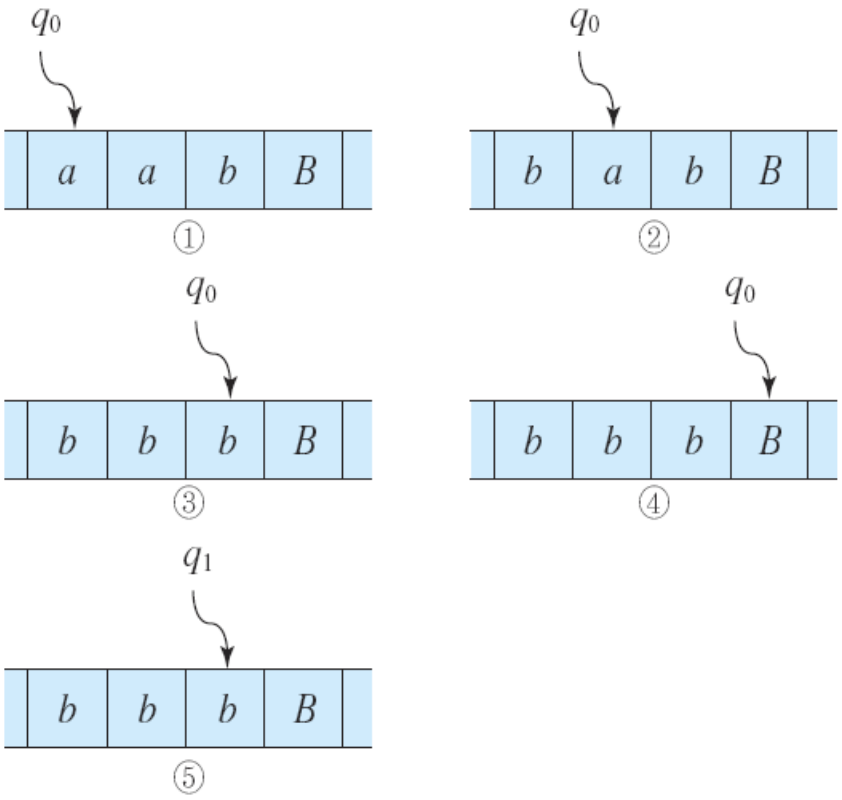
$$\delta(q_0, a) = (q_0, b, R)$$
$$\delta(q_0, b) = (q_0, b, R)$$
$$\delta(q_0, B) = (q_1, B, L)$$

13.5 튜링머신

풀이 만약 이 튜링머신의 테이프 헤드가 심볼 a에서 출발한다면 적용될 전이 규칙은 $(q_0, a) = (q_0, b, R)$이다. 그러므로 테이프의 헤드는 a 대신에 b로 치환하고 오른쪽으로 한 칸 이동하는데 q_0 상태에 계속 머무르게 된다.

그 다음의 a는 b로 바뀌고 테이프 헤드는 오른쪽으로 한 칸 이동한다. 그 후 q_0에서 b를 만났을 경우에는 위의 두 번째 전이 규칙에 따라 b는 바뀌지 않고 테이프 헤드만 한 칸 오른쪽으로 이동한다. 튜링머신이 첫 번째 블랭크(B)를 만날 경우에는 테이프 헤드는 한 칸 왼쪽으로 이동하면서 최종 상태인 q_1에서 멈추게 된다. 〈그림 13.14〉는 초기의 형상에서부터 출발하여 여러 단계를 거치는 과정을 보여준다.

13.5 튜링머신



〈그림 13.14〉 튜링머신의 작동 예

13.6 촘스키 포함 관계

- 형식 언어의 선구자 촘스키(Chomsky, 1928~현재)는 4가지 문법의 패밀리에다 숫자를 붙여서 포함 관계를 나타냄
- **무제한 문법, 문맥민감 문법, 문맥자유 문법, 정규 문법을 각각 Type 0, Type 1, Type 2, Type 3 문법으로** 명명하였는데, 문법의 숫자가 커질수록 제한도 많아짐
- 문법의 포함 관계는 해당 언어들의 포함 관계로 이어져서 소위 **촘스키 포함 관계**를 형성함
- 모든 Type *i* 언어는 Type(*i*-1) 언어에 속하는데, 이러한 포함 관계는 진부분 집합의 관계임

13.6 촘스키 포함 관계

정리 13-1

촘스키 포함 관계

(1) 정규 언어는 CFL(Context-Free Language)의 진부분 집합이다.

(2) 공스트링이 아닌 CFL은 CSL(Context-Sensitive Language)의 진부분 집합이다.

(3) CSL은 r. e(recursively enumerable) 언어의 진부분 집합이다.

13.6 촘스키 포함 관계

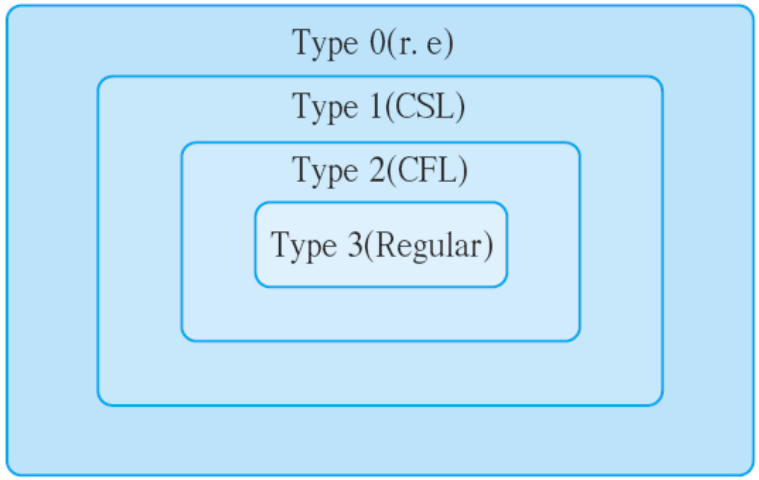
주요 4가지 타입의 오토마타 요약 (문법, 언어, 오토마타)

〈표 13.1〉 4가지 타입의 오토마타

문법	언어	오토마타
Type 0 (무제한 문법)	r.e Language	튜링머신
Type 1 (문맥민감 문법)	CSL	선형제한 오토마타
Type 2 (문맥자유 문법)	CFL	푸시다운 오토마타
Type 3 (정규 문법)	Regular Language	유한 오토마타

13.6 촘스키 포함 관계

촘스키 포함 관계



〈그림 13.15〉 촘스키 포함 관계

13.7 오토마타 응용과 4차 산업혁명과의 관계

(1) 오토마타의 응용 분야

① 오토마타 응용의 다양한 분야들

오토마타의 간단한 응용으로는 물시계와 모래시계 등을 들 수 있으며, 근래에는 〈그림 13.17〉과 같은 나무로 만든 장난감이나 뻐꾸기시계, 음료수 자판기 등에 다양하게 응용되고 있다. 그 외에도 오토마타 원리는 엘리베이터 제어, 디지털 디자인, 논리 제어, 시스템 설계, 신경생리학, 통신, 신경망, 언어론 등 다양한 분야에 직접 활용되고 있다.

〈그림 13.17〉 나무로 만든 장난감 오토마타

13.7 오토마타 응용과 4차 산업혁명과의 관계

② 컴퓨터 관련 분야의 응용들

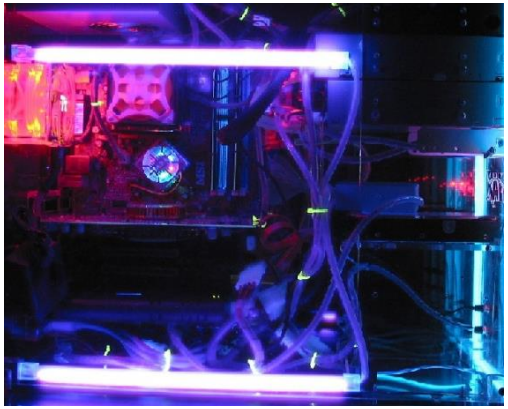
오토마타는 디지털 컴퓨터의 추상적 모델로서 디지털 컴퓨터가 작동하는 이론적인 메커니즘이라 볼 수 있다. 특히 인간과 컴퓨터를 이어주는 C나 파이선을 비롯한 각종 프로그래밍 언어도 오토마타 원리에 의해 만들어졌으며, 그것을 번역하는 컴파일러(Compiler)도 오토마타의 주요 응용 영역에 속한다.

13.7 오토마타 응용과 4차 산업혁명과의 관계

(2) 오토마타와 4차 산업혁명과의 관계 : 광 컴퓨터

광 컴퓨터(Optical computer)는 〈그림 13.18〉과 같이 광 신호로 작동하는 논리 소자를 이용한 신호를 통하여 빛에 의해 연산하는 컴퓨터를 말하는데, 광 컴퓨터가 개발되면 패턴 인식, 로봇의 제어 등에서 큰 발전이 기대된다.

4차 산업혁명의 한 기술 분야인 광 컴퓨터는 광 재료 및 디바이스, 기억소자, 상호 연결된 네트워크 등에서 오토마타에 의해 정교하게 제어되어야 하므로 오토마타의 역할과 응용이 특히 필요한 분야이다.



〈그림 13.18〉 광 컴퓨터

요약 및 생활 속의 응용

- 오토마타란 디지털 컴퓨터의 수학적 모델인 오토마톤의 복수형으로서 '자동 기계 장치' 란 뜻을 가지고 있다.
- 오토마타는 디지털 컴퓨터의 추상적인 모델인데, 입력 장치, 출력 장치, 저장 장치, 제어 장치를 가지고 있으므로 현대적인 디지털 컴퓨터가 작동하는 이론적인 메커니즘이라 볼 수 있다.
- 결정적 오토마타는 전이에 의한 다음 상태가 현재의 형상에 따라 유일하게 결정되는 오토마타이며, 비결정적 오토마타는 현재의 형상에서 2가지 이상의 이동도 가능한 오토마타이다.
- 오토마타는 출력 여부에 따라 인식기와 트랜스듀서로 나누어지는데, 인식기는 주어진 입력에 대해 인식하거나 기각할 수 있는 기능만을 가지지만, 트랜스듀서는 인식이나 기각의 기능 외에 출력도 할 수 있다.
- 유한 오토마타는 이산적인 입력과 출력을 가지는 시스템의 모델이고, 유한 상태 시스템은 유한개의 상태를 가진 오토마타를 말한다.

요약 및 생활 속의 응용

- 오토마타는 보통 q_0로 나타내는 시작 상태에서 시작하여, 최종 상태 또는 인식 상태에서 끝나는데, 그래프에서는 최종 상태를 이중의 서클로 표시한다.

- 결정적 유한 오토마타(DFA)는 다음과 같은 5개의 순서쌍으로 이루어진다.

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

  Q : 상태들의 유한 집합
  Σ : 입력 알파벳들의 집합
  δ : $Q \times \Sigma \rightarrow Q$인 전이 함수
  q_0 : $q_0 \in Q$인 시작 상태
  F : $F \subseteq Q$인 최종 상태의 집합

- 유한 오토마타에는 인식이나 기각을 하는 출력이 없는 오토마타와 출력이 있는 오토마타가 있다.

- 오토마타 이론에서 가장 중요한 3가지 개념으로는 언어, 문법, 오토마타를 들 수 있는데, 이 세 가지는 상호 간에 깊은 연관성을 가지고 있다.

요약 및 생활 속의 응용

- 알파벳은 유한개의 심볼들의 집합 Σ이며, 이들 심볼들로부터 스트링들이 만들어지는데, 알파벳으로부터 얻어진 심볼들의 유한개의 시퀀스로 이루어진다.
- 팰린드롬은 *aba*나 *abba*와 같이 앞에서 읽으나 뒤에서 읽으나 똑같은 스트링을 말한다.
- 스트링에서의 연산에는 연결, 역, 스트링의 반복 등이 있다.
- 문법 G는 다음과 같은 4개의 순서쌍으로 정의된다. $G = (N, T, P, S)$, 여기서 N : 넌터미널 심볼들의 유한 집합, T : 터미널 심볼의 유한 집합, P : 생성 규칙의 유한 집합, S : 시작 심볼이다.
- 튜링머신은 1936년 영국의 수학자인 앨런 튜링에 의해 만들어졌는데, 각 프로시저는 유한하게 기술될 수 있어야 하며, 이산적인 단계들로 이루어진다.

요약 및 생활 속의 응용

- 형식 언어의 선구자 촘스키는 무제한 문법, 문맥민감 문법, 문맥자유 문법, 정규 문법을 각각 type 0, type 1, type 2, type 3 문법으로 명명하였는데, 문법의 숫자가 커질수록 제한도 많아진다.
- 문법의 포함 관계는 해당 언어들의 포함 관계로 이어져서 소위 촘스키 포함 관계를 형성하는데, 모든 type i 언어는 type$(i-1)$ 언어의 진부분 집합이다.
- 오토마타의 응용은 디지털 디자인, 프로그래밍 언어, 컴파일러, 신경생리학, 통신, 신경망, 언어론 등 다양한 분야에 직접 활용할 수 있다.
- 유한 상태 시스템의 응용 예는 엘리베이터 제어, 논리 제어, 컴퓨터, 문서 편집기, 어휘분석기 등이 있다.
- 오토마타와 4차 산업혁명과의 관계로는 광 컴퓨터를 들 수 있다.

CHAPTER 13
FINISH