

8.1 Turing Machine

Turing Machine(TM)은 1936년 Allan Turing에 의하여 발표된 이론적인 자동기계(automatic machine)이다. TM은 컴퓨터에 **이론적인 모델**로 알려져 있다. TM 이상의 컴퓨터 모델이 있는가의 의문은 당연히 생길 수 있는 질문인데, Church에 의하여 정의된 partial recursive function(prf)¹⁾과 TM이 동등함(equivalent)이 증명²⁾하였고, first-order logic, rewriting system등 많은 수학적 모델이 TM 혹은 prf와 동등하다는 증명이 있고, 그 이상의 수학적 모델도 개발되지 않아 TM을 계산할 수 있는 **최고의 모델**로 보는 것이 현대 전산학의 흐름이다.

TM으로 **계산할 수 있는**(recursively enumerable; RE, computable, programmable) 문제(problem)가 전산학 혹은 현대수학에서 다룰 수 있는 문제이다. TM으로도 **계산할 수 없는 문제**(problem)는 1901년에 이미 발표된 Russel의 모순(paradox)³⁾로 설명되는⁴⁾ halting problem 혹은 uncountable등은 수학이아 현대수학에서도 **계산할 수 없는 문제**(non-recursively enumerable; non-RE, nonprogrammable)로 알려져 있다.

TM으로 **계산할 수 있는** 문제는 **끝나는 문제**(recursive, decidable, terminate, algorithm)와 **끝나지 않는 문제**(nonrecursive but RE, undecidable, nonterminate)를 포함한다. **끝나지 않는 문제**는 컴퓨터 프로그램에서 **무한 반복(infinite loop)**으로 설명되며, partial recursive function에서는 **함수 값이 정의되지 않는** 경우이다.

Chomsky's Hierarchy

이미 배운 언어 클래스 중 정규(regular) 언어가 촘스키의 언어계급 중 가장 낮은 type 3로 정의하며, 문맥자유(context-free) 언어가 type 2로 recursive가 type 1으로 **recursively enumerable**이 type 0로 각각 정의된다. type 0가 아닌(non-RE) 경우까지를 고려하면 우리는 모두 5개의 언어(type 2,3) 혹은 문제(type 1, 0, -1) 클래스를 가지고 있다.

Type 3 정규언어를 표현하는 문법(grammar)는 정규(regular)문법으로, type 2는 문맥자유(context-free) 문법으로, type 1 혹은 type 언어는 제약이 없는⁵⁾ 일반적인 문법으로 표현 가능하다⁶⁾.

Type 3 정규언어를 해결하는 기계는 finite automata로, type 2인 문맥자유 언어는 fa에 기억장치로 stack을 추가한 pda로 type 1 혹은 그 이상 문제인 type 0는 TM으로 각각 대응된다. Type 1과 type 0에 구분은 끝나는(terminate)가 아닌가에 있는데, type 1을 위한 TM으로 tape의 크기에 **유한한 한계**를 두는 linear bounded automata(lba)를 정의하기도 한다.

1) μ (minimization)-recursive function이라고도 부른다.

2) Turing Church's Hypothesis(가설)라고 부른다.

3) Let $R = \{x \mid x \notin x\}$. Then $R \in R \Leftrightarrow R \notin R$. \therefore 모순.

4) Russel-like paradox

5) 문법 규칙의 좌변에 터미널이나 언 터미널 문자열(string)이 올 수 있다.

6) Non-RE은 물론 표현할 수 있는 문법이나 기계가 없다.