A Turing machine is a model of what a person can do mechanically, simply by following written instructions. It was introduced by Alan Turing in 1936. Nowadays it is considered a theoretical model of a computer. The components of a Turing machine are a tape, a reading head, and a program.

A tape and a reading head

- First cell
- Cells
- Tape has no end
- Machine state
- Reading head
- Each cell contains 0 or 1
- Eventually every cell contains 0

Definition of a Turing machine

A Turing machine is a finite set of quintuples (s, a, b, m, r), at most one for each s and a.

- If the machine is in state s, reading a, then it writes b, moves according to m and switches to state r.
- s and r are elements of a finite set called the set of states.
- a and b are elements of \{0, 1\}
- m is an element of \{L, R\}. “L” means “move to the left”. “R” means “move to the right”
### Input and output
- The **input** is on the tape in the beginning, the **output** in the end, provided the machine halts.
- If the machine has n arguments, and the input is a₁,...,aₙ, then the input tape is
  
  ![Input Tape Example](example_image)

### Running the machine
- In the beginning the reading head is at the beginning of the tape.
- Then the machine looks for an applicable quintuple in its program and proceeds according to the quintuple.

### Computable functions
- A function f of n arguments is said to be **computable** if there is a Turing machine which with a₁,...,aₙ as input always halts and its output is f (a₁,...,aₙ), that is, a sequence of f (a₁,...,aₙ) ones in the beginning of the tape, followed by a zero.
Which functions are computable

- The usual functions $+$ and $\cdot$ are computable.
- The family of computable functions is closed under composition and recursion.
- There are non-computable functions, but they are a bit difficult to define.

Church’s Thesis

- It has turned out that all sufficiently strong concepts of mechanical computability are equivalent.
- Church’s Thesis claims that indeed all mechanically computable functions are computable with a Turing machine.
- This is widely believed to be true, but cannot be formally proved because “mechanically computable” is an intuitive concept.

Mechanically decidable problems

- A k-ary predicate of numbers $n_1,\ldots,n_k$ is mechanically decidable if there is a Turing machine which with input $n_1,\ldots,n_k$ always halts and gives output 1 if the predicate holds and 0 otherwise.
- The predicates $n_1=n_2$, $n_1<n_2$, $n_1+n_2=n_3$, $n_1\cdot n_2=n_3$ are mechanically decidable.
The Halting Problem

- The **Halting Problem** is the problem to decide whether a given Turing machine with a given input halts or not.
- This is the simplest example of an **undecidable** problem, that is, of a problem that cannot be mechanically decided.

Problem: Give a Turing machine which always halts if the input is a sequence of ones in the beginning of the tape, and which gives a tape consisting of zeros only as the output.

Solution: \(M=\{(s_0,0,0,R,s_1), (s_0,1,0,R,s_0)\}\)

How this machine works:

- \(M=\{(s_0,0,0,R,s_1), (s_0,1,0,R,s_0)\}\)
- Assuming the input is a sequence of ones in the beginning of the tape, this machine replaces all the ones by zeros and then halts.
How this machine works:

- \( M=\{(s_0,0,0,R,s_1), (s_0,1,0,R,s_0)\} \)
- Assuming the input is a sequence of ones in the beginning of the tape, this machine replaces all the ones by zeros and then halts.

\[
\begin{array}{cccccccc}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

...
How this machine works:

- $M = \{(s_0, 0, 0, R, s_1), (s_0, 1, 0, R, s_0)\}$
- Assuming the input is a sequence of ones in the beginning of the tape, this machine replaces all the ones by zeros and then halts.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
```

Problem: Give a Turing machine which does not halt, whatever the input. (A looping machine)

- $M = \{(s_0, 0, 0, R, s_0), (s_0, 1, 1, R, s_0)\}$
- This machine moves right on the tape indefinitely. It is an example of a non-halting Turing machine.

```
1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
```
A looping machine

- **M** = \{(s₀, 0, 0, R, s₀), (s₀, 1, 1, R, s₀)\}
- This machine moves right on the tape indefinitely. It is an example of a **non-halting** Turing machine.

```
1 1 0 1 0 1 1 0 0 0 0 ...
```

This machine moves right on the tape indefinitely. It is an example of a **non-halting** Turing machine.
A looping machine

- $M = \{(s_0, 0, 0, R, s_0), (s_0, 1, 1, R, s_0)\}$
- This machine moves right on the tape indefinitely. It is an example of a non-halting Turing machine.

...
A looping machine

- $M = \{(s_0, 0, 0, R, s_0), (s_0, 1, 1, R, s_0)\}$
- This machine moves right on the tape indefinitely. It is an example of a non-halting Turing machine.

1 1 0 1 0 1 1 1 0 0 0 0 ...
A looping machine

- $M=\{(s_0,0,0,R,s_0), (s_0,1,1,R,s_0)\}$
- This machine moves right on the tape indefinitely. It is an example of a non-halting Turing machine.

This machine moves right on the tape indefinitely. It is an example of a non-halting Turing machine.

...
A looping machine

- $M = \{(s_0,0,0,R,s_0), (s_0,1,1,R,s_0)\}$
- This machine moves right on the tape indefinitely. It is an example of a non-halting Turing machine.

```
1 1 0 1 0 1 1 0 0 0 0 ...
```

Problem: Show that the function $f(a) = a + 1$ is computable.

Solution: $M = \{(s_0,0,1,R,s_1), (s_0,1,1,R,s_0)\}$

```
1 1 1 1 0 0 0 0 0 0 0 ...
```

How this machine works:

- $M = \{(s_0,0,1,R,s_1), (s_0,1,1,R,s_0)\}$
- Assuming the input is a sequence of ones in the beginning of the tape, this machine adds a one to the end.

```
1 1 1 1 0 0 0 0 0 0 0 ...
```
How this machine works:

- \( M = \{(s_0, 0, 1, R, s_1), (s_0, 1, 1, R, s_0)\} \)
- Assuming the input is a sequence of ones in the beginning of the tape, this machine adds a one to the end.

\[ 1 1 1 1 0 0 0 0 0 0 \ldots \]

\[ A \]
How this machine works:

- $M = \{ (s_0, 0, 1, R, s_1), (s_0, 1, 1, R, s_0) \}$
- Assuming the input is a sequence of ones in the beginning of the tape, this machine adds a one to the end.

```
1 1 1 1 0 0 0 0 0 0 0 ...
```

The Halting Problem

- The **Halting Problem** is the problem to decide whether a given Turing machine with a given input halts or not.
- This is the simplest example of an **undecidable** problem, that is, of a problem that cannot be mechanically decided.
Problem: Prove the undecidability of the Halting Problem

- Let $M_0, M_1, \ldots$ be a list of all Turing machines.
- Suppose that there is a Turing machine $M'$ that decides whether machine $M_n$ halts on input $m$ or not. We derive a contradiction.
- From $M'$ one easily constructs a Turing machine $M''$ that takes a number $n$ as input and halts if $M_n$ on input $n$ does not halt, and otherwise $M''$ does not halt.
- Let this machine by $M_k$.
- Now $M''$ on input $k$ halts if and only if it does not halt. A contradiction!