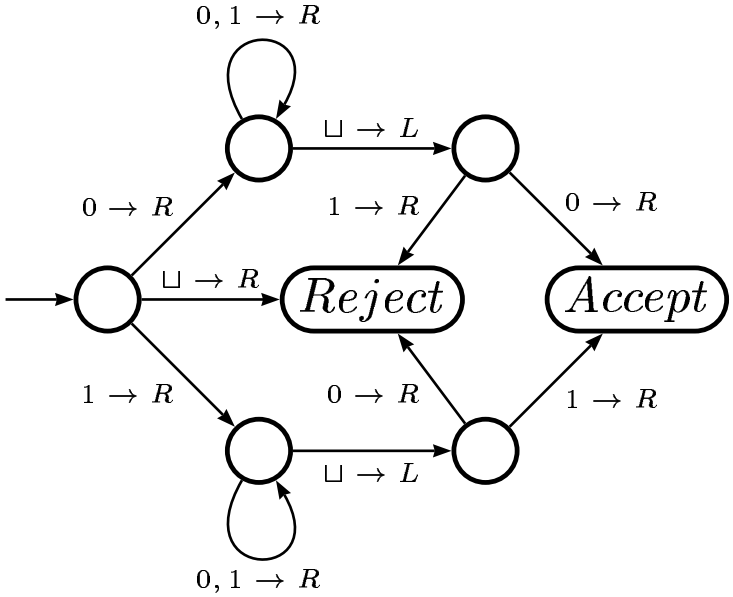
Turing Machine Example 1

This TM accepts all strings in the language

$$\{w \mid w \in \{0,1\}^+ \text{ and the first and last symbol of } w \text{ are the same}\}$$

and rejects all others. (Note that this is actually a regular language: $\Sigma \cup 0\Sigma^*0 \cup 1\Sigma^*1$, where $\Sigma = \{0,1\}$.)

State transition diagram:



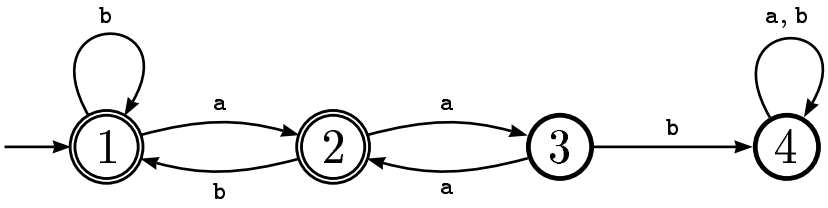
Informal implementation-level description:

$M =$ "On input string w:

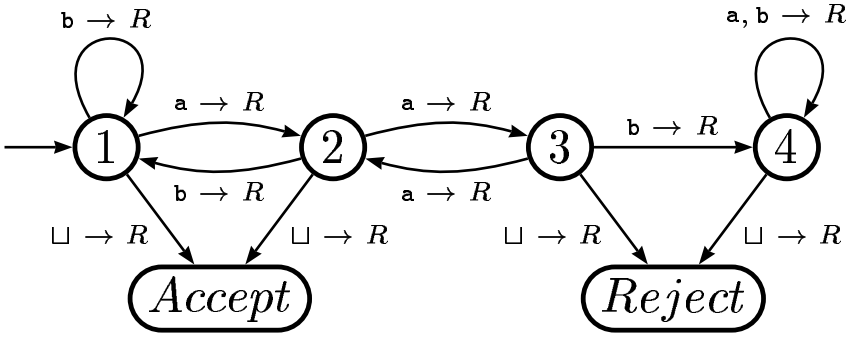
1. If the first symbol is blank, *reject*.
2. Record the first symbol in the finite control.
3. Sweep right to find the last (non-blank) symbol.
4. If it matches the stored first symbol, *accept*; if not, *reject*."

Turing Machine Example 2

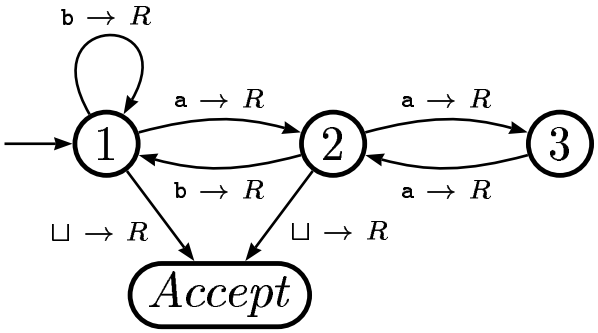
Any DFA can be converted into an equivalent TM that never writes to its tape and only moves its tape head to the right. For example, consider this DFA:



By connecting each accept state of the DFA to a new TM accept state and each non-accept state of the DFA to a new TM reject state, via $\sqcup \to R$ arrows, we get this equivalent TM:



Implicit Reject Convention: To simplify TM state transition diagrams, it is standard to remove the reject state and all arrows (and states) on paths that only lead to the reject state, with the understanding that any transition not accounted for in the diagram implicitly leads to the reject state. Using this convention, our TM can be simplified as follows:



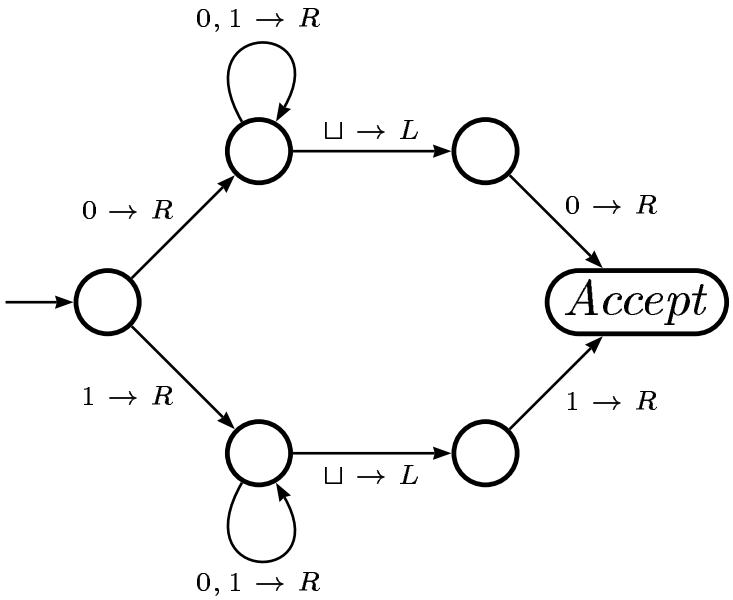
Turing Machine Example 1
(Simplified Using Implicit Reject Convention)

This TM accepts all strings in the language

$$\{w \mid w \in \{0,1\}^+ \text{ and the first and last symbol of } w \text{ are the same}\}$$

and rejects all others. (Note that this is actually a regular language: $\Sigma \cup 0\Sigma^*0 \cup 1\Sigma^*1$, where $\Sigma = \{0,1\}$.)

State transition diagram:



Informal implementation-level description:

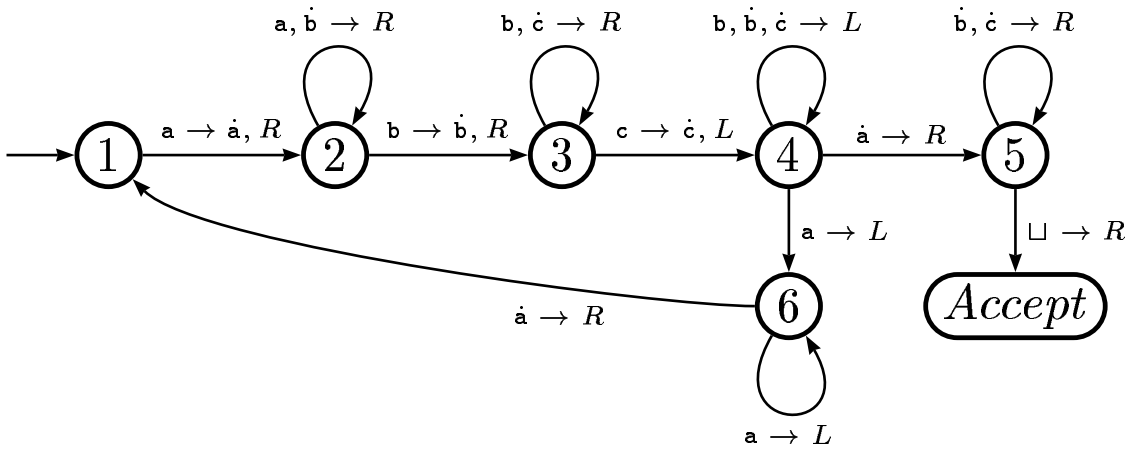
$M =$ "On input string w:

1. If the first symbol is blank, *reject*.
2. Record the first symbol in the finite control.
3. Sweep right to find the last (non-blank) symbol.
4. If it matches the stored first symbol, *accept*; if not, *reject*."

Turing Machine Example 3

This TM accepts all strings in the language $\{a^n b^n c^n \mid n \geq 1\}$ and rejects all others. (Recall that this language is not context-free.) Here we use the tape alphabet $\Gamma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \dot{\mathtt{a}}, \dot{\mathtt{b}}, \dot{\mathtt{c}}, \sqcup\}$.

State transition diagram:



Informal implementation-level description:

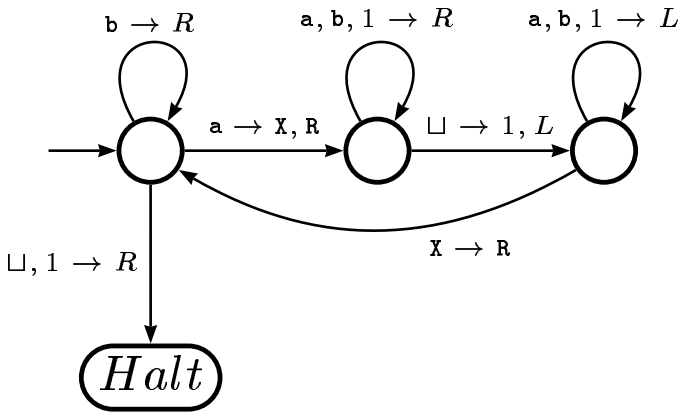
$M =$ "On input string w:

1. If the first symbol is a, mark it; if not, *reject*.
2. Sweep right looking for the first b. If one found, mark it; if not, *reject*.
3. Sweep right looking for the first c. If one found, mark it; if not, *reject*.
4. Sweep left looking for the rightmost marked a. If any a's encountered along the way, go to stage 1.
5. If no a's encountered during stage 4, sweep right to the end of the string.
   If no b's or c's found during this sweep, *accept*; else *reject*."

Turing Machine Example 4

Turing machines can also be used as *transducers*, producing output on their tape instead of making an accept/reject decision. Here is an example of one whose input is a string $w \in \{\mathtt{a}, \mathtt{b}\}^*$ and which produces a unary representation of the number of a's in the string. It writes this on the tape immediately following the rightmost symbol of the input. It uses the tape alphabet $\Gamma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{X}, \mathtt{1}, \sqcup\}$.

State transition diagram:



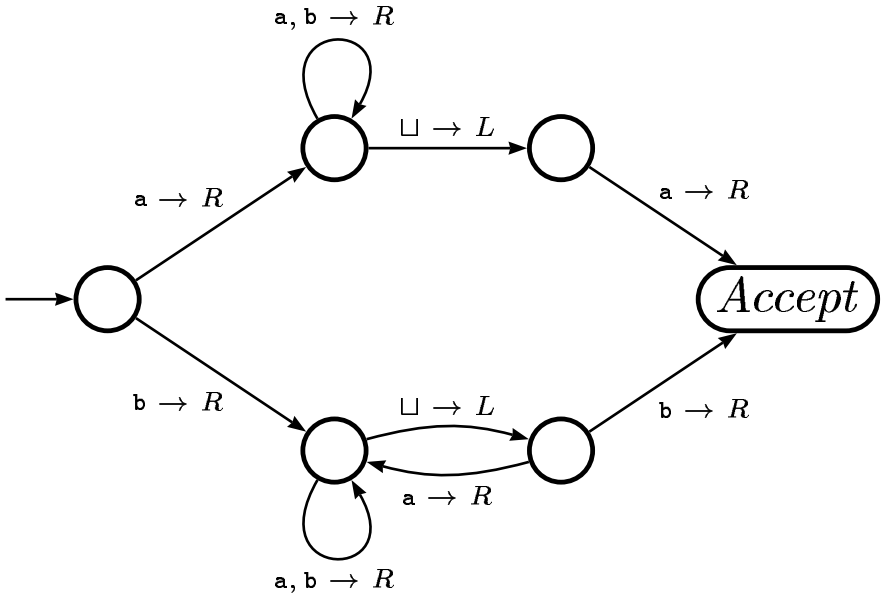
Informal implementation-level description:

$M =$ "On input string w:

1. Sweep right looking for the first (remaining) a.
2. If there are none, *halt*.
3. If an a is found, cross it out and sweep right to find the first blank cell.
4. Place a 1 in that first blank cell.
5. Sweep left, looking for the rightmost crossed-out cell.
6. Go to stage 1."

Turing Machine Example 5

Consider the following TM for strings over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$:



Accepts: any string of the form $\mathtt{a}\Sigma^*\mathtt{a} \cup \mathtt{b}\Sigma^*\mathtt{b} \cup \Sigma$

Rejects: any string of the form $\mathtt{a}\Sigma^*\mathtt{b} \cup \varepsilon$

Loops on: any string of the form $\mathtt{b}\Sigma^*\mathtt{a}$

Turing Machines: Recognizers vs. Deciders

A TM T is a *recognizer* for a language L if the set of strings it accepts is equal to L. The above TM is a recognizer for $\mathtt{a}\Sigma^*\mathtt{a} \cup \mathtt{b}\Sigma^*\mathtt{b} \cup \Sigma$.

A TM T is a *decider* for a language L if it accepts every string in L and rejects every string not belonging to L. The above TM is *not* a decider for $\mathtt{a}\Sigma^*\mathtt{a} \cup \mathtt{b}\Sigma^*\mathtt{b} \cup \Sigma$ because there are some strings it neither accepts nor rejects.

However, the TMs of Examples 1, 2, and 3 are all deciders. In fact, the TM of Example 1 is a decider for essentially this same language (with alphabet $\{0, 1\}$ instead of $\{\mathtt{a}, \mathtt{b}\}$).

Languages: Turing-Recognizable vs. Decidable

A language L is called *Turing-recognizable* if there is some TM that recognizes it (i.e., that accepts all strings belonging to L).

A language L is called *decidable* if there is some TM that decides it (i.e., that accepts all strings belonging to L and rejects all strings not belonging to L).