3. Basics of Classical Computer

Vaughan Sohn

October 6, 2024

Contents

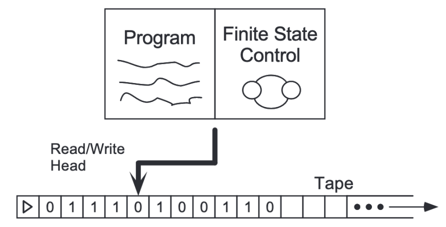
Turing machine

Components of a Turing machine

- State: TM이 가질 수 있는 state들의 finite set $\{Q\}$
- Transition function: TM이 tape에서 symbol을 읽고 자신의 state에 따라서 어떻게 동작할 것인지를 결정하는 *partial* function.

$$\delta : Q \times S \rightarrow Q \times S \times \{L, N, R\}$$

- Tape: input string이 적인 infinite length의 메모리.
  - S: tape에 쓰일 수 있는 모든 symbol들의 finite set $\{blank, \#, *\} \subseteq A$
  - A: S의 부분집합 $\{blank, \#\} \nsubseteq A$
- Read/Write Head: tape에서 symbol을 읽고 쓰는 위치를 나타내는 head.



Operation of a Turing Machine

turing machine M, 그리고 input string x에 대하여,

1. initial state q_0으로 초기화한다.
2. input string x를 tape에 작성한다. (남은 부분은 blank)
3. halt가 되지 않으면, 매 step마다 다음을 반복한다.
   3.1 head위치 i에서 symbol s_i를 읽는다.
   3.2 transition function을 참고하여 (q, s_i)일 때, 어떻게 행동해야하는지를 구한다.
   3.3 transition function의 규칙에 따라 symbol, state, head를 업데이트한다.
   $\Rightarrow (q', s'_i, i + \Delta h)$

Halting 규칙은 다음과 같다.

- head가 tape의 범위를 벗어났을 때
- (q, s_i)에 대한 transition function이 정의되지 않았을 때
- $s_i = \#$일 때

Universial Turing machine

Definition 1

A machine that can solve any problem is called a **Universial Turing machine**.

- UTM은 자신이 시뮬레이션 할 TM의 index i와 그 TM의 시뮬레이션에 사용할 input$x$을 입력으로 받게 된다. $(i,\ x)$
- UTM의 program; transition function안에는 각각의 TM에 대한 동작이 모두 기술되어있기 때문에 그 내용을 따라서 진행하면 $M_i(x)$의 결과를 얻게되고, 이를 output으로 반환하게된다.

$$U(i, x) = \varphi_{M_i}(x)$$

- 각 Turing machine의 state, symbol은 모두 finite set이며, 따라서 transition function도 finite하기 때문에 모든 TM에 대한 내용을 하나의 TM U에 담을 수 있다.

Definition 2

A partial function $f : A^* \to A$ is computable if there exists a Turing machine M such that $\varphi_M = f$. In this case, we say that f is computed by M.

Church-Turing thesis

The class of functions computable by a Turing machine corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm.

- Church-Turing thesis는 실제 알고리즘과 튜링머신을 연관시켜 생각하는 강력한 가설이다.
- 그러나, TM이 모든 function을 계산할 수 있는 것은 아니다! (e.g., Halting problem)

Halting problem

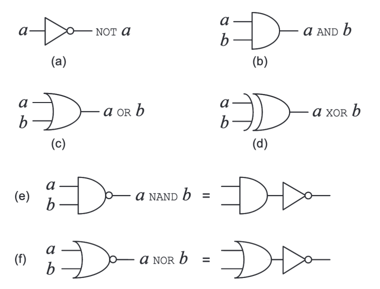
Dose turing machine M halt for given input x?
$\rightarrow$ *We can't compute halting problem by any turing machine!*

$*$ Proof: (귀류법) Halting 문제를 풀 수 있는 TM HALT가 존재한다고 가정하자.

Circuit model

Circuit model

- 무한히 긴 메모리(tape)를 가지는 이상적인 TM 대신, circuit model을 사용할 수도 있다.
- Circuit; 회로는 wire와 gate로 이루어진다.
    - wire: bit value가 전달되는 길
    - gate: 각 single/multi bit에 대해 연산을 수행하는 장치
- 각 gate는 오른쪽과 같은 symbol로 표현된다.



Theorem 3

Circuit model can solve every type of boolean function.

$$f : \{0,1\}^n \to \{0,1\}^m$$

$*$ Proof: (mathematical induction)

- 먼저 $n = m = 1$인 boolean function에 대해 생각해보자. 이 함수는 앞에서 언급한 gate들을 이용하여 쉽게 구현할 수 있다.
- (귀납적 가정) $m = 1$일 때, 어떤 n에 대해서도 boolean function을 구현할 수 있는 회로가 존재한다고 가정하자.
- 그럼 $n + 1$에 대한 boolean function은 $n + 1$번째 bit값이 0인지 1인지에 따라서 동작하는 서로다른 $n$bit function f_0, f_1을 사용하여 구현할 수 있다.$\square$

Computation power of Circuit model

Can circuit model solve halting problem?

Circuit model은 어떤 boolean function도 구현할 수 있기 때문에, *halting problem*을 푸는 *function*도 구현할 수 있다! 그러나 halting problem을 푸는 회로 h_n을 어떻게 설계해야하는지 알 수 없기 때문에 여전히 halting problem은 풀 수 없다.
$\Rightarrow h_n$ *is non-uniform circuit family.*

Definition 4 (circuit and circuit family)

A **circuit family** consists of a collection of circuits, $\{C_n\}$. The circuit C_n has n input bits and the output of the circuit C_n, upon input x $(len(x) \leq n)$ is denoted by $C_n(x)$. The function computed by the circuit family $\{C_n\}$ is the function $C(\cdot)$.

- uniform circuit family: $C(\cdot)$에 대해, 어떠한 입력 크기 n에 대해서도 회로 C_n의 구조를 출력하는 알고리즘이 존재하는 회로 집합.
- non-uniform circuit family: $C(\cdot)$에 대해, 어떠한 입력 크기 n에 대해서도 회로 C_n의 구조를 출력하는 알고리즘이 존재하지 않는 회로 집합.

References

- M. A. Nielson and I. L. Chuang, Quantum Computation and Quantum Information
- Lecture notes for QU511: Quantum Computing (Fall 2024)