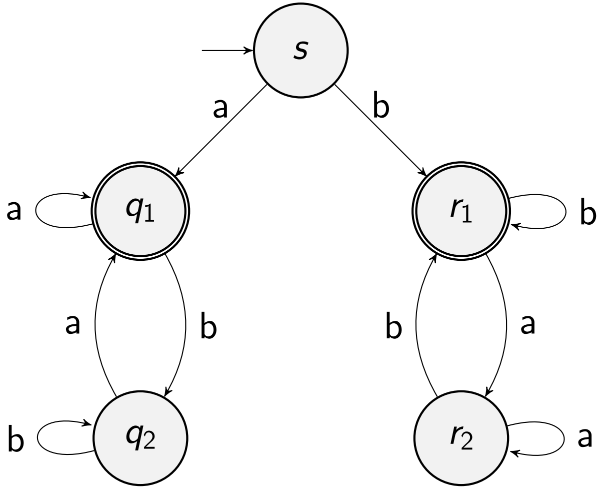
Example 1.11 I

- Fig 1.12

Example 1.11 II



Example 1.11 III

- $L(M) = ?$

$$a \dots a, b \dots b$$

  where "..." can be any string of a and b

- First we check that any string accepted by the machine must be

$$a \dots a, b \dots b$$

- Second we check that any

$$a \dots a, b \dots b$$

  can be recognized by the machine

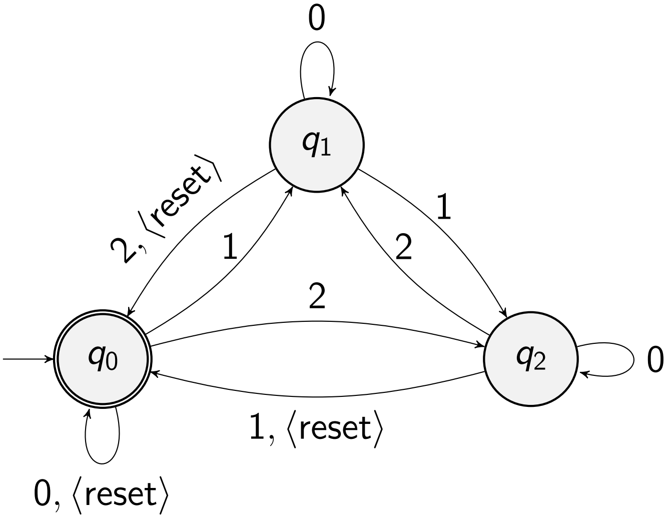
Example 1.11 IV

- This machine handles strings with the same character in the beginning and in the end

Example 1.13 I

- Figure 1.14

Example 1.13 II



Example 1.13 III

- $\Sigma = \{\langle \textit{reset} \rangle, 0, 1, 2\}$

$$
\begin{aligned}
L(M) &= \ldots\ldots \langle \textit{reset} \rangle \ldots \langle \textit{reset} \rangle \ldots \\
&= \{\text{sum of the last segment} \mod 3 = 0\}
\end{aligned}
$$

- Example:

$$10\langle \textit{reset} \rangle 22 \langle \textit{reset} \rangle 012$$

Example 1.13 IV

- An example of running a string

$$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{\langle reset \rangle} q_0 \xrightarrow{2} q_2 \xrightarrow{2} q_1$$
$$\xrightarrow{\langle reset \rangle} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{2} q_0$$

  Accepted
- Each node stores the sum of the current segment mod 3

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
- This is a very important definition
- Examples described earlier are regular languages
- We say some automata, so it's possible to have several automata for the same language

Formal Definition of Computation II

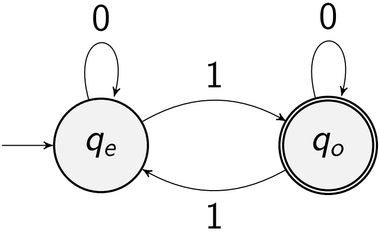
- As long as there is one, then the language is regular

Designing Automata I

- 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)
- Earlier we had the opposite: a machine is given and we check the corresponding language
- Example: an automaton recognizing $\{0, 1\}$ strings with an odd $\#$ of 1's

  Fig 1.20

Designing Automata II



- Sample strings

  01

$$q_e \xrightarrow{0} q_e \xrightarrow{1} q_o$$

  010101

$$q_e \xrightarrow{0} q_e \xrightarrow{1} q_o \xrightarrow{0} q_o \xrightarrow{1} q_e \xrightarrow{0} q_e \xrightarrow{1} q_o$$

Designing Automata III

- Two ways to think about the design
  - After the first 1, we go to q_o. Subsequently, every $1, \ldots, 1$ pair is cancelled out by

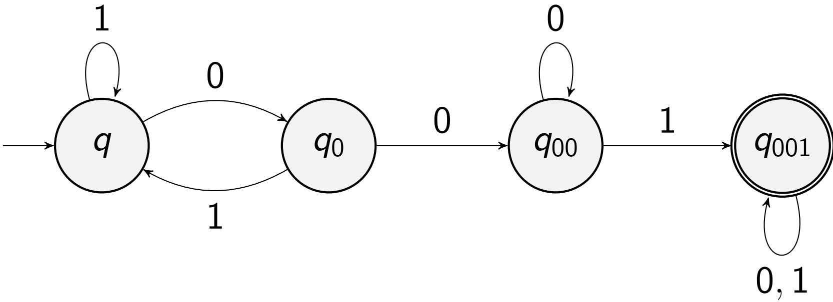
$$q_o \xrightarrow{1} q_e \rightarrow \cdots \rightarrow q_e \xrightarrow{1} q_o$$

  - q_e, q_o respectively remember whether the number of 1's so far is even or odd
- Example 1.21

  Language: strings containing 001

  Fig 1.22

Designing Automata IV



- q_0, q_{00} indicate that before the current input character, we have 0 and 00, respectively