15. Uncomputability

- If a language or a problem is not Turing-decidable, it is said to be **unsolvable**.
- If Church's thesis is correct then there exist no algorithms to solve such a problem.
- In this lecture, we will see a few unsolvable problems.

Theorem 1 *The language*

$K_1 = \{\rho(M) : \textit{Turing machine } M \textit{ accepts input string } \rho(M)\}$

is not Turing-decidable.

Proof:

- If K_1 was Turing-decidable, by Theorem 2 of Lecture 12, $\overline{K}_1$ would also be Turing-decidable and hence Turing-acceptable.
- Let M' be a Turing machine that accepts $\overline{K}_1$.
- Does M' accepts $\rho(M')$?
- If yes, $\rho(M') \in K_1$. Hence M' accepts a string $\rho(M')$ outside $\overline{K}_1$. Contradiction.
- If no, $\rho(M') \in \overline{K}_1$. Hence M' does not accepts a string $\rho(M')$ in $\overline{K}_1$. Contradiction also.
- Hence K_1 is not Turing-acceptable. $\square$

Essence of proof:

M' accepts $\rho(M')$ iff it does not

Analogy: This sentence is wrong.

Theorem 2 *(***The halting problem***) The language*

$$K_0 = \{\rho(M)\rho(w) : \textit{Turing machine } M \textit{ accepts input string } w\}$$

is not Turing-decidable.

Notes:

- M accepts w iff it halts on w.
- The theorem says that there is no algorithm to determine, for arbitrary given Turing machine M and input w, whether M will eventually halts on w.

Proof:

- If K_0 was decided by a Turing machine M_0, then we could construct a machine M_1 that decides K_1 as follows.
- M_1 converts its input $\#w\#$ into $\#w\rho(w)\#$ and pass it to M_0. M_1 halts and returns Y or N iff M_0 does.
- M_0 halts and returns $\boxed{\text{Y}}$ on input $\#w\rho(w)\#$ if
  - $w = \rho(M)$ for some TM, and
  - M accepts $w = \rho(M)$.

  Otherwise, it halts and return $\boxed{\text{N}}$.

- Hence M_1 halts on input $\#w\#$ and returns $\boxed{\text{Y}}$ if
  - $w = \rho(M)$ for some TM, and
  - M accepts $w = \rho(M)$.

  Otherwise, it halts and return $\boxed{\text{N}}$.
- In other words, M_1 decides K_1. A contradiction.

Even though not being Turing-decidable, K_0 is Turing acceptable. Because

- For any string $\rho(M)\rho(w)$ in K_0, M accepts w, i.e. M hatls on input w.
- This is the same as that the universal Turing machine halts on input $\rho(M)\rho(w)$ (except for some minor technical caveats that can be easily fixed).

Theorem 3 *Not every Turing-acceptable language is Turing-decidable.*

- **Thue system**: A finite set of unordered pairs of strings.

  J: {ab, cd}, {cbd, bc}, {b, dab}, {da, abc}

- Given a Thue system, two strings are **equivalent** if one can be obtained from the other by successively replacing one string in a pair by the other.

  *ab*a $<=>_J$ c*da* $<=>_J$ ca*b*c $<=>_J$ cad*abc* $<=>_J$ cadda

- **Word problem for Thue systems**: Write a program that takes a Thue system and two strings and tell whether the two strings are equivalent.

Theorem 4 *The word problem for Thue system is unsolvable.*

Proof:

- A special version of the halting problem:

  Given a TM M and a string w:

  $$(s, \#w\underline{\#}) \vdash_M^* (h, \underline{\#})?$$

  Unsolvable.

- For any TM M, let G be the grammar in Theorem 1 of Lecture 14. Then

  $$(s, \#w\underline{\#}) \vdash_M^* (h, \underline{\#}) \text{ iff } [\#ws\#] \Rightarrow_G^* [h\#]$$

- Construct a Thue system

  $$J = \{\{u, v\} : u \to v \text{ is a rule in } G\}.$$

- One can prove (see textbook) that

  $$[\#ws\#] \Rightarrow_G^* [h\#] \text{ iff } [\#ws\#] <=>_J [h\#] \tag{1}$$

- Thus, if the word problem was solvable, then the special version of halting problem would be solvable. Therefore, the word problem is unsolvable. □