Turing machines
Algorithms, informally

No precise definition of “algorithm” at the time Hilbert posed the *Entscheidungsproblem*, just examples.

Common features of the examples:

- finite description of the procedure in terms of elementary operations
- deterministic (next step uniquely determined if there is one)
- procedure may not terminate on some input data, but we can recognize when it does terminate and what the result is.

*E.g.* multiply two decimal digits by looking up their product in a table.
Register Machine computation abstracts away from any particular, concrete representation of numbers (e.g. as bit strings) and the associated elementary operations of increment/decrement/zero-test.

Turing’s original model of computation (now called a Turing machine) is more concrete: even numbers have to be represented in terms of a fixed finite alphabet of symbols and increment/decrement/zero-test programmed in terms of more elementary symbol-manipulating operations.

TM control structure is also more elementary than for RMS
Turing machines, informally

- The machine is in one of a finite set of states.
- Linear tape, unbounded to the right, divided into cells containing a symbol from a finite alphabet of tape symbols. Only finitely many cells contain non-blank symbols.
Turing machines, informally

- The machine is in one of a finite set of states.
- A special blank symbol is present.
- A linear tape, unbounded to the right, divided into cells containing a symbol from a finite alphabet of tape symbols. Only finitely many cells contain non-blank symbols.
Turing machines, informally

The machine is in one of a finite set of states.

A tape symbol is being scanned by the tape head.

The tape is a linear tape, unbounded to the right, divided into cells containing a symbol from a finite alphabet of tape symbols. Only finitely many cells contain non-blank symbols.
Turing machines, informally

- Machine starts with tape head pointing to the special left endmarker ▶.
Turing machines, informally

- Machine starts with tape head pointing to the special left endmarker $\triangleright$.
- Machine computes in discrete steps, each of which depends only on current state ($q$) and symbol being scanned by tape head ($0$).
Turing machines, informally

- Machine starts with tape head pointing to the special left endmarker $\triangleright$.
- Machine computes in discrete steps, each of which depends only on current state ($q$) and symbol being scanned by tape head (0).
- Action at each step is to overwrite the current tape cell with a symbol, move left or right one cell, or stay stationary, and change state.
Turing Machines

are specified by:

- $Q$, finite set of machine states
- $\Sigma$, finite set of tape symbols (disjoint from $Q$) containing distinguished symbols $\rhd$ (left endmarker) and $\sqcup$ (blank)
- $s \in Q$, an initial state
- $\delta \in (Q \times \Sigma) \rightarrow (Q \cup \{\text{acc, rej}\}) \times \Sigma \times \{L, R, S\}$, a transition function—specifies for each state and symbol a next state (or accept $\text{acc}$ or reject $\text{rej}$), a symbol to overwrite the current symbol, and a direction for the tape head to move ($L$=left, $R$=right, $S$=stationary).
Turing Machines

are specified by:

- $Q$, finite set of machine states
- $\Sigma$, finite set of tape symbols (disjoint from $Q$) containing distinguished symbols $\triangleright$ (left endmarker) and $\square$ (blank)
- $s \in Q$, an initial state
- $\delta \in (Q \times \Sigma) \rightarrow (Q \cup \{ \text{acc, rej} \}) \times \Sigma \times \{L, R, S\}$, a transition function, satisfying:

  for all $q \in Q$, there exists $q' \in Q \cup \{ \text{acc, rej} \}$ with $\delta(q, \triangleright) = (q', \triangleright, R)$
  (i.e. left endmarker is never overwritten and machine always moves to the right when scanning it)
Example Turing Machine

\[ M = (Q, \Sigma, s, \delta) \]  where

states \( Q = \{s, q, q'\} \) (\( s \) initial)

symbols \( \Sigma = \{\triangleright, \downarrow, 0, 1\} \)

transition function

\[ \delta \in (Q \times \Sigma) \rightarrow (Q \cup \{\text{acc, rej}\}) \times \Sigma \times \{L, R, S\}: \]

<table>
<thead>
<tr>
<th></th>
<th>\triangleright</th>
<th>\downarrow</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>(s, \triangleright, R)</td>
<td>(q, \downarrow, R)</td>
<td>(rej, 0, S)</td>
<td>(rej, 1, S)</td>
</tr>
<tr>
<td>q</td>
<td>(rej, \triangleright, R)</td>
<td>(q', 0, L)</td>
<td>(q, 1, R)</td>
<td>(q, 0, R)</td>
</tr>
<tr>
<td>q'</td>
<td>(rej, \triangleright, R)</td>
<td>(acc, \downarrow, S)</td>
<td>(rej, 0, S)</td>
<td>(q', 1, L)</td>
</tr>
</tbody>
</table>
Turing machine configuration: \((q, w, u)\)

where

- \(q \in Q \cup \{\text{acc, rej}\} = \) current state
- \(w = \) non-empty string \((w = va)\) of tape symbols under and to the left of tape head, whose last element \((a)\) is contents of cell under tape head
- \(u = (\) possibly empty\) string of tape symbols to the right of tape head (up to some point beyond which all symbols are \(\_\))

\((So \; wu \in \Sigma^* \; represents \; the \; current \; tape \; contents.)\)

\((\text{not uniquely})\)
Turing machine configuration: \((q, w, u)\)

where

- \(q \in Q \cup \{\text{acc, rej}\} = \) current state
- \(w = \) non-empty string \((w = va)\) of tape symbols under and to the left of tape head, whose last element \((a)\) is contents of cell under tape head
- \(u = \) (possibly empty) string of tape symbols to the right of tape head (up to some point beyond which all symbols are \(\_\))

Initial configurations: \((s, \triangleright, u)\)
Turing machine computation

Given a TM $M = (Q, \Sigma, s, \delta)$, we write

$$(q, w, u) \rightarrow_M (q', w', u')$$

to mean $q \neq \text{acc, rej}$, $w = va$ (for some $v, a$) and

either $\delta(q, a) = (q', a', L)$, $w' = v$, and $u' = a'u$

or $\delta(q, a) = (q', a', S)$, $w' = va'$ and $u' = u$

or $\delta(q, a) = (q', a', R)$, $u = a''u''$ is non-empty, $w' = va'a''$ and $u' = u''$

or $\delta(q, a) = (q', a', R)$, $u = \varepsilon$ is empty, $w' = va' \sqcup$ and $u' = \varepsilon$. 
\( \delta(q, a) = (q', a', L) \)

\[
\begin{array}{c}
q \\
\downarrow
\end{array}
\quad \sim \\
\begin{array}{c}
q'
\end{array}
\]

\[
\begin{array}{c|c|c}
v & a & u \\
\hline

\begin{array}{c|c|c}
v & a' & u \\
\hline
\end{array}
\]

\[
(q, va, u) \quad \rightarrow^m \quad (q', v, a'u)
\]
\[ \delta(q, a) = (q', a', S) \]

\[ (q, va, u) \xrightarrow{M} (q', va', u) \]
\[ \delta(q, a) = (q', a', R) \]

\[ (q, v, a, u) \xrightarrow{M} (q', ?, ?, ?) \]
\[ \delta(q, a) = (q', a', R) \]

\[ (q, va, u) \quad \rightarrow \quad_M \quad (q', ?, ?) \]

**Two cases:**

\[ \begin{cases} u = \alpha'' \ u'' & \text{is non-empty} \\ \u = \varepsilon & \text{is empty} \end{cases} \]
\[ \delta(q, a) = (q', a', R) \]

\[ (q, va, a''u') \xrightarrow{\mathcal{M}} (q', va'a'', u) \]

Two cases:
\[ \left\{ \begin{array}{l}
    u = a''u'' \text{ is non-empty} \\
    u = \varepsilon \text{ is empty}
\end{array} \right. \]
\[ \delta(q, a) = (q', a', R) \]

\[ (q', va, \varepsilon) \rightarrow_{\text{M}} (q', va', \varepsilon) \]

Two cases:
\[ \begin{cases} 
  u = a'' u'' & \text{is non-empty} \\
  u = \varepsilon & \text{is empty}
\end{cases} \]
Turing machine computation

A computation of a TM $M$ is a (finite or infinite) sequence of configurations $c_0, c_1, c_2, \ldots$

where

- $c_0 = (s, \triangleright, u)$ is an initial configuration
- $c_i \xrightarrow{M} c_{i+1}$ holds for each $i = 0, 1, \ldots$

The computation

- does not halt if the sequence is infinite
- halts if the sequence is finite and its last element is of the form $(\text{acc}, w, u)$ or $(\text{rej}, w, u)$. 
Example Turing Machine

\( M = (Q, \Sigma, s, \delta) \) where

states \( Q = \{s, q, q'\} \) (\( s \) initial)

symbols \( \Sigma = \{\triangleright, \sqsubseteq, 0, 1\} \)

transition function

\( \delta \in (Q \times \Sigma) \rightarrow (Q \cup \{\text{acc, rej}\}) \times \Sigma \times \{L, R, S\} : \)

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>( \triangleright )</th>
<th>( \sqsubseteq )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s )</td>
<td>(( s, \triangleright, R ))</td>
<td>(( q, \sqsubseteq, R ))</td>
<td>(( \text{rej}, 0, S ))</td>
<td>(( \text{rej}, 1, S ))</td>
</tr>
<tr>
<td>( q )</td>
<td>(( \text{rej}, \triangleright, R ))</td>
<td>(( q', 0, L ))</td>
<td>(( q, 1, R ))</td>
<td>(( q, 1, R ))</td>
</tr>
<tr>
<td>( q' )</td>
<td>(( \text{rej}, \triangleright, R ))</td>
<td>(( \text{acc, } \sqsubseteq, S ))</td>
<td>(( \text{rej}, 0, S ))</td>
<td>(( q', 1, L ))</td>
</tr>
</tbody>
</table>

**Claim:** the computation of \( M \) starting from configuration \((s, \triangleright, \sqsubseteq 1^n 0)\) halts in configuration \((\text{acc, } \triangleright, \sqsubseteq 1^{n+1} 0)\).
Example Turing Machine

\[ M = (Q, \Sigma, s, \delta) \] where

states \( Q = \{ s, q, q' \} \) \( (s \text{ initial}) \)

symbols \( \Sigma = \{ \triangleright, \sqsubset, 0, 1 \} \)

transition function

\[ \delta \in (Q \times \Sigma) \rightarrow (Q \cup \{ \text{acc, rej} \}) \times \Sigma \times \{ L, R, S \} : \]

\[
\begin{array}{c|cccc}
\delta & \triangleright & \sqsubset & 0 & 1 \\
\hline
s & (s, \triangleright, R) & (q, \sqsubset, R) & (\text{rej}, 0, S) & (\text{rej}, 1, S) \\
q & (\text{rej}, \triangleright, R) & (q', 0, L) & (q, 1, R) & (q, 1, R) \\
q' & (\text{rej}, \triangleright, R) & (\text{acc}, \sqsubset, S) & (\text{rej}, 0, S) & (q', 1, L) \\
\end{array}
\]

a string of \( n \) 1s

**Claim:** the computation of \( M \) starting from configuration \( (s, \triangleright, \sqsubset 1^n 0) \) halts in configuration \( (\text{acc}, \triangleright, \sqsubset 1^{n+1} 0) \).
The computation of $M$ starting from configuration $(s, \triangleright, \sqcup 1^n 0)$:

$(s, \triangleright, \sqcup 1^n 0) \rightarrow_M (s, \triangleright \sqcup, 1^n 0)$

$\vdots$

$(q, \triangleright \sqcup 1^n, 0)$

$(q, \triangleright \sqcup 1^n 0, \varepsilon)$

$(q, \triangleright \sqcup 1^{n+1} \sqcup, \varepsilon)$

$(q', \triangleright \sqcup 1^{n+1}, 0)$

$\vdots$

$(q', \triangleright \sqcup, 1^{n+1} 0)$

$(\text{acc}, \triangleright \sqcup, 1^{n+1} 0)$

\{tape head moving right\}
The computation of $M$ starting from configuration $(s, \triangleright, \sqcup 1^n0)$:

$(s, \triangleright, \sqcup 1^n0) \rightarrow_M (s, \triangleright \sqcup, 1^n0) \rightarrow_M (q, \triangleright \sqcup 1, 1^{n-1}0) \rightarrow_M (q, \triangleright \sqcup 1^n0, 0) \rightarrow_M (q, \triangleright \sqcup 1^{n+1}0, \varepsilon) \rightarrow_M (q', \triangleright \sqcup 1^{n+1}, 0) \rightarrow_M (q', \triangleright \sqcup, 1^{n+1}0) \rightarrow_M (\text{acc}, \triangleright \sqcup, 1^{n+1}0)$
Theorem. The computation of a Turing machine $M$ can be implemented by a register machine.

Proof (sketch).

Step 1: fix a numerical encoding of $M$’s states, tape symbols, tape contents and configurations.

Step 2: implement $M$’s transition function (finite table) using RM instructions on codes.

Step 3: implement a RM program to repeatedly carry out $\rightarrow_M$. 
Step 1

- Identify states and tape symbols with particular numbers:

\[
\begin{align*}
\text{acc} &= 0 \\
\text{ rej} &= 1 \\
Q &= \{2, 3, \ldots, n\} \\
\Sigma &= \{0, 1, \ldots, m\}
\end{align*}
\]

- Code configurations \( c = (q, w, u) \) by:

\[
\overline{c} = \overline{[q, \overline{[a_n, \ldots, a_1]}, \overline{[b_1, \ldots, b_m]}]}\]

where \( w = a_1 \cdots a_n \ (n > 0) \) and \( u = b_1 \cdots b_m \ (m \geq 0) \) say.
Step 1

Code configurations $c = (q, w, u)$ by:

$\overline{c} = \overline{[q, [a_n, \ldots, a_1], [b_1, \ldots, b_m]]}$

where $w = a_1 \cdots a_n \ (n > 0)$ and $u = b_1 \cdots b_m \ (m \geq 0)$ say.

reversal of $w$ makes it easier to use our RM programs for list manipulation
Step 2

Using registers

\[ Q = \text{current state} \]

\[ A = \text{current tape symbol} \]

\[ D = \text{current direction of tape head} \]

**NB.** (with \( L = 0, R = 1 \) and \( S = 2 \), say)

one can turn the finite table of (argument, result)-pairs specifying \( \delta \) into a RM program \( (Q, A, D) \rightarrow \delta(Q, A) \rightarrow \) so that starting the program with \( Q = q, A = a, D = d \) (and all other registers zeroed), it halts with \( Q = q', A = a', D = d' \), where \( (q', a', d') = \delta(q, a) \).
Step 3

The next slide specifies a RM to carry out $M$'s computation. It uses registers

\[ C = \text{code of current configuration} \]

\[ W = \text{code of tape symbols at and left of tape head (reading right-to-left)} \]

\[ U = \text{code of tape symbols right of tape head (reading left-to-right)} \]

Starting with $C$ containing the code of an initial configuration (and all other registers zeroed), the RM program halts if and only if $M$ halts; and in that case $C$ holds the code of the final configuration.