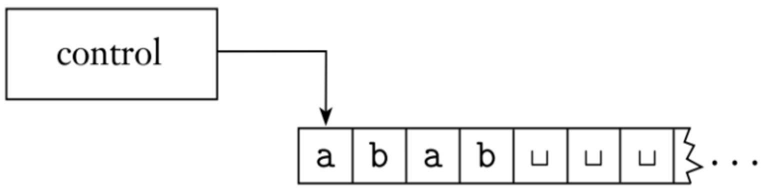
Turing Machines

Handnote Courtesy: Rafiad Sadat Shahir Sir

There exists a much more powerful model of computation, first proposed by Alan Turing in 1936, called the Turing machine. A Turing machine is a much more accurate model of a general purpose computer.

Description: A Turing machine consists of three parts:

1. A finite-state control that issues commands,
2. An infinite tape with input and blank cells,
3. A tape head that can read and write a single tape cell.



At each step, the Turing machine

1. Reads and writes a symbol to the tape cell under the tape head,
2. Changes state,
3. Moves the tape head to the left or to the right.

Difference with finite automata:

1. A Turing machine can both read from the tape and write on it.
2. The tape head can move both to the left and to the right.
3. The tape is infinite.
4. The special states for rejecting and accepting take effect immediately.

Turing recognizable and decidable: When we start a Turing machine on an input, three outcomes are possible: accept, reject, or loop. By loop it means that the machine simply does not halt. A Turing machine can fail to accept an input by rejecting, or by looping.

The language accepted by a Turing machine is called Turing recognizable or recursively enumerable language and the machine is called recognizer. Moreover, if the machine halts on all inputs, the language is called Turing decidable or recursive language and the machine is called decider.

Universal Turing machines: Turing proved that we can build a Turing machine (U) that acts as an interpreter for other Turing machines. In other words, U's input tape can contain a description of another Turing machine, which is then simulated step by step. Such a machine U is called a Universal Turing machine.

Closure of Turing recognizable and decidable languages: Turing recognizable and decidable Languages are closed under union, intersection, concatenation, and Kleene star operations. Though Turing decidable languages are closed under complement operation, Turing recognizable languages are not.

Proving a language decidable: One way to prove a language is decidable is to produce a decider. Let A_{DFA} be the language of all string representations of any DFA D and any string w, such that D accepts the string w. The decider for A_{DFA} decides whether a pair of strings $\langle D, w\rangle$ is in the language of A_{DFA} or not.

Example 4: $A_{DFA} \rightarrow \{\langle D, w\rangle \mid D \text{ is a DFA that accepts input string } w\}$. Prove that A_{DFA} is decidable.
Proof: We want to build a TM M that decides A_{DFA}:
M = "On input $\langle D, w\rangle$,
1. Run (or simulate) D on w.
2. If D ends in an accept state, accept; otherwise reject"

Since the simulation always ends after $|w|$ steps, M is a decider.
If $w \in L(A)$, then M will accept. If $w \notin L(A)$, then M will reject.

Church–Turing thesis: The Church–Turing thesis is a fundamental claim that is stated as: Any effectively calculable function can be computed by a Turing machine. Church used the λ-calculus to define algorithms whereas Turing did it with his Turing machines.