Computability and Complexity Theory

Computability and complexity

- **Computability theory**
  - What problems can be solved on a computer?
  - What is a computable function?
  - Decidable vs. undecidable problems

- **Complexity theory**
  - How much time and memory is needed to solve a problem?
  - Tractable vs. intractable problems

**What is a computable function?**

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

- 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**

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.
- \( \Sigma \subset \Gamma \setminus \{#\} \) is the set of input symbols.
- \( \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \) 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,
  - \( \alpha \in \Gamma^* \) is the string on the tape up to the rightmost nonblank symbol,
  - the head is scanning the leftmost symbol of \( \alpha_r \).
- Move: \( \alpha_i q \alpha_r \vdash \alpha'_i q'_r \), by one step of the machine.
- Language accepted
  \[
  L(M) = \{ w \in \Sigma^* | q_0 w \vdash^* \alpha_i q \alpha_r, \text{ for some } q \in F \text{ and } \alpha_i, \alpha_r \in \Gamma^* \}
  \]
- \( M \) 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 \( L = \{ 0^n 1^n | n \geq 1 \} \)

<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
  \[
  q_0011 \vdash Xq_1011 \vdash X0q_111 \vdash Xq_20Y1 \vdash \\
  q_2X0Y1 \vdash Xq_00Y1 \vdash XXq_1Y1 \vdash XXXq_1Y \vdash \\
  XXq_2YY \vdash Xq_2XYY \vdash XXq_0YY \vdash XXXq_3Y \vdash \\
  XXXYYq_3 \vdash XXXYYq_4
  \]