Recursive languages

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

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

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

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

- 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 \( G(M) = \{ w \in \Sigma^* \mid w \text{ is written to } T \} \).

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: \( i \in \mathbb{N} \mapsto \underbrace{\| \ldots \|}_{i \text{ times}} = |i| \)
  
  (binary encoding would also be possible)

- \( M \) computes \( f : \mathbb{N}^k \rightarrow \mathbb{N} \) with \( f(i_1, \ldots, i_k) = m \):
  
  - Start: \( \underbrace{|i_1 \ 0| \ldots \ 0}_{|k|} \)
  
  - End: \( |m| \)

- \( f \) partially recursive:

\[
\begin{align*}
i_1, \ldots, i_k & \rightarrow M \\
\begin{cases}
halts \text{ with } f(i_1, \ldots, i_k) = m, \\
\text{does not halt, i.e., } f \text{ undefined.}
\end{cases}
\]

- \( f \) recursive:

\[
\begin{align*}
i_1, \ldots, i_k & \rightarrow M \\
\text{halts with } f(i_1, \ldots, i_k) = m.
\]
Turing machines codes

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

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

- \( \delta(q_i, X) = (q_j, Y, Z) \) encoded by
  \[ 010...01010...0 \]

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

- 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.** The language of Turing machine codes is recursive.
- **Proposition.** There exists a Turing machine \( \text{Gen} \) that generates the binary encodings of all Turing machines.
- **Theorem.** There exists a bijection between the set of natural numbers, Turing machine codes and Turing machines.

\[
\begin{align*}
\text{Gen} & \quad \rightarrow \quad \langle M \rangle & \quad \rightarrow & \quad \text{Equality test} \\
& \quad \rightarrow & \quad \text{number n} \\
\text{Gen} & \quad \rightarrow & \quad \text{Count} \\
\text{number n} & \quad \rightarrow & \quad \langle M \rangle & \quad \rightarrow & \quad M
\end{align*}
\]

Diagonalization

- Let \( w_i \) be the \( i \)-th word in \( \{0,1\}^* \) and \( M_j \) the \( j \)-th Turing machine.
- Table \( T \) with \( t_{ij} = \begin{cases} 1, & \text{if } w_i \in L(M_j) \\ 0, & \text{if } w_i \notin L(M_j) \end{cases} \)

\[
\begin{array}{cccccccc}
| i | & 1 & 2 & 3 & 4 & \ldots \\
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 &amp; \ldots</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 &amp; \ldots</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 &amp; \ldots</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\end{array}
\]

- **Diagonal language** \( L_d = \{ w_i \in \{0,1\}^* \mid w_i \notin L(M_i) \} \).
- **Theorem.** \( L_d \) is not recursively enumerable.
- **Proof:** Suppose \( L_d = L(M_k) \), for some \( k \in \mathbb{N} \). Then \( w_k \in L_d \iff w_k \notin L(M_k) \), contradicting \( L_d = L(M_k) \).
Universal language

- \( \langle M, w \rangle \): encoding \( \langle M \rangle \) of \( M \) concatenated with \( w \in \{0,1\}^* \).
- Universal language
  \[ L_u = \{ \langle M, w \rangle \mid M \text{ accepts } w \} \]
- **Theorem.** \( L_u \) is recursively enumerable.
- A Turing machine \( U \) accepting \( L_u \) is called universal Turing machine.
- **Theorem** (Turing 1936). \( L_u \) is not recursive.
  
  **Proof:** Assume \( L_u \) is recursive and show that this would imply \( \overline{L}_d \) (and thus \( L_d \)) is recursive.

Decision problems

- Decision problems are problems with answer either yes or no.
- Associate with a language \( L \subseteq \Sigma^* \) the decision problem \( D_L \)
  
  Input: \( w \in \Sigma^* \)

  Output: \( \{ \text{yes, if } w \in L \mid \text{no, if } w \notin L \} \)

  and vice versa.
- \( D_L \) is decidable (resp. semi-decidable) if \( L \) is recursive (resp. recursively enumerable).
- \( D_L \) is undecidable if \( L \) is not recursive.

Reductions

- A many-one reduction of \( L_1 \subseteq \Sigma_1^* \) to \( L_2 \subseteq \Sigma_2^* \) is a computable function \( f : \Sigma_1^* \to \Sigma_2^* \) with \( w \in L_1 \iff f(w) \in L_2 \).
- **Proposition.** If \( L_1 \) is many-one reducible to \( L_2 \), then
  1. \( L_1 \) is decidable if \( L_2 \) is decidable.
  2. \( L_2 \) is undecidable if \( L_1 \) is undecidable.

Post’s correspondence problem

- Given pairs of words
  \[ (v_1, w_1), (v_2, w_2), \ldots, (v_k, w_k) \]
  over an alphabet \( \Sigma \), does there exist a sequence of integers \( i_1, \ldots, i_m, m \geq 1 \), such that
  \[ v_{i_1} \ldots v_{i_m} = w_{i_1} \ldots w_{i_m} \]
  
  **Example**

<table>
<thead>
<tr>
<th>( i )</th>
<th>( v_i )</th>
<th>( w_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>111</td>
</tr>
<tr>
<td>2</td>
<td>1011</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

  \( \Rightarrow v_2 v_1 v_3 w_2 w_1 w_3 = 101111110 \)

- **Theorem** (Post 1946). Post’s correspondence problem is undecidable.