What is a computable function?

- Non-trivial question $\leadsto$ various formalizations, e.g.
  
  - General recursive functions \quad G\ddot{o}d\ddot{e}l/Herbrand/Kleene 1936
  - $\lambda$-calculus \quad Church 1936
  - $\mu$-recursive functions \quad G\ddot{o}d\ddot{e}l/Kleene 1936
  - Turing machines \quad Turing 1936
  - Post systems \quad Post 1943
  - Markov algorithms \quad Markov 1951
  - Unlimited register machines \quad Shepherdson-Sturgis 1963

- All these approaches have turned out to be equivalent.

Church-Turing thesis

The class of intuitively computable functions is equal to the class of Turing computable functions.

Turing machine

\[
\begin{array}{cccccccc}
  a_1 & a_2 & \ldots & a_i & \ldots & a_n & \# & \# & \ldots
\end{array}
\]

infinite tape

\[
\begin{array}{c}
  \text{finite control} \\
  q
\end{array}
\]

Depending on the symbol scanned and the state of the control, in each step the machine

- changes state,
- prints a symbol on the cell scanned, replacing what is written there,
- moves the head left or right one cell.

Formal definition

- \( M = (Q, \Sigma, \Gamma, \delta, q_0, \#, F) \)
- \( Q \) is the finite set of states.
- \( \Gamma \) is the finite alphabet of allowable tape symbols.
- \( \# \in \Gamma \) is the blank.
- \( \Sigma \subseteq \Gamma \setminus \{\#\} \) is the set of input symbols.
- \( \delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\} \) is the next move function (possibly undefined for some arguments)
- \( q_0 \in Q \) is the start state.
- \( F \subseteq Q \) is the set of final (accepting) states.
Recognizing languages

- **Instantaneous description**: $\alpha_l q \alpha_r$, where
  - $q$ is the current state,
  - $\alpha_l \alpha_r \in \Gamma^*$ is the string on the tape up to the rightmost nonblank symbol,
  - the head is scanning the leftmost symbol of $\alpha_r$.

- **Move**: $\alpha_l q \alpha_r \vdash \alpha'_l q' \alpha'_r$, by one step of the machine.

- **Language accepted**
  
  $L(M) = \{w \in \Sigma^* \mid q_0 w \vdash^* \alpha_l q \alpha_r, \text{ for some } q \in F \text{ and } \alpha_l, \alpha_r \in \Gamma^*\}$

- $M$ may not halt, if $w$ is not accepted.

**Example**

- **Turing machine**
  
  $M = (\{q_0, \ldots, q_4\}, \{0, 1\}, \{0, 1, X, Y, \#\}, \delta, q_0, \#, \{q_4\})$

  accepting the language $L = \{0^n 1^n \mid n \geq 1\}$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$0$</th>
<th>$1$</th>
<th>$X$</th>
<th>$Y$</th>
<th>$#$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$(q_1, X, R)$</td>
<td>–</td>
<td>–</td>
<td>$(q_3, Y, R)$</td>
<td>–</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$(q_1, 0, R)$</td>
<td>$(q_2, Y, L)$</td>
<td>–</td>
<td>$(q_1, Y, R)$</td>
<td>–</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$(q_2, 0, L)$</td>
<td>–</td>
<td>$(q_0, X, R)$</td>
<td>$(q_2, Y, L)$</td>
<td>–</td>
</tr>
<tr>
<td>$q_3$</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>$(q_3, Y, R)$</td>
<td>$(q_4, #, R)$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

- **Example computation**

  
  $q_0 0011 \vdash X q_1.011 \vdash X_0 q_1.11 \vdash X q_2.0 Y 1 \vdash$
  
  $q_2 X 0 Y 1 \vdash X q_0 0 Y 1 \vdash X X q_1 Y 1 \vdash X Y q_1 1 \vdash$

  $X X q_2 Y Y \vdash X q_2 X Y Y \vdash X X q_0 Y Y \vdash X Y q_3 Y \vdash$

  $X Y Y q_3 \vdash X Y Y \# q_4$
Recursive languages

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

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

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

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

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

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 $w <_{\text{lex}} w'$).

- **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: $i \in \mathbb{N} \mapsto || \cdots ||_i = |i|$

  (binary encoding would also be possible)

- $M$ computes $f : \mathbb{N}^k \rightarrow \mathbb{N}$ with $f(i_1, \ldots, i_k) = m$:
  
  - Start: $|i_1| 0| \cdots |i_k| 0 | \cdots |
  
  - 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
  \[
  0^i 10^{i-1} 0^i 10^{i-1} 0 \to X
  0^i 10^{i-1} 0^i 10^{i-1} 0 \to Y
  0^i 10^{i-1} 0^i 10^{i-1} 0 \to R
  \]

- \( \delta \) 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.

\[
\begin{array}{cccc}
M & \rightarrow & \langle M \rangle & \rightarrow & \text{Equality test} + \text{counter} & \rightarrow & \text{number } n \\
\end{array}
\]

Diagonalization

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

\[
\begin{array}{cccccc}
\quad & 1 & 2 & 3 & 4 & \ldots \\
1 & 0 & 1 & 1 & 0 & \ldots \\
2 & 1 & 1 & 0 & 1 & \ldots \\
3 & 0 & 0 & 1 & 0 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

- **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 \( L_d = L(M_k) \).