Formal Definition of Computation I

- $M$ accepts $w = w_1 \cdots w_n$ if $\exists$ states $r_0 \cdots r_n$ such that
  1. $r_0 = q_0$
  2. $\delta(r_i, w_{i+1}) = r_{i+1}, i = 0, \ldots, n - 1$
  3. $r_n \in F$

Definition: A language is regular if recognized by some automata
Given a language, how do we construct a machine to recognize it?

Basically we need to get a state diagram (where the number of states is finite)

An automaton recognizing \{0, 1\} strings with odd # of 1’s

Fig 1.20
Example

01

\[ q_e \xrightarrow{0} q_e \xrightarrow{1} q_o \]

010101

\[ q_e \xrightarrow{0} q_e \xrightarrow{1} q_o \xrightarrow{0} q_o \xrightarrow{1} q_e \xrightarrow{0} q_e \xrightarrow{1} q_o \]
Two ways to think about the design

After the first 1, we go to $q_o$. Subsequently, every 1, ..., 1 pair is cancelled out by

$$q_o \xrightarrow{1} q_e \xrightarrow{} \cdots \xrightarrow{} q_e \xrightarrow{1} q_o$$

$q_e, q_o$ respectively remember whether the number of 1’s so far is even or odd

Example 1.21

strings contain 001

Fig 1.22
\( q_0, q_{00} \) indicate that before the current input character, we have 0 and 00, respectively.
Regular Operations I

- Regular operations can be used to study whether languages are regular or not.
- That is, we aim to check if for a given language, where there are finite automata to recognize it or not.
- Three definitions:
  - $A, B$ are given languages
    - union
      \[ A \cup B \]
concatenation

\[ A \circ B = \{ xy \mid x \in A, y \in B \} \]

star:

\[ A^* = \{ x_1 \cdots x_k \mid k \geq 0, x_i \in A \} \]
If $k = 0$, what do we mean $x_1 \cdots x_k$?

We define

$\epsilon : \text{empty string}$

in this situation

Thus

$\epsilon \in A^*$
Example

\[ \Sigma = \{ a, \ldots, z \} \]
\[ A = \{ \text{good, bad} \} \]
\[ B = \{ \text{boy, girl} \} \]
\[ A \circ B = \{ \text{goodboy, \ldots} \} \]
\[ A^* : \{ \epsilon, \text{good, bad, goodgood, \ldots} \} \]

We say an operation \( R \) is **closed** if the following property holds

if \( x \in A, y \in A \), then \( xRy \in A \)
Example: $N = \{1, 2, \ldots\}$ is closed under multiplication

- Th 1.25: regular languages are closed under the union operation

$A_1, A_2$ are regular languages

$\Rightarrow A_1 \cup A_2$ is regular

- Proof
Assume we are given two automata

\[ M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \]
\[ M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \]
Construct a new machine

\[ M = (Q, \Sigma, \delta, q, F) \]
\[ Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\} \]
\[ \delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \]
\[ q_0 = (q_1, q_2) \]
\[ F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\} \]

Example: combining

\[ \{w \mid w \text{ has an odd } \# \text{ 1's}\} \cup \]
\[ \{w \mid w \text{ has an odd } \# \text{ 0's}\} \]
Regular Operations VIII

\[
\begin{array}{c}
\text{\(q_e\)}
\
\text{\(q_o\)}
\
\text{\(s_e\)}
\
\text{\(s_o\)}
\end{array}
\]
Is this proof rigourously enough?
A formal proof should be done by induction. But we don’t provide it here

- Th 1.26: closed under concatenation
  - If \( A, B \) are regular, then \( A \circ B \) is regular
  - But the proof is not easy
  - It’s unclear where to break the input

- To easily do the proof, we introduce a new technique called **nondeterminism**