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

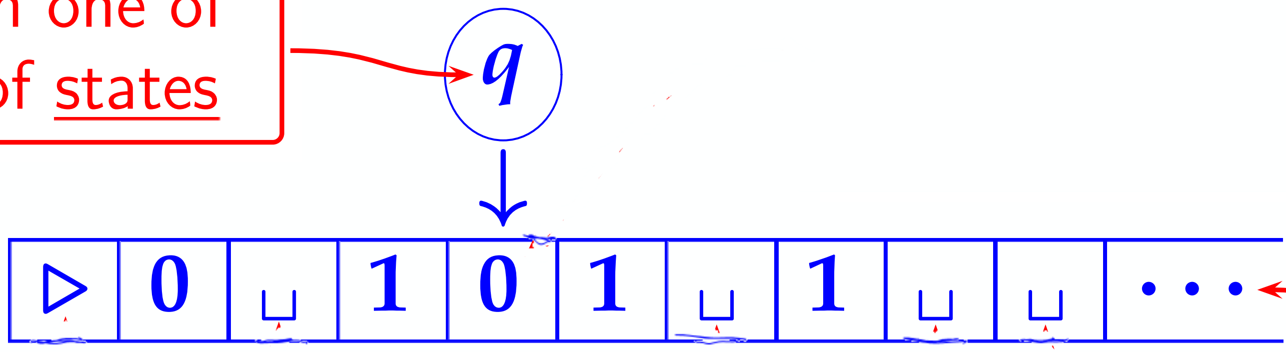
Register Machine computation abstracts away from any particular, concrete representation of numbers (e.g. as bit strings) and the associated elementary operations of increment/decrement/zero-test.

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

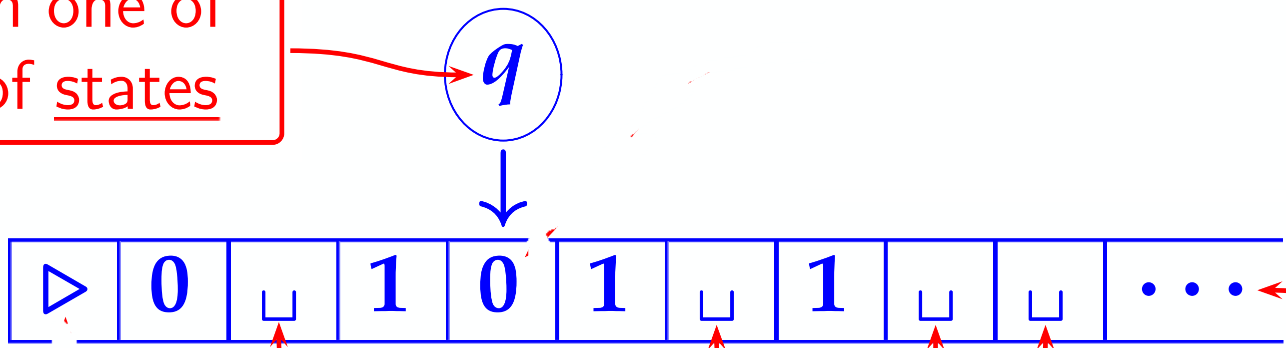
machine is in one of a finite set of states



linear tape, unbounded to 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

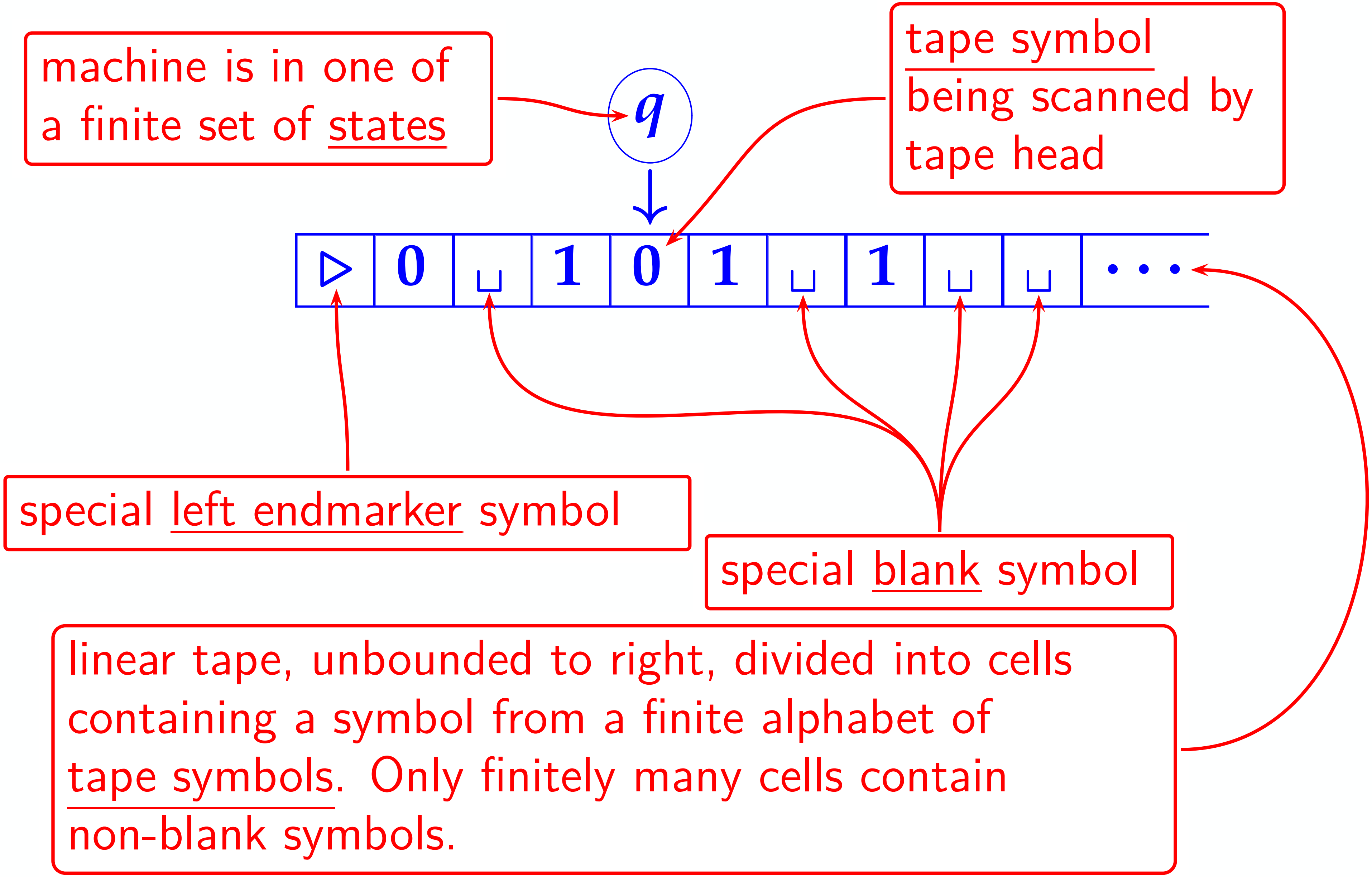
machine is in one of
a finite set of <u>states</u>



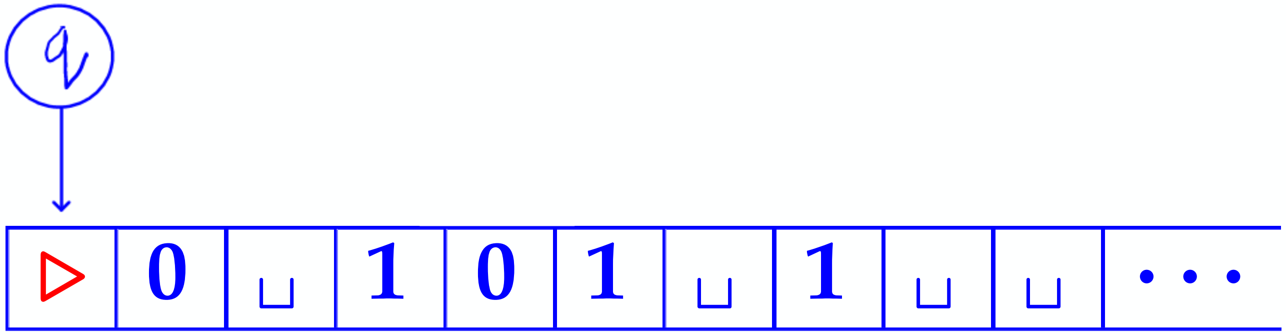
special <u>blank</u> symbol

linear tape, unbounded to right, divided into cells containing a symbol from a finite alphabet of <u>tape symbols</u>. Only finitely many cells contain non-blank symbols.

Turing machines, informally

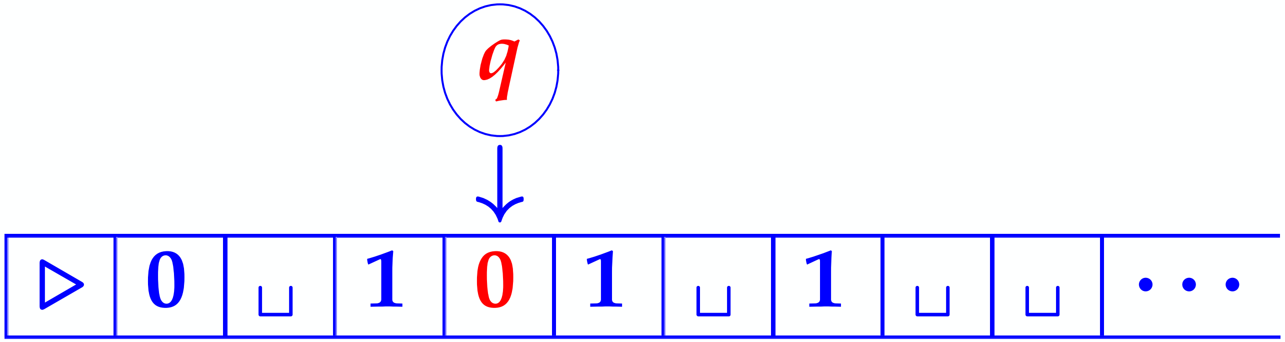


Turing machines, informally



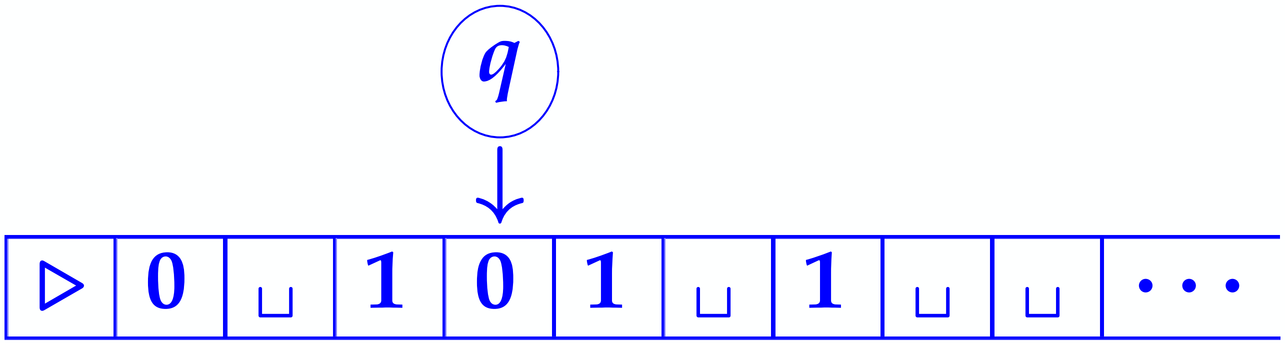
- Machine starts with tape head pointing to the special left endmarker ▷.

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 ($\mathbf{0}$).

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 ($\mathbf{0}$).
- 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
- Σ, finite set of tape symbols (disjoint from Q) containing distinguished symbols $\triangleright$ (left endmarker) and $\sqcup$ (blank)
- $s \in Q$, an initial state
- $\delta \in (Q \times \Sigma) \to (Q \cup \{\mathtt{acc}, \mathtt{rej}\}) \times \Sigma \times \{L, R, S\}$, a transition function—specifies for each state and symbol a next state (or accept $\mathtt{acc}$ or reject $\mathtt{rej}$), 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
- Σ, finite set of tape symbols (disjoint from Q) containing distinguished symbols $\rhd$ (left endmarker) and $\sqcup$ (blank)
- $s \in Q$, an initial state
- $\delta \in (Q \times \Sigma) \to (Q \cup \{\mathtt{acc}, \mathtt{rej}\}) \times \Sigma \times \{L, R, S\}$, a transition function, satisfying:

  for all $q \in Q$, there exists $q' \in Q \cup \{\mathtt{acc}, \mathtt{rej}\}$ with $\delta(q, \rhd) = (q', \rhd, R)$
  (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 $\Sigma = \{\triangleright, \sqcup, 0, 1\}$

transition function

$\delta \in (Q \times \Sigma) \to (Q \cup \{\mathtt{acc}, \mathtt{rej}\}) \times \Sigma \times \{L, R, S\}$:

δ	$\triangleright$	$\sqcup$	0	1
s	$(s, \triangleright, R)$	$(q, \sqcup, R)$	$(\mathtt{rej}, 0, S)$	$(\mathtt{rej}, 1, S)$
q	$(\mathtt{rej}, \triangleright, R)$	$(q', 0, L)$	$(q, 1, R)$	$(q, 1, R)$
q'	$(\mathtt{rej}, \triangleright, R)$	$(\mathtt{acc}, \sqcup, S)$	$(\mathtt{rej}, 0, S)$	$(q', 1, L)$

Turing machine computation

Turing machine configuration: (q, w, u)

where

- $q \in Q \cup \{\mathtt{acc}, \mathtt{rej}\}$ = 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 $\sqcup$)

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

(not uniquely)

Turing machine computation

Turing machine configuration: (q, w, u)

where

- $q \in Q \cup \{\mathtt{acc}, \mathtt{rej}\}$ = 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 $\sqcup$)

Initial configurations: $(s, \triangleright, u)$

Turing machine computation

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

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

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

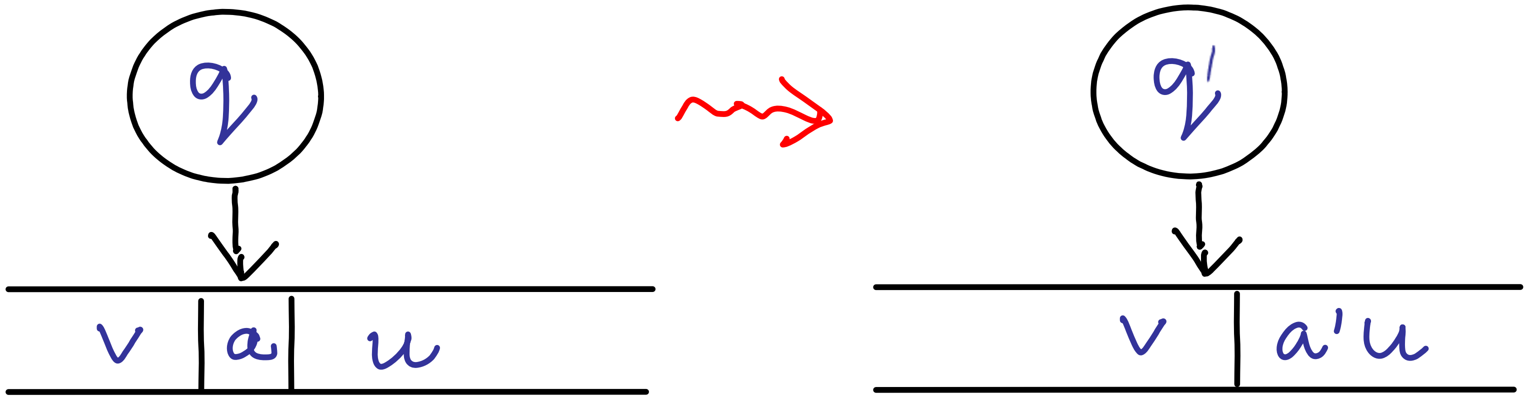
either $\delta(q, a) = (q', a', L)$, $w' = v$, and $u' = a'u$

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 $u' = u''$

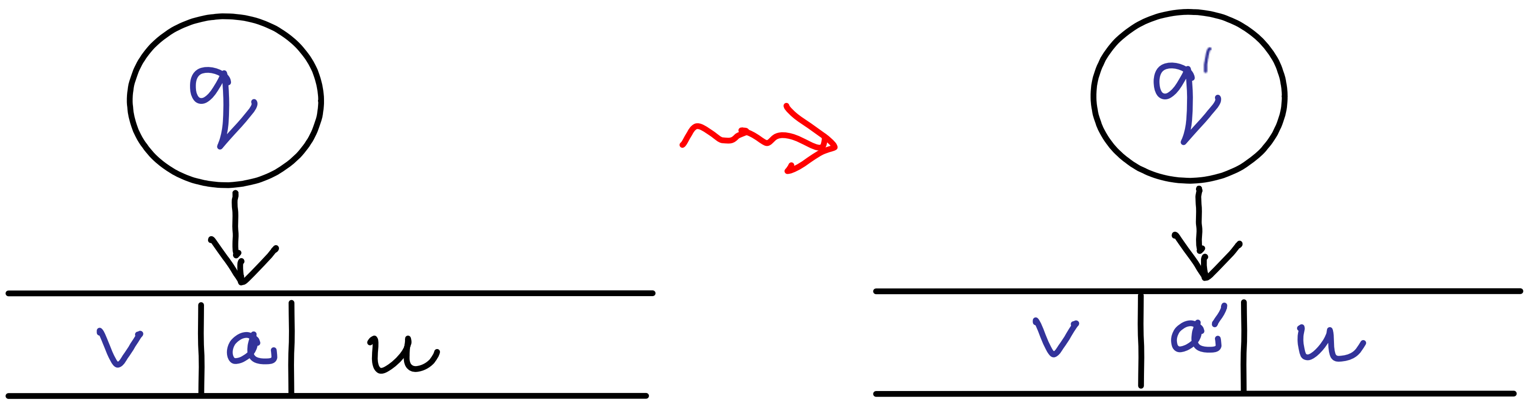
or $\delta(q, a) = (q', a', R)$, $u = \varepsilon$ is empty, $w' = va'{\sqcup}$ and $u' = \varepsilon$.

$$\delta(q, a) = (q', a', L)$$



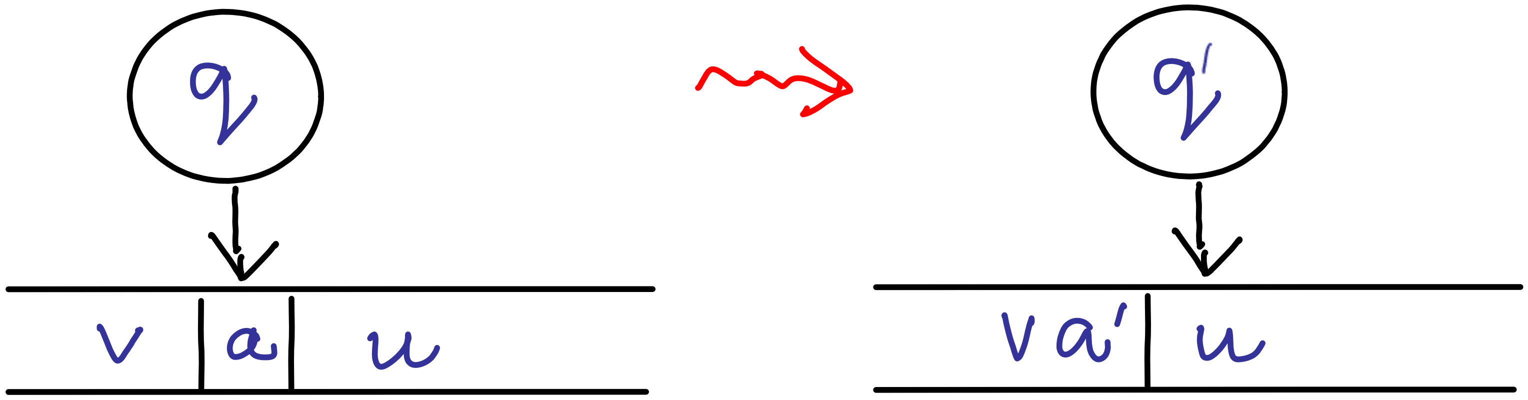
$$(q, va, u) \xrightarrow[M]{} (q', v, a'u)$$

$$\delta(q, a) = (q', a', S)$$



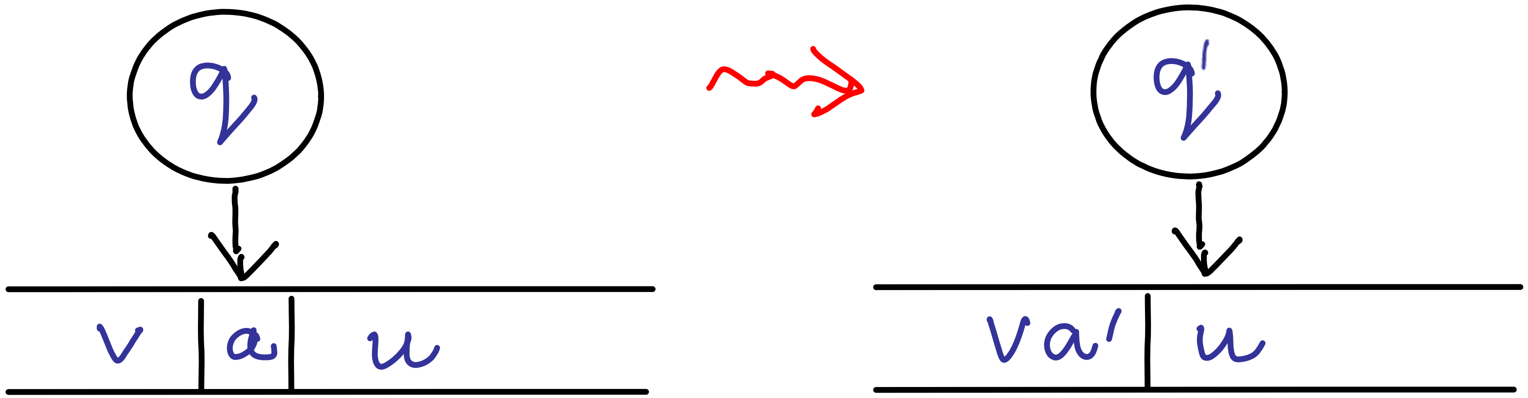
$$(q, va, u) \xrightarrow[M]{} (q', va', u)$$

$$\delta(q, a) = (q', a', R)$$



$$(q, va, u) \longrightarrow_M (q', ?, ?)$$

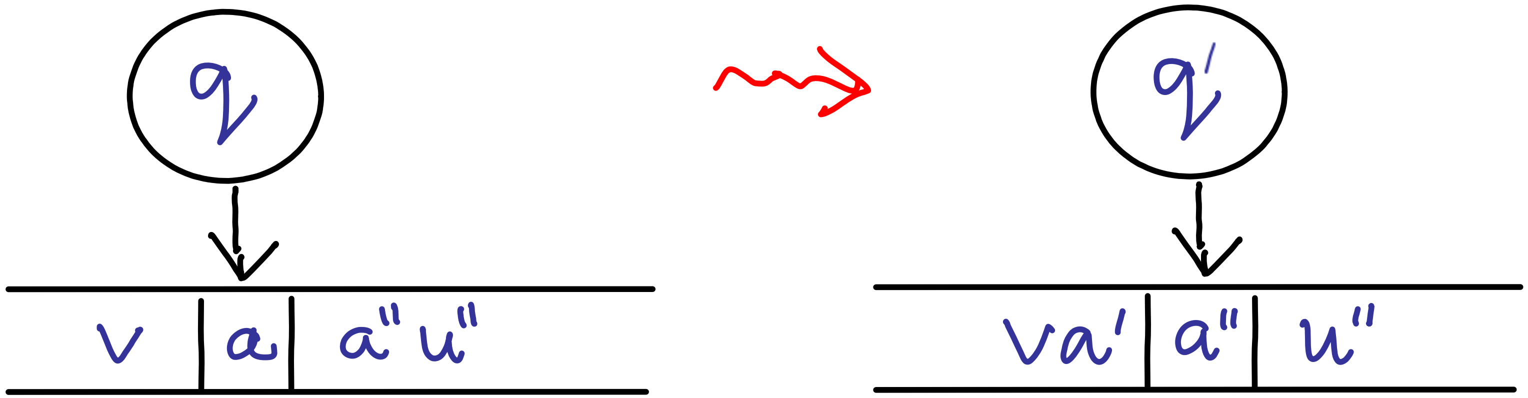
$$\delta(q, a) = (q', a', R)$$



$$(q, va, u) \longrightarrow_M (q', ?, ?)$$

Two cases : $\begin{cases} u = a'' u'' & \text{is non-empty} \\ u = \varepsilon & \text{is empty} \end{cases}$

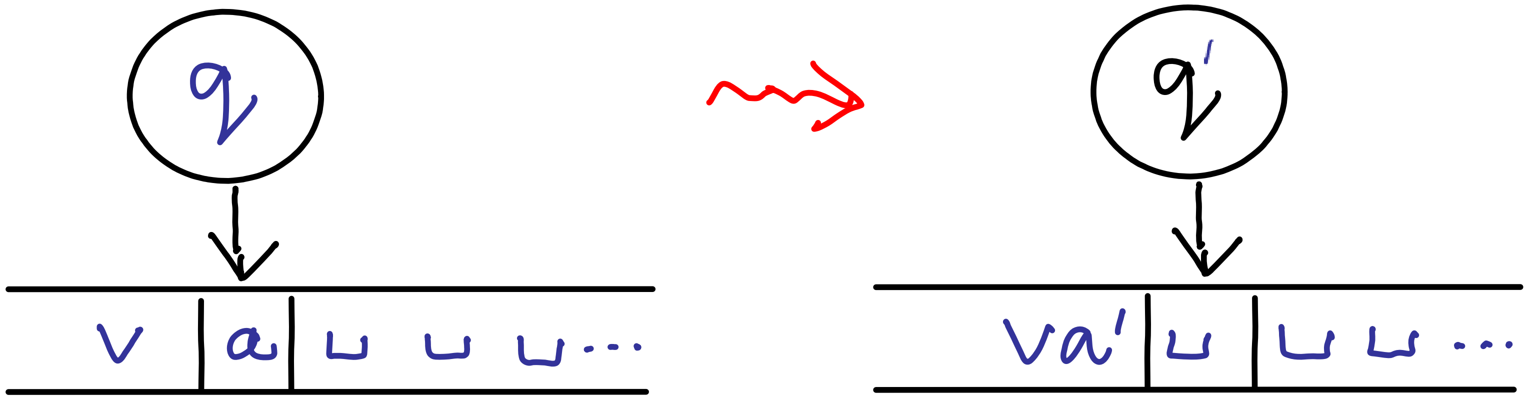
$$\delta(q, a) = (q', a', R)$$



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

Two cases : $\begin{cases} u = a''u'' & \text{is non-empty} \\ u = \varepsilon & \text{is empty} \end{cases}$

$$\delta(q, a) = (q', a', R)$$



$$(q, va, \varepsilon) \longrightarrow_M (q', va'\sqcup, \varepsilon)$$

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
- $c_i \rightarrow_M c_{i+1}$ 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 $(\mathtt{acc}, w, u)$ or $(\mathtt{rej}, w, u)$.

Example Turing Machine

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

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

symbols $\Sigma = \{\triangleright, \sqcup, 0, 1\}$

transition function

$\delta \in (Q \times \Sigma) \to (Q \cup \{\mathtt{acc}, \mathtt{rej}\}) \times \Sigma \times \{L, R, S\}$:

δ	$\triangleright$	$\sqcup$	0	1
s	$(s, \triangleright, R)$	$(q, \sqcup, R)$	$(\mathtt{rej}, 0, S)$	$(\mathtt{rej}, 1, S)$
q	$(\mathtt{rej}, \triangleright, R)$	$(q', 0, L)$	$(q, 1, R)$	$(q, 1, R)$
q'	$(\mathtt{rej}, \triangleright, R)$	$(\mathtt{acc}, \sqcup, S)$	$(\mathtt{rej}, 0, S)$	$(q', 1, L)$

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

Example Turing Machine

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

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

symbols $\Sigma = \{\triangleright, \sqcup, 0, 1\}$

transition function

$\delta \in (Q \times \Sigma) \to (Q \cup \{\texttt{acc}, \texttt{rej}\}) \times \Sigma \times \{L, R, S\}$:

δ	$\triangleright$	$\sqcup$	0	1
s	$(s, \triangleright, R)$	$(q, \sqcup, R)$	$(\texttt{rej}, 0, S)$	$(\texttt{rej}, 1, S)$
q	$(\texttt{rej}, \triangleright, R)$	$(q', 0, L)$	$(q, 1, R)$	$(q, 1, R)$
q'	$(\texttt{rej}, \triangleright, R)$	$(\texttt{acc}, \sqcup, S)$	$(\texttt{rej}, 0, S)$	$(q', 1, L)$

a string of n 1s

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

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

$$
\begin{aligned}
(s, \triangleright, {\sqcup}1^n0) &\to_M (s, \triangleright{\sqcup}, 1^n0)\\
&\to_M (q, \triangleright{\sqcup}1, 1^{n-1}0)\\
&\quad\vdots\\
&\to_M (q, \triangleright{\sqcup}1^n, 0)\\
&\to_M (q, \triangleright{\sqcup}1^n0, \varepsilon)\\
&\to_M (q, \triangleright{\sqcup}1^{n+1}{\sqcup}, \varepsilon)\\
&\to_M (q', \triangleright{\sqcup}1^{n+1}, 0)\\
&\quad\vdots\\
&\to_M (q', \triangleright{\sqcup}, 1^{n+1}0)\\
&\to_M (\mathrm{acc}, \triangleright{\sqcup}, 1^{n+1}0)
\end{aligned}
$$

tape head moving right

The computation of M starting from configuration $(s, \triangleright, \sqcup 1^n 0)$:

$$
\begin{aligned}
(s, \triangleright, \sqcup 1^n 0) &\to_M (s, \triangleright\sqcup, 1^n 0) \\
&\to_M (q, \triangleright\sqcup 1, 1^{n-1} 0) \\
&\quad\vdots \\
&\to_M (q, \triangleright\sqcup 1^n, 0) \\
&\to_M (q, \triangleright\sqcup 1^n 0, \varepsilon) \\
&\to_M (q, \triangleright\sqcup 1^{n+1}\sqcup, \varepsilon) \\
&\to_M (q', \triangleright\sqcup 1^{n+1}, 0) \\
&\quad\vdots \\
&\to_M (q', \triangleright\sqcup, 1^{n+1} 0) \\
&\to_M (\mathrm{acc}, \triangleright\sqcup, 1^{n+1} 0)
\end{aligned}
$$

tape head moving right

tape head moving left

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 $\to_M$.

Step 1

- Identify states and tape symbols with particular numbers:

N.B.

$$\begin{array}{rcl|rcl} \mathtt{acc} & = & \mathbf{0} & \sqcup & = & \mathbf{0} \\ \mathtt{rej} & = & \mathbf{1} & \triangleright & = & \mathbf{1} \\ Q & = & \{2,3,\ldots,n\} & \Sigma & = & \{0,1,\ldots,m\} \end{array}$$

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

$$\ulcorner c \urcorner = \ulcorner [q, \ulcorner [a_n, \ldots, a_1] \urcorner, \ulcorner [b_1, \ldots, b_m] \urcorner] \urcorner$$

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

Step 1

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

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

$$\ulcorner c \urcorner = \ulcorner [q, \ulcorner [a_n, \ldots, a_1] \urcorner, \ulcorner [b_1, \ldots, b_m] \urcorner] \urcorner$$

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

Step 2

Using registers

$$\begin{aligned} \mathtt{Q} &= \text{current state} \\ \mathtt{A} &= \text{current tape symbol} \\ \mathtt{D} &= \text{current direction of tape head} \end{aligned}$$

N.B. $\longrightarrow$ (with $L = 0$, $R = 1$ and $S = 2$)

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

Step 3

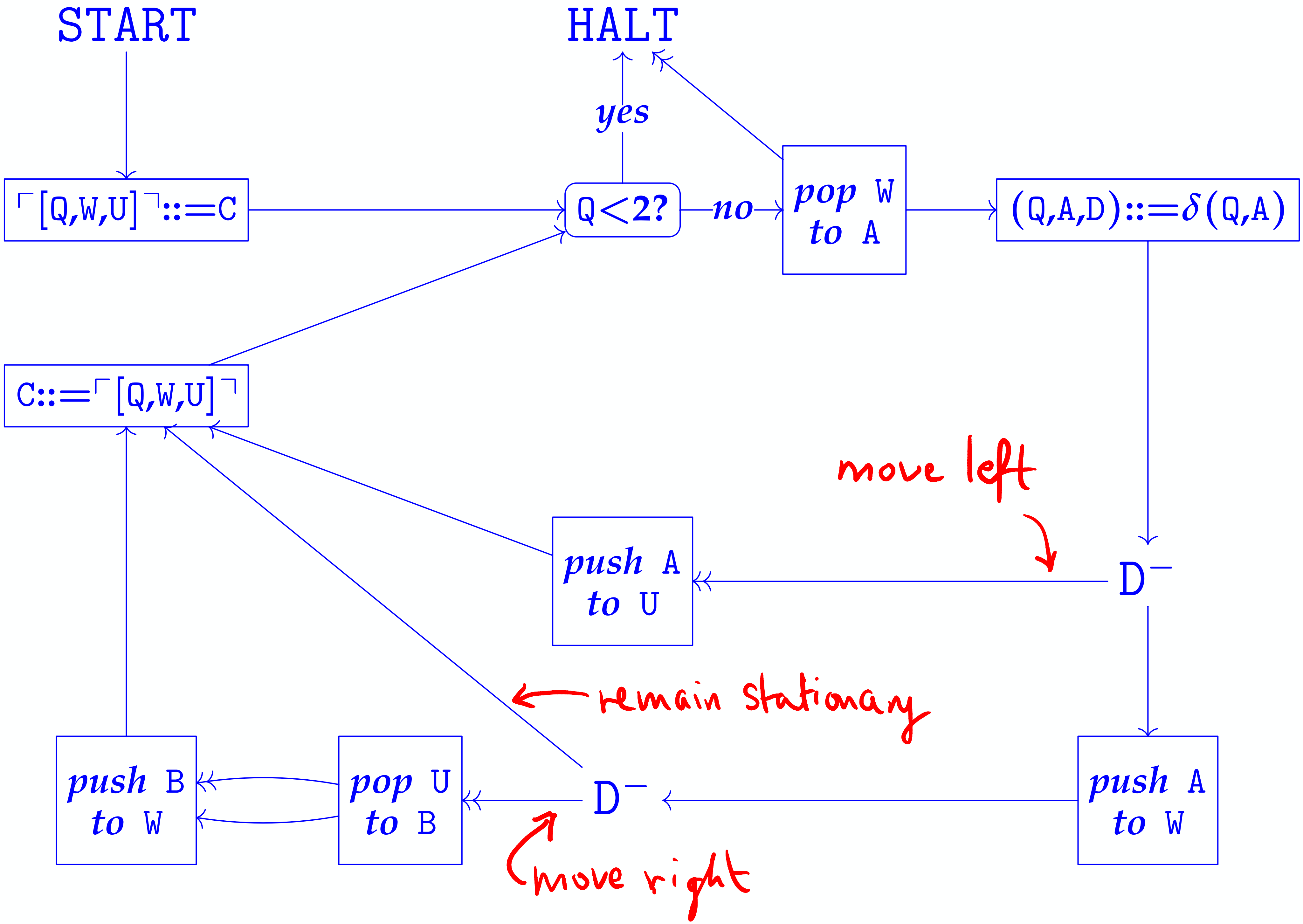
The next slide specifies a RM to carry out $\boldsymbol{M}$'s computation. It uses registers

$\mathtt{C}$ = code of current configuration

$\mathtt{W}$ = code of tape symbols at and left of tape head (reading right-to-left)

$\mathtt{U}$ = code of tape symbols right of tape head (reading left-to-right)

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


START
HALT
yes
no
⌜[Q,W,U]⌝::=C
Q<2?
pop W to A
(Q,A,D)::=δ(Q,A)
C::=⌜[Q,W,U]⌝
move left
D⁻
push A to U
remain stationary
push B to W
pop U to B
D⁻
push A to W
move right