Recursive languages

- A language \( L \subseteq \Sigma^* \) is recursively enumerable if \( L = L(M) \), for some Turing machine \( M \).

\[
\begin{align*}
w & \rightarrow M \\
& \begin{cases}
  \text{yes,} & \text{if } w \in L \\
  \text{no,} & \text{if } w \notin L \\
  \text{M does not halt,} & \text{if } w \notin L
\end{cases}
\end{align*}
\]

- A language \( L \subseteq \Sigma^* \) is recursive if \( L = L(M) \) for some Turing machine \( M \) that halts on all inputs \( w \in \Sigma^* \).

\[
\begin{align*}
w & \rightarrow M \\
& \begin{cases}
  \text{yes,} & \text{if } w \in L \\
  \text{no,} & \text{if } w \notin L
\end{cases}
\end{align*}
\]

- Lemma. \( L \) is recursive iff both \( L \) and \( \overline{L} = \Sigma^* \setminus L \) are recursively enumerable.

Enumerating languages

- An enumerator is a Turing machine \( M \) with extra output tape \( T \), where symbols, once written, are never changed.
- \( M \) writes to \( T \) words from \( \Sigma^* \), separated by $.$
- Let \( G(M) = \{ w \in \Sigma^* | w \text{ is written to } T \} \).

Some results

- Lemma. For any finite alphabet \( \Sigma \), there exists a Turing machine that generates the words \( w \in \Sigma^* \) in canonical ordering (i.e., \( w < w' \iff |w| < |w| \) or \( |w| = |w| \) and \( w \prec_{\text{lex}} w' \)).

- Lemma. There exists a Turing machine that generates all pairs of natural numbers (in binary encoding).

  Proof: Use the ordering \((0, 0), (1, 0), (0, 1), (2, 0), (1, 1), (0, 2), \ldots \).

- Proposition. \( L \) is recursively enumerable iff \( L = G(M) \), for some Turing machine \( M \).

Computing functions

- Unary encoding of natural numbers: \( i \in \mathbb{N} \mapsto | \ldots | = | i \)

  \( \text{(binary encoding would also be possible)} \)
- \( M \) computes \( f : \mathbb{N}^k \rightarrow \mathbb{N} \) with \( f(i_1, \ldots, i_k) = m \):
  - Start: \( |^k 0 |^k 0 \ldots |^k \)
  - End: \( |^m \)
- \( f \) partially recursive:
  \[
  i_1, \ldots, i_k \rightarrow M \rightarrow \begin{cases}
    \text{halts with } f(i_1, \ldots, i_k) = m, \\
    \text{does not halt, i.e., } f \text{ undefined.}
  \end{cases}
  \]
- \( f \) recursive:
  \[
  i_1, \ldots, i_k \rightarrow M \rightarrow \text{halts with } f(i_1, \ldots, i_k) = m.
  \]
Turing machines codes

- May assume
  \[ M = (Q, \{0, 1\}, \{0, 1, \#\}, \delta, q_1, \#, \{q_2\}) \]

- Unary encoding
  \[ 0 \mapsto 0, 1 \mapsto 00, \# \mapsto 000, L \mapsto 0, R \mapsto 00 \]

- \( \delta(q_i, X) = (q_j, Y, R) \) encoded by
  \[ 0^{i10_{10}^{i10}}10^{j10}0 \]

- \( \delta \) encoded by
  \[ 111 \text{ code}_1, 11 \text{ code}_2, 11 \ldots, 11 \text{ code}_r, 111 \]

- Encoding of Turing machine \( M \) denoted by \( \langle M \rangle \).

Numbering of Turing machines

- **Lemma.** There exists a Turing machine that generates the natural numbers in binary encoding.

- **Lemma.** There exists a Turing machine \( \text{Gen} \) that generates the binary encodings of all Turing machines.

- **Proposition.** The language of Turing machine codes is recursive.

- **Corollary.** There exist a bijection between the set of natural numbers, Turing machine codes and Turing machines.

\[ M \longrightarrow \langle M \rangle \longrightarrow \text{Equality test} \quad \text{+ counter} \longrightarrow \text{number n} \]

Diagonalization

- Let \( w_i \) be the \( i \)-th word in \( \{0, 1\}^* \) and \( M_j \) the \( j \)-th Turing machine.

- Table \( T \) with \( t_{ij} = \begin{cases} 1, & \text{if } w_i \in L(M_j) \\ 0, & \text{if } w_i \not\in L(M_j) \end{cases} \)

\[ \begin{array}{ccccccc} j \rightarrow & 1 & 2 & 3 & 4 & \ldots \\ \hline 1 & 0 & 1 & 1 & 0 & \ldots \\ i & 2 & 1 & 1 & 0 & 1 & \ldots \\ \downarrow & 3 & 0 & 0 & 1 & 0 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array} \]

- Diagonal language \( L_d = \{ w_i \in \{0, 1\}^* \mid w_i \not\in L(M_i) \} \).

- **Theorem.** \( L_d \) is not recursively enumerable.

- **Proof:** Suppose \( L_d = L(M_k) \), for some \( k \in \mathbb{N} \). Then
  \[ w_k \in L_d \iff w_k \not\in L(M_k), \]
  contradicting \( L_d = L(M_k) \).