Lecture 25 – Undecidability

COSE215: Theory of Computation

Jihyeok Park

PLRG

2024 Spring

Recall

- A language $L(M)$ accepted by a TM M is **Recursively Enumerable**:

$$L(M) = \{w \in \Sigma^* \mid q_0\ w \vdash^* \alpha\ q_f\ \beta \nvdash \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

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

Recall

- A language $L(M)$ accepted by a TM M is **Recursively Enumerable**:

$$L(M) = \{w \in \Sigma^* \mid q_0\ w \vdash^* \alpha\ q_f\ \beta \not\vdash \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

  where $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.
- Let's learn another class of languages: **decidable languages (DLs)**.

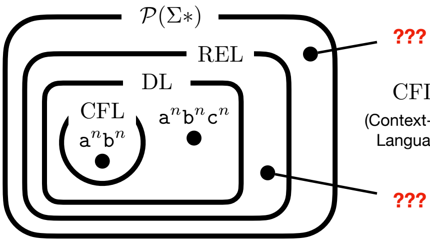
Recall

- A language $L(M)$ accepted by a TM M is **Recursively Enumerable**:

$$L(M) = \{w \in \Sigma^* \mid q_0\ w \vdash^* \alpha\ q_f\ \beta \nvdash \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

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

- Let's learn another class of languages: **decidable languages (DLs)**.



$\mathcal{P}(\Sigma*)$
REL
DL
CFL
$\mathtt{a}^n\mathtt{b}^n$
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$
???
???

$$\underset{\text{(Context-Free Language)}}{\text{CFL}} \subset \underset{\text{(Decidable Language)}}{\text{DL}} \subset \underset{\text{(Recursively Enumerable Language)}}{\text{REL}}$$

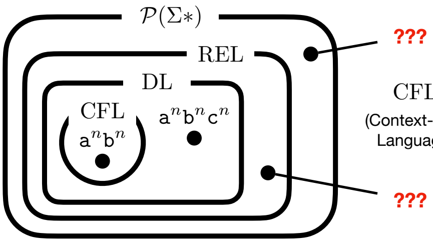
Recall

- A language $L(M)$ accepted by a TM M is **Recursively Enumerable**:

$$L(M) = \{w \in \Sigma^* \mid q_0\ w \vdash^* \alpha\ q_f\ \beta \nvdash \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

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

- Let's learn another class of languages: **decidable languages (DLs)**.



$$\underset{\text{(Context-Free Language)}}{\text{CFL}} \subset \underset{\text{(Decidable Language)}}{\text{DL}} \subset \underset{\text{(Recursively Enumerable Language)}}{\text{REL}}$$

- Is there a language that is **NOT** REL?

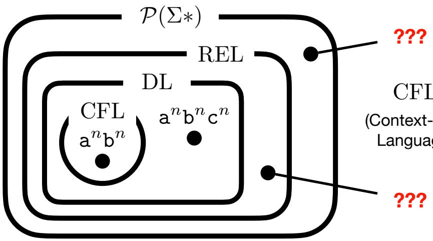
Recall

- A language $L(M)$ accepted by a TM M is **Recursively Enumerable**:

$$L(M) = \{w \in \Sigma^* \mid q_0\ w \vdash^* \alpha\ q_f\ \beta \nvdash \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

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

- Let's learn another class of languages: **decidable languages (DLs)**.



CFL (Context-Free Language) $\subset$ DL (Decidable Language) $\subset$ REL (Recursively Enumerable Language)

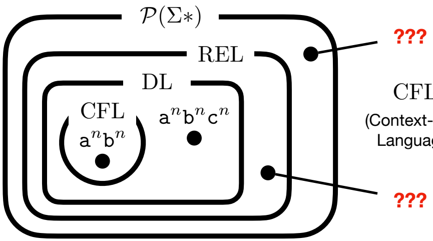
- Is there a language that is **NOT** REL? **Yes!**

Recall

- A language $L(M)$ accepted by a TM M is **Recursively Enumerable**:

$$L(M) = \{w \in \Sigma^* \mid q_0\ w \vdash^* \alpha\ q_f\ \beta \nvdash \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

  where $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.
- Let's learn another class of languages: **decidable languages (DLs)**.



$$\underset{\text{(Context-Free Language)}}{\text{CFL}} \subset \underset{\text{(Decidable Language)}}{\text{DL}} \subset \underset{\text{(Recursively Enumerable Language)}}{\text{REL}}$$

- Is there a language that is **NOT** REL? **Yes!**
- Is there a language that is REL but **NOT** decidable?

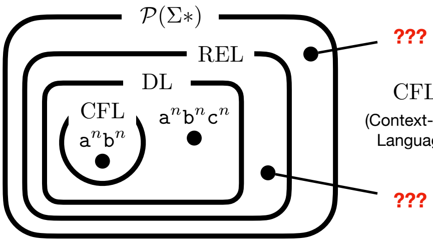
Recall

- A language $L(M)$ accepted by a TM M is **Recursively Enumerable**:

$$L(M) = \{w \in \Sigma^* \mid q_0\ w \vdash^* \alpha\ q_f\ \beta \nvdash \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

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

- Let's learn another class of languages: **decidable languages (DLs)**.



$$\text{CFL} \subset \text{DL} \subset \text{REL}$$

(Context-Free Language) (Decidable Language) (Recursively Enumerable Language)

- Is there a language that is **NOT** REL? **Yes!**
- Is there a language that is REL but **NOT** decidable? **Yes!**

Contents

Contents

Enumerating Binary Words

- We can define a **bijection** $f : \{0, 1\}^* \to \mathbb{N}$:

$$f(w) = (\text{the number represented by } 1w \text{ in binary})$$

Enumerating Binary Words

- We can define a **bijection** $f : \{0, 1\}^* \to \mathbb{N}$:

$$f(w) = (\text{the number represented by } 1w \text{ in binary})$$

- It means that the set of all binary words is **countably infinite**.

Enumerating Binary Words

- We can define a **bijection** $f : \{0,1\}^* \to \mathbb{N}$:

$$f(w) = (\text{the number represented by } 1w \text{ in binary})$$

- It means that the set of all binary words is **countably infinite**.
- And, we can enumerate them in w_i for $i \in \mathbb{N}$:

$$\begin{array}{rll} f(\epsilon) = 1 & (1 \text{ in binary}) & w_1 = \epsilon \\ f(0) = 2 & (10 \text{ in binary}) & w_2 = 0 \\ f(1) = 3 & (11 \text{ in binary}) & w_3 = 1 \\ f(00) = 4 & (100 \text{ in binary}) & w_4 = 00 \\ f(01) = 5 & (101 \text{ in binary}) & w_5 = 01 \\ f(10) = 6 & (110 \text{ in binary}) & w_6 = 10 \\ \vdots & & \vdots \end{array}$$

Enumerating Binary Words

- We can define a **bijection** $f : \{0, 1\}^* \to \mathbb{N}$:

$$f(w) = (\text{the number represented by } 1w \text{ in binary})$$

- It means that the set of all binary words is **countably infinite**.
- And, we can enumerate them in w_i for $i \in \mathbb{N}$:

$$\begin{array}{rlrl} f(\epsilon) = 1 & (1 \text{ in binary}) & w_1 & = \epsilon \\ f(0) = 2 & (10 \text{ in binary}) & w_2 & = 0 \\ f(1) = 3 & (11 \text{ in binary}) & w_3 & = 1 \\ f(00) = 4 & (100 \text{ in binary}) & w_4 & = 00 \\ f(01) = 5 & (101 \text{ in binary}) & w_5 & = 01 \\ f(10) = 6 & (110 \text{ in binary}) & w_6 & = 10 \\ \vdots & & \vdots & \end{array}$$

- We will use w_i to denote the i-th binary word.

Encoding TMs as Binary Words

$$M = (Q, \{0, 1\}, \Gamma, \delta, q_1, B, F)$$

where

- $Q = \{q_1, q_2, \cdots, q_r\}$
- $\Gamma = \{X_1, X_2, \cdots, X_s\}$
- Direction: $L = D_1$ and $R = D_2$

Encoding TMs as Binary Words

$$M = (Q, \{0, 1\}, \Gamma, \delta, q_1, B, F)$$

where

- $Q = \{q_1, q_2, \cdots, q_r\}$
- $\Gamma = \{X_1, X_2, \cdots, X_s\}$
- Direction: $L = D_1$ and $R = D_2$

We can encode a transition $\delta(q_i, X_j) = (q_k, X_l, D_m)$ as a binary word:

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Encoding TMs as Binary Words

$$M = (Q, \{0, 1\}, \Gamma, \delta, q_1, B, F)$$

where

- $Q = \{q_1, q_2, \cdots, q_r\}$
- $\Gamma = \{X_1, X_2, \cdots, X_s\}$
- Direction: $L = D_1$ and $R = D_2$

We can encode a transition $\delta(q_i, X_j) = (q_k, X_l, D_m)$ as a binary word:

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Then, we can encode a TM M as a binary word:

$$T_1 11 T_2 11 \cdots 11 T_n 111 0^{f_1} 1 0^{f_2} 1 \cdots 1 0^{f_t}$$

where T_i is the encoding of the i-th transition and $F = \{q_{f_1}, q_{f_2}, \cdots, q_{f_t}\}$.

Encoding TMs as Binary Words – Example

$$M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{X_1 = 0, X_2 = 1, X_3 = B\}, \delta, q_1, B, \{q_3\})$$

$$\begin{array}{lll}
\delta(q_1, 0) & = (q_1, 1, R) & \text{(encoded as 01010100100)} \\
\delta(q_1, 1) & = (q_1, 0, R) & \text{(encoded as 01001010100)} \\
\delta(q_1, B) & = (q_2, B, L) & \text{(encoded as 0100010010001 0)} \\
\delta(q_2, 0) & = (q_2, 0, L) & \text{(encoded as 00101001010)} \\
\delta(q_2, 1) & = (q_2, 1, L) & \text{(encoded as 001001001 0010)} \\
\delta(q_2, B) & = (q_3, B, R) & \text{(encoded as 0010001000100 0100)}
\end{array}$$

The encoding of M as a binary word is:

0101010010011010010101001101000100100010 11
00101001010110010010010010110010001000100010 0111
000

Enumerating TMs

Definition

We define M_i to be a TM encoded as the i-th binary word w_i.

Enumerating TMs

Definition

We define M_i to be a TM encoded as the i-th binary word w_i.

- However, not all binary words are valid encodings of TMs.

Enumerating TMs

Definition

We define M_i to be a TM encoded as the i-th binary word w_i.

- However, not all binary words are valid encodings of TMs.
- If w_i is not a valid encoding of a TM, we define M_i to be the TM that rejects all inputs.

Enumerating TMs

Definition

We define M_i to be a TM encoded as the i-th binary word w_i.

- However, not all binary words are valid encodings of TMs.
- If w_i is not a valid encoding of a TM, we define M_i to be the TM that rejects all inputs.
- For example, M_4 denotes a TM encoded as fourth binary word $w_4 = 00$. However, there is no TM encoded as 00. It means that M_4 is the TM that rejects all inputs (i.e., $L(M_4) = \varnothing$).

Diagonal Language L_d

Definition

The **diagonal language** $L_d = \{w_i \mid w_i \notin L(M_i)\}$

Diagonal Language L_d

Definition

The **diagonal language** $L_d = \{w_i \mid w_i \notin L(M_i)\}$

$$
\begin{array}{cc|ccccccc}
 & & \epsilon & 0 & 1 & 00 & 01 & 10 & \cdots \\
 & & w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & \cdots \\
\hline
\epsilon & M_1 & 1 & 1 & 0 & 1 & 0 & 1 & \cdots \\
0 & M_2 & 1 & 0 & 1 & 0 & 1 & 0 & \cdots \\
1 & M_3 & 1 & 1 & 1 & 0 & 0 & 1 & \cdots \\
00 & M_4 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
01 & M_5 & 1 & 1 & 1 & 1 & 0 & 1 & \cdots \\
10 & M_6 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots &
\end{array}
$$

where 1 and 0 denote **accept** and **reject**, respectively. Then, L_d is the language consisting of the words in the complement of the diagonal:

$$L_d = \{w_2, w_4, w_5, \cdots\}$$

L_d is Not Recursively Enumerable

Theorem

L_d is **NOT** *recursively enumerable.*

Proof) No TM can recognize L_d. Why?

L_d is Not Recursively Enumerable

Theorem

L_d is **NOT** *recursively enumerable.*

Proof) No TM can recognize L_d. Why?

Assume that the i-th TM M_i recognizes L_d. Then, there are two cases for w_i but both lead to a contradiction.

- If $w_i \in L_d$, then $w_i \notin L(M_i)$ by definition of L_d.
- If $w_i \notin L_d$, then $w_i \in L(M_i)$ by definition of L_d.

Contents

Decidable Languages (DLs)

Definition (Decidable Language (DL))

A language L is **decidable** if there is a TM M such that 1) $L(M) = L$ and 2) M halts on all inputs.

Decidable Languages (DLs)

Definition (Decidable Language (DL))

A language L is **decidable** if there is a TM M such that 1) $L(M) = L$ and 2) M halts on all inputs.

If L only satisfies 1), then L is **recursively enumerable**.

Decidable Languages (DLs)

Definition (Decidable Language (DL))

A language L is **decidable** if there is a TM M such that 1) $L(M) = L$ and 2) M halts on all inputs.

If L only satisfies 1), then L is **recursively enumerable**. In other words, a language L is recursively enumerable by a TM M if and only if

1. If $w \in L$, then M **halts** on w and **accepts** w with a **final state**.
2. If $w \notin L$, then there is two cases:
    1. M **halts** on w and **rejects** w with a **non-final** state.
    2. M **does not halt** on w.

Decidable Languages (DLs)

Definition (Decidable Language (DL))

A language L is **decidable** if there is a TM M such that 1) $L(M) = L$ and 2) M halts on all inputs.

If L only satisfies 1), then L is **recursively enumerable**. In other words, a language L is recursively enumerable by a TM M if and only if

1. If $w \in L$, then M **halts** on w and **accepts** w with a **final state**.
2. If $w \notin L$, then there is two cases:
    1. M **halts** on w and **rejects** w with a **non-final** state.
    2. M **does not halt** on w.

However, a **decidable language (DL)** L satisfies 2) as well.

Decidable Languages (DLs)

Definition (Decidable Language (DL))

A language L is **decidable** if there is a TM M such that 1) $L(M) = L$ and 2) M halts on all inputs.

If L only satisfies 1), then L is **recursively enumerable**. In other words, a language L is recursively enumerable by a TM M if and only if

1. If $w \in L$, then M **halts** on w and **accepts** w with a **final state**.
2. If $w \notin L$, then there is two cases:
    1. M **halts** on w and **rejects** w with a **non-final** state.
    2. M **does not halt** on w.

However, a **decidable language (DL)** L satisfies 2) as well. In other words, a language L is decidable by a TM M if and only if

1. If $w \in L$, then M **halts** on w and **accepts** w with a **final state**.
2. If $w \notin L$, then M **halts** on w and **rejects** w with a **non-final** state.

Closure Properties of DLs

Definition (Closure Properties)

The class of DLs is **closed** under an n-ary operator op if and only if $\text{op}(L_1, \cdots, L_n)$ is decidable for any DLs $L_1, \cdots, L_n$. We say that such properties are **closure properties** of DLs.

Closure Properties of DLs

Definition (Closure Properties)

The class of DLs is **closed** under an n-ary operator op if and only if $\text{op}(L_1, \cdots, L_n)$ is decidable for any DLs $L_1, \cdots, L_n$. We say that such properties are **closure properties** of DLs.

The class of DLs is closed under the following operations:

- Union
- Concatenation
- Kleene Star
- Intersection
- Complement (Let's focus on this property)

Closure Properties of DLs – Complement

Theorem (Closure under Complement)

If L is a decidable language, then so is $\overline{L}$.

Closure Properties of DLs – Complement

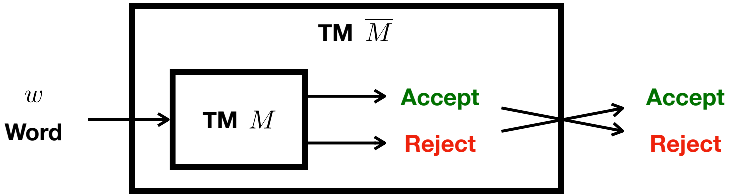
Theorem (Closure under Complement)

If L is a decidable language, then so is $\overline{L}$.

Proof) For a given DL L, we can always construct a TM M:

1. If $w \in L$, then M **halts** on w and **accepts** w with a **final state**.
2. If $w \notin L$, then M **halts** on w and **rejects** w with a **non-final** state.

Then, we can construct a TM $\overline{M}$ that simulates M and accepts w if M rejects w and vice versa by flipping the **final** and **non-final** states.



Contents

The Universal Language L_u

Definition

The language L_u is the set of all pairs (M, w) such that M accepts w:

$$L_u = \{(M, w) \mid w \in L(M)\}$$

where M is a TM and w is a binary word. In other words, L_u is the language accepted by the **universal Turing machine (UTM)**.

L_u is Recursively Enumerable but Not Decidable

Theorem

L_u *is* **recursively enumerable** *but* **NOT decidable**.

L_u is Recursively Enumerable but Not Decidable

Theorem

L_u is **recursively enumerable** *but* **NOT decidable***.*

Proof) We need to prove the following two statements:

1. **L_u is recursively enumerable.**

   Let's construct a TM M_u that accepts L_u.

2. **L_u is not decidable.**

   Let's prove by contradiction. Assume that L_u is decidable. Then, we will show that it is possible construct a TM M_d that accepts L_d. However, we already proved that L_d is not recursively enumerable. This is a contradiction.

L_u is Recursively Enumerable

It is enough to construct a (universal) TM M_u that accepts L_u:

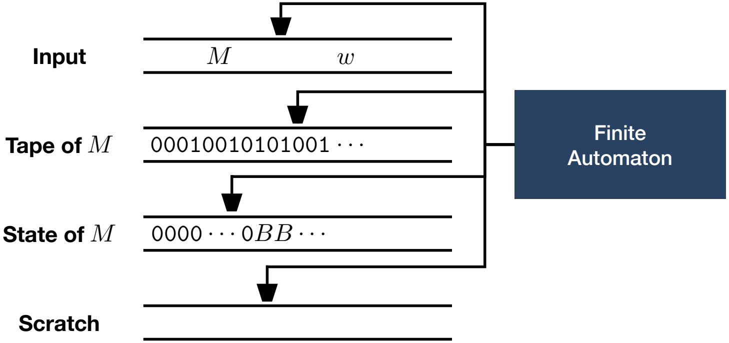
$$L_u = \{(M, w) \mid w \in L(M)\}$$

L_u is Recursively Enumerable

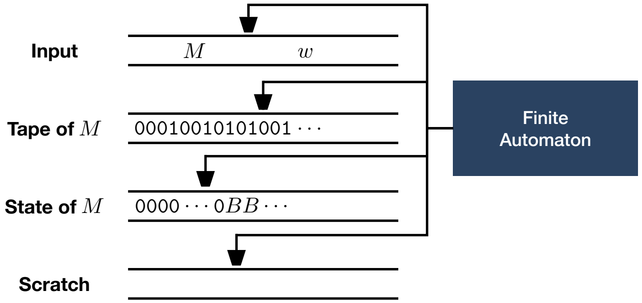
It is enough to construct a (universal) TM M_u that accepts L_u:

$$L_u = \{(M, w) \mid w \in L(M)\}$$

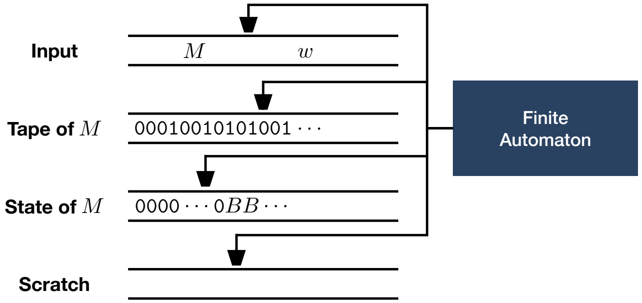
Idea) We can construct M_u that simulates M on w with **multiple tapes**:



L_u is Recursively Enumerable

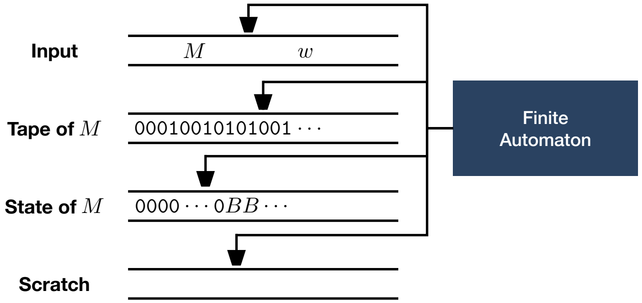


L_u is Recursively Enumerable



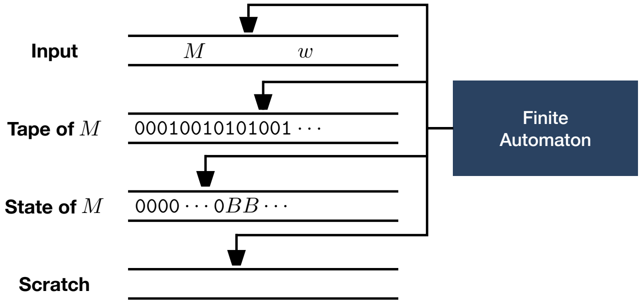
- The **1st** tape (**Input**) stores 1) the **encoding of** M and 2) the **input word** w in binary.

L_u is Recursively Enumerable



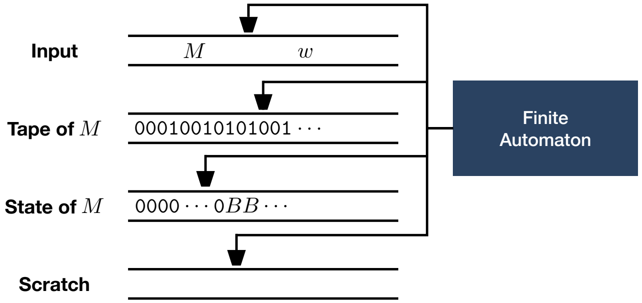
- The **1st** tape (**Input**) stores 1) the **encoding of** M and 2) the **input word** w in binary.
- The **2nd** tape (**Tape of** M) stores the **simulated tape** of M in binary. Each tape symbol X_i is encoded as 0^i , and separated by 1.

L_u is Recursively Enumerable



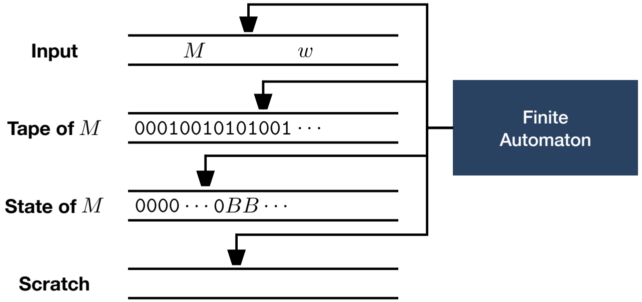
- The **1st** tape (**Input**) stores 1) the **encoding of** M and 2) the **input word** w in binary.
- The **2nd** tape (**Tape of** M) stores the **simulated tape** of M in binary. Each tape symbol X_i is encoded as 0^i , and separated by 1.
- The **3rd** tape (**State of** M) stores the **simulated state** of M in binary. The current state q_i is encoded as 0^i.

L_u is Recursively Enumerable



- The **1st** tape (**Input**) stores 1) the **encoding of** M and 2) the **input word** w in binary.
- The **2nd** tape (**Tape of** M) stores the **simulated tape** of M in binary. Each tape symbol X_i is encoded as 0^i , and separated by 1.
- The **3rd** tape (**State of** M) stores the **simulated state** of M in binary. The current state q_i is encoded as 0^i.
- The **4th** tape (**Scratch**) is used for the simulation.

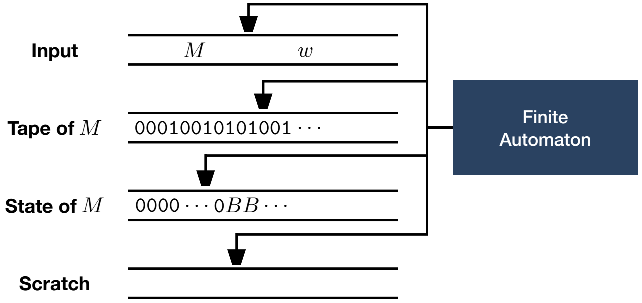
L_u is Recursively Enumerable



To simulate a move of M, M_u searches the corresponding transition in the 1st tape and updates the 2nd and 3rd tapes accordingly. For example,

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \text{ encoded as } 0^i10^j10^k10^l10^m \text{ in the 1st tape}$$

L_u is Recursively Enumerable



To simulate a move of M, M_u searches the corresponding transition in the 1st tape and updates the 2nd and 3rd tapes accordingly. For example,

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \text{ encoded as } 0^i10^j10^k10^l10^m \text{ in the 1st tape}$$

Then, M_u updates the 2nd and 3rd tapes as follows:

- The 2nd tape: Replace 0^j with 0^l, and Move the head according to m ($m = 0$ for left and $m = 1$ for right).
- The 3rd tape: Replace 0^i with 0^k.

L_u is Not Decidable

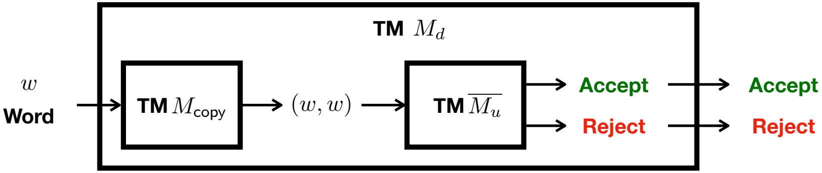
- Let's prove by contradiction. Assume that L_u is decidable.

L_u is Not Decidable

- Let's prove by contradiction. Assume that L_u is decidable.
- Then, the complement $\overline{L_u}$ of L_u is also decidable because DLs are **closed under complement**.

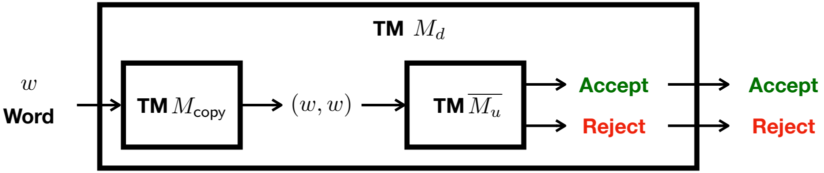
L_u is Not Decidable

- Let's prove by contradiction. Assume that L_u is decidable.
- Then, the complement $\overline{L_u}$ of L_u is also decidable because DLs are **closed under complement**.
- Consider another TM M_{copy} that **copies** the input word w to (w, w).
- Now, we can construct a TM M_d that accepts the diagonal language L_d using M_{copy} and $\overline{L_u}$ as follows (i.e., $L(M_d) = L_d$):



L_u is Not Decidable

- Let's prove by contradiction. Assume that L_u is decidable.
- Then, the complement $\overline{L_u}$ of L_u is also decidable because DLs are **closed under complement**.
- Consider another TM M_{copy} that **copies** the input word w to (w, w).
- Now, we can construct a TM M_d that accepts the diagonal language L_d using M_{copy} and $\overline{L_u}$ as follows (i.e., $L(M_d) = L_d$):



- However, we already proved that L_d is not recursively enumerable. This is a contradiction. Thus, L_u is **NOT** decidable. □

Contents

Decision Problems

Definition (Decision Problem)

A **decision problem** π is a computational problem whose answer is either **yes** or **no** for a given input.

Decision Problems

Definition (Decision Problem)

A **decision problem** π is a computational problem whose answer is either **yes** or **no** for a given input.

We say that a decision problem π is **decidable** (**solvable**) by a TM M if M halts on all inputs and $L(M) = \{w \mid \pi(w) = \text{yes}\}$.

Decision Problems

Definition (Decision Problem)

A **decision problem** π is a computational problem whose answer is either **yes** or **no** for a given input.

We say that a decision problem π is **decidable** (**solvable**) by a TM M if M halts on all inputs and $L(M) = \{w \mid \pi(w) = \text{yes}\}$.

If not, π is an **undecidable problem**. There are many examples:

- **Halting Problem** – Is there a TM that halts on a given input?
- **Equivalence of CFGs** – Are two CFGs equivalent?
- **Ambiguity of CFGs** – Is a CFG ambiguous?
- ...

Decision Problems

Definition (Decision Problem)

A **decision problem** π is a computational problem whose answer is either **yes** or **no** for a given input.

We say that a decision problem π is **decidable** (**solvable**) by a TM M if M halts on all inputs and $L(M) = \{w \mid \pi(w) = \text{yes}\}$.

If not, π is an **undecidable problem**. There are many examples:

- **Halting Problem** – Is there a TM that halts on a given input?
- **Equivalence of CFGs** – Are two CFGs equivalent?
- **Ambiguity of CFGs** – Is a CFG ambiguous?
- ...

If you are interested in more undecidable problems, please refer to:

https://en.wikipedia.org/wiki/List_of_undecidable_problems

Summary

- The **diagonal language** L_d:

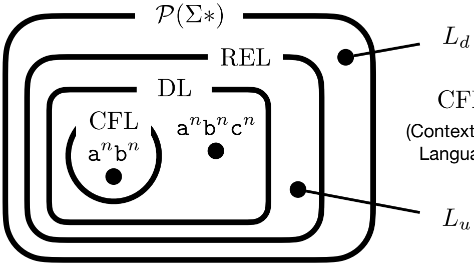
$$L_d = \{w_i \mid w_i \notin L(M_i)\}$$

where w_i is the i-th binary word and M_i is the i-th TM.

- The **universal language** L_u accepted by the **universal TM (UTM)**:

$$L_u = \{(M, w) \mid w \in L(M)\}$$

where M is a TM and w is a binary word.



$\mathcal{P}(\Sigma*)$
REL
DL
CFL
$\mathtt{a}^n\mathtt{b}^n$
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$
L_d
L_u

$$\text{CFL} \subset \text{DL} \subset \text{REL}$$

CFL (Context-Free Language) ⊂ DL (Decidable Language) ⊂ REL (Recursively Enumerable Language)

Final Exam

- The final exam will be given in class.
- **Date:** 13:30-14:45 (1 hour 15 minutes), June 19 (Wed.).
- **Location:** 604, Woojung Hall of Informatics (우정정보관 604호)
- **Coverage:** Lectures 14 – 26
- **Format:** 7–9 questions with closed book and closed notes
    - Filling blanks in some tables, sentences, or expressions.
    - Construction of automata or grammars for given languages.
    - Proofs of given statements related to automata or grammars.
    - Yes/No questions about concepts in the theory of computation.
    - etc.
- Note that there is **no class** on **June 17 (Mon.)**.
- Please refer to the **previous exams** in the course website:

https://plrg.korea.ac.kr/courses/cose215/

Next Lecture

- P, NP, and NP-Complete Problems

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr