KAIST 전산학과 최광무

(정의) **문제**(problem) P
$P: N \rightarrow \{yes, no\}$을 문제로 정의한다.

(사실) **문제**의 개수는 **셀 수 없게**[1](uncountable) 많다.
(사실) **언어**(languages)의 개수도 **셀 수 없게**[2](uncountable) 많다.
(증명) Cantor's Diagonal Argument
infinite binary string, 자연수의 부분집합, Σ^*의 부분집합

(약속) 앞으로 **언어**와 **문제**는 구분하지 않는다.

(사실) TM 또는 프로그램은 **셀 수 있게**[3](countable) 많다.
(증명) TM 또는 프로그램은 Σ^*의 원소이므로, 셀 수 있게 많다.

(약속) 앞으로 **TM**과 **프로그램**은 구분하지 않는다.[4]

(약속) **셀 수 없게**(uncountable) 많음과 **셀 수 있게**(countable) 많음은 **다르므로 구분한다**.

(정의) **프로그램**[5]으로 풀 수 있는 문제[6]를 **recursively enumerable**(**type 0**) 이라 부른다.

(사실) 문제 중에는 **프로그램으로 풀 수 없는**(non-recursively enumerable) 문제가 있다.
(증명) 문제의 개수는 셀 수 **없게**(uncountable) 많고, 프로그램은 셀 수 **있게** 많다

(정의) TM 또는 프로그램이 풀 수 있는 문제 중에서 **항상 끝나는** 문제[7]를
decidable(**recursive**, terminate, algorithm, **type 1**)이라고 부른다.

(사실) Decidable하지 않은 Undecidable은 프로그램으로 **표현 가능하나 끝나지 않는**(RE but non recursive)와 프로그램으로 **표현 가능하지 않는**(non-RE) 두 가지 경우가 있다.

(**정리 9.3**) Decidable 한 문제 L의 complement $\overline{L}$도 decidable이다.
(증명) L을 위한 프로그램(TM)이 **항상 끝나므로**(recursive), yes로 끝나면 no를, no로 끝나면 yes를 보고하도록 바꾸면 이는 $\overline{L}$을 위한 **항상 끝나는**(recursive) 프로그램(TM)이다.

---

1) $f: A \rightarrow B$일 때, $|f| = |B|^{|A|}$이고, N이 자연수 일 때 $2^{|N|}$은 uncountable이다.
2) 언어 L은 $L \subseteq \Sigma^*$이다.
3) TM M이나 프로그램 P는 $M, P \in \Sigma^*$이다.
4) TM과 프로그램이 같음에 관한 증명은 1967년 Meyer와 Ritchie가 하였다.
5) 위의 약속에 의하여 TM으로 바꾸어도 같다.
6) Turing이 정의한 TM이 Church가 정의한 μ(minimization)-recursive **partial** function과 같음이 **증명**되어 있고, 이 **증명**에 근거한 가설을 Turing Church's Hypothesis(가설)라고 부른다.
7) **항상 끝나는** TM은 μ-recursive **total** function과 같음이 증명되어 있다.

(**정리 9.4**) 문제 L과 complement $\overline{L}$가 모두 Recursively enumerable하면, L과 $\overline{L}$ 모두 recursive하다.

(증명) 문제 L과 complement $\overline{L}$를 위한 프로그램 P와 $\overline{P}$를 동시에(in parallel) 돌리면서, P가 yes로 **끝나면** yes를, $\overline{P}$가 yes[8]로 **끝나면** no를 보고하면 문제 L을 위한 프로그램은 **항상 끝난다**(recursive). 문제 $\overline{L}$도 마찬가지이다.

(사실) 정리 9.3과 9.4의 결과를 정리하면, 문제와 문제의 complement의 경우는
i) 둘 다 recursive한 경우[9]
ii) 하나는 RE하지만 recursive하지 않고, 나머지 하나는 non-RE한 경우[10]
iii) 둘 다 non-RE인 경우[11]
세 가지 뿐이다.

<u>Universal and Diagonal Languages</u>

TM과 스트링들의 집합 Σ^*은 모두 **셀 수 있게** 많으므로(countable) 자연수로 나열(enumerate)할 수 있다.

TM: $M_0,\ M_1,\ M_2,\ \cdots,\ M_i,\ \cdots$

Σ^*: $x_0,\ x_1,\ x_2,\ \cdots,\ x_i,\ \cdots$

TM의 집합 L_u를 아래와 같이 정의하고, universal language라고 부른다.

$$L_u = \{M_i |\ x_i \in L(M_i)\}.$$

Universal language L_u의 complement인 L_d를 아래와 같이 정의하고, diagonalization language라고 부른다.

$$L_d = \overline{L_u} = \{M_i |\ x_i \notin L(M_i)\}.$$

Universal language L_u와 diagonalization language L_d의 **언어계급**(language hierarchy)을 살펴보자.

(정리 9.2) L_d는 recursively enumerable이 **아니다**.

(증명) L_d가 RE이라고 가정하자. L_d을 받아들이는 TM M이, $L(M) = L_d$, 존재한다.
$\therefore\ \exists i \in N,\ M = M_i$. 이 때 M_i에 대응되는 x_i에 대하여 생각해보자.
만일 $x_i \in L_d$라면 L_d의 정의에 따라서, $x_i \notin L_d$이어야 한다.
만일 $x_i \notin L_d$라면 L_d의 정의에 따라서, $x_i \in L_d$이어야 한다.
즉 $x_i \in L_d \Leftrightarrow x_i \notin L_d$이다.
따라서 L_d가 RE이라고 가정은 모순이다.

---

8) RE은, yes면 **항상 끝난다**.
9) 알고리즘에서 다루는 대부분에 문제들이 이 경우이다.
10) 교과서의 diagonalization language L_d가 RE가 아닌 경우이고, L_d의 complement $\overline{L_d}$와 같은 universal language L_u가 RE이나 recursive는 아닌 경우이고, 앞으로 배울 Rice Theorem에 의하면 TM과 관련된 모든 문제가 이 경우에 속한다.
11) 최광무교수는(만?) 이런 문제를 알지 못한다.^^

$\therefore$ L_d는 recursively enumerable이 **아니다**.

(정리 9.6) L_u는 recursively enumerable이지만 recursive는 아니다.

(증명) L_u가 RE이라고 가정하자. L_u을 받아들이는 universal TM을 U이라 하자.
U는 (M_i, x_i)를 차례로 만들어 내며, x_i가 만들어지면 M_i를 simulate 한다.
이 때 M_i가 x_i를 받아들이면[12], U도 M_i를 받아들인다.
$\therefore$ L_u는 recursively enumerable하다.

$L_d = \overline{L_u}$가 RE이 아니므로, L_d는 recursive가 아니다.

Reduction of one problem to another

문제 P_1: $D_1 \rightarrow \{0,1\}$과 문제 P_2: $D_2 \rightarrow \{0,1\}$에 대하여 아래 조건을 만족하는 함수 h: $D_1 \rightarrow D_2$가 존재하면

$\forall\, d_1 \in D_1$: $P_1(d_1) \Rightarrow P_2(h(d_2)) \ \wedge\ \neg P_1(d_1) \Rightarrow \neg P_2(h(d_2))$,

즉 $Y_{P_1} \subseteq Y_{P_2} \ \wedge\ N_{P_1} \subseteq N_{P_2}$[13],

문제 P_1을 문제 P_2으로 **줄인다**[14](reduce)라고 말하고, $P_1 \leq P_2$로 쓰고, 문제 P_1이 문제 P_2보다 **더 쉽거나 같거나**, 문제 P_2가 문제 P_1보다 **더 어렵거나 같다**.

예를 들어 어떤 문제 L을 universal language L_u로 줄이면($L_u \leq L$), 새로운 문제 L은 L_u보다 어렵거나 같으므로, 적어도 undecidable 이상이라고 할 수 있다. 이는 어떤 문제가 undecidable 이상임을 증명하는데 많이 쓰이는 방법이다.

12) $x_i \in L_u$.
13) $P: D \rightarrow \{0,1\}$일 때, $Y_P = \{d \in D \mid P(d) = 1\}$과 $N_P = \{d \in D \mid P(d) = 0\}$로 정의한다.
14) 문제는 줄었지만, 더 어려워졌다.