## 4.2 Extensions of Turing machines

### 1. Multitrack machines

The tape of the Turing machines is considered to consist of $k$ parallel tracks, all of which are read and written in a single computation step.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | L | A | N | # | # | # | # | | |
| M | A | T | H | I | S | O | N | | ⋯ |
| T | U | R | I | N | G | # | # | | |

tape head:

The values of the transition functions are then of the form

$$\delta(q,(a_1,\ldots,a_k)) = (q',(b_1,\ldots,b_k),\Delta),$$

where $a_1,\ldots,a_k$ are the symbols read from tracks $1,\ldots,k$, $b_1,\ldots,b_k$ the symbols written to replace them, and $\Delta \in \{L,R\}$ is the direction the tape head is moved.

In the beginning the input is placed at the start of track 1; the corresponding locations on other tracks are filled with special blank signs #.





Formally a *k-track Turing machine* is a 7-tuple

$$M = (Q,\Sigma,\Gamma,\delta,q_0,q_{\text{acc}},q_{\text{rej}}),$$

where all components are as in the standard model, except the transition function:

$$\delta : (Q - \{q_{\text{acc}},q_{\text{rej}}\}) \times (\Gamma^k \cup \{>,<\}) \to Q \times (\Gamma^k \cup \{>,<\}) \times \{L,R\}.$$

The definitions of the leads to -relation, initial state etc. are the same as in the standard model, with small modifications.





**Theorem 4.1.** If a formal language $L$ can be recognized with a $k$-track Turing machine, then it can also be recognized by a standard model Turing machine.

*Proof.* Let $M = (Q,\Sigma,\Gamma,\delta,q_0,q_{\text{acc}},q_{\text{rej}})$ be $k$-track Turing machine, that recognizes the language $L$. The corresponding standard model Turing machine $\widehat{M}$ is constructed as follows:

$$\widehat{M} = (\hat{Q},\Sigma,\hat{\Gamma},\hat{\delta},\hat{q}_0,q_{\text{acc}},q_{\text{rej}}),$$

where $\hat{Q} = Q \cup \{\hat{q}_0,\hat{q}_1,\hat{q}_2\}$, $\hat{\Gamma} = \Sigma \cup \Gamma^k$ and for all $q \in Q$ we have

$$\hat{\delta}(q,\begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}) = (q',\begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix},\Delta),$$

when $\delta(q,(a_1,\ldots,a_k)) = (q',(b_1,\ldots,b_k),\Delta)$.

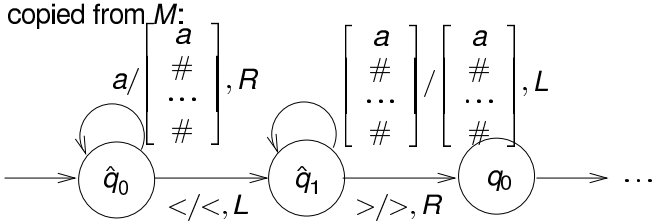




At the beginning of the computation of $\widehat{M}$ the input must be "lifted" to track 1, that is, to replace $a_1a_2\ldots a_n$ on the input tape by

$$\begin{bmatrix} a_1 \\ \# \\ \vdots \\ \# \end{bmatrix}\begin{bmatrix} a_2 \\ \# \\ \vdots \\ \# \end{bmatrix}\cdots\begin{bmatrix} a_n \\ \# \\ \vdots \\ \# \end{bmatrix}.$$

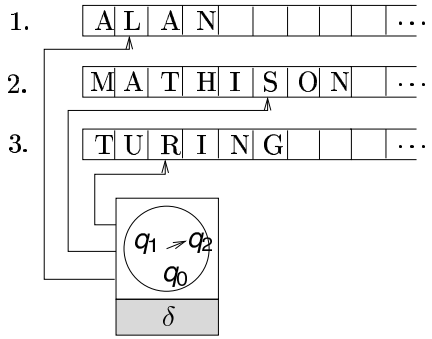
For this purpose a small "preprocessor" is added to the transition function copied from $M$:




$</<, L$ $>/>, R$

□

## 2. Multitape machines




The Turing machine is allowed to have $k$ independent tapes, each of which has a separate tape head. The machine reads and writes all tapes in a single computation step. At the beginning of the computation the input is placed at the beginning of tape 1 and all tape heads are placed at the beginning of their respective tapes.





The values of the transition function of such a machine have the form

$$\delta(q, a_1, \ldots, a_k) = (q', (b_1, \Delta_1), \ldots, (b_k, \Delta_k)),$$

where $a_1, \ldots, a_k$ are the symbols read from tapes $1, \ldots, k$, which will be replaced by $b_1, \ldots, b_k$, and $\Delta_1, \ldots, \Delta_k \in \{L, R\}$ the directions the tape heads are moved.

Formally a *$k$-tape Turing machine* is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

where the other components are as in the standard model, except the transition function:

$$\delta : (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times (\Gamma \cup \{>, <\})^k \to Q \times ((\Gamma \cup \{>, <\}) \times \{L, R\})^k.$$

The leads to -relation and other concepts are defined as for the standard models, with minor modifications.



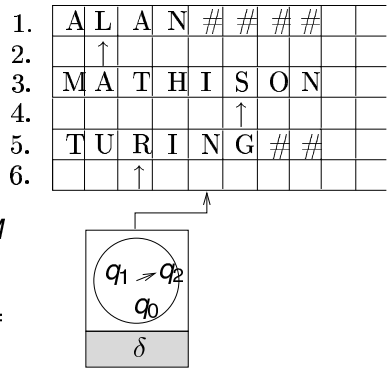


**Theorem 4.2.** If a formal language $L$ can be recognized by a $k$-tape Turing machine, it can also be recognized by a standard model Turing machine.

*Proof (idea).* Let

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

be a $k$-tape Turing machine, that recognizes $L$. The machine $M$ can be simulated by a $2k$-track machine $\widehat{M}$ so that the odd tracks $1, 3, 5, \ldots, 2k-1$ of $\widehat{M}$ correspond with tapes $1, 2, \ldots, k$ of $M$ and on the even track following each odd track the symbol $\uparrow$ denotes the location of the simulated tape head.








First the input string is lifted to track 1 of $\widehat{M}$. In its first transition $\widehat{M}$ places the tape head indicators $\uparrow$ in the first positions of the even-numbered tapes.

After this $\widehat{M}$ "sweeps" the tape back and forth between the beginning and the end. Sweeping from left to right $\widehat{M}$ collects information about the symbol at each tape head in $M$. When all input symbols are determined, $\widehat{M}$ simulates a step of $M$, and in sweeping back from right to left it replaces the characters on the odd tapes at the $\uparrow$-pointers and moves the pointers. □

## 3. Nondeterministic machines

Formally a *nondeterministic Turing machine* is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

where all other components are as in the deterministic standard model except the transition function:

$$\delta : (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times (\Gamma \cup \{>, <\}) \to \mathcal{P}(Q \times (\Gamma \cup \{>, <\}) \times \{L, R\}).$$

The interpretation of the value of the transition function

$$\delta(q, a) = \{(q_1, b_1, \Delta_1), \ldots, (q_k, b_k, \Delta_k)\}$$

is that when the machine reads $a$ in state $q$ it may act according to one of the triplets $(q_i, b_i, \Delta_i)$.





The configurations, leads to -relation etc. are defined formally as for the deterministic machine, except that the condition $\delta(q, a) = (q', b, \Delta)$ is replaced by $(q', b, \Delta) \in \delta(q, a)$.

Because of this change the successor relation $\underset{M}{\vdash}$ is no longer a function: The configuration $(q, w)$ may now have several alternative successors, i.e., configurations $(q', w')$, for which $(q, w) \underset{M}{\vdash} (q', w')$.

The language $M$ recognizes is defined by

$$L(M) = \{x \in \Sigma^* \mid (q_0, \underline{x}) \underset{M}{\vdash}^* (q_{\text{acc}}, w) \text{ for some } w \in \Gamma^*\}.$$

For a nondeterministic machine $M$ the string $x$ is in the language recognized by $M$, jos *some* possible configuration sequence of $M$ leads from the initial configuration with input $x$ to an accepting final configuration.





**Example.** Recognizing composite numbers by a nondeterministic Turing machine.

A non-negative integer $n$ is *composite*, if it has integer factors $p, q \geq 2$, for which $pq = n$. If some $n \geq 2$ is not composite, it is *prime*.

Assume that we have already designed a deterministic machine CHECK_MULT, that recognizes the language

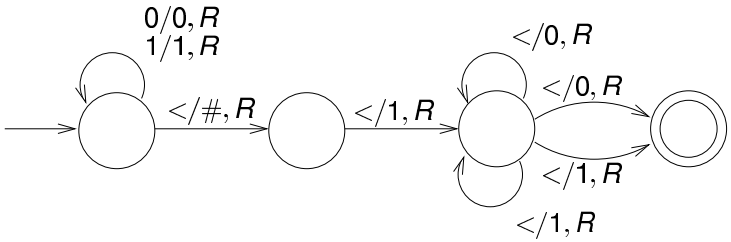
$$L(\text{CHECK\_MULT}) = \{n\#p\#q \mid n, p, q \text{ binary numbers}, n = pq\}.$$

Let GO_START be a deterministic Turing machine, that moves the tape head back to the first position.





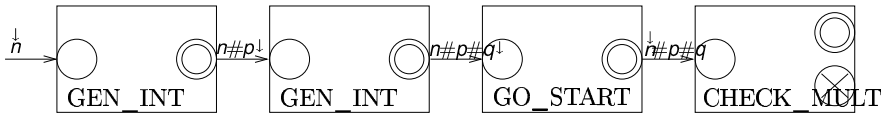
Let GEN_INT be the following nondeterministic Turing machine that writes an arbitrary binary number at the end of the tape:



The nondeterministic Turing machine TEST_COMPOSITE, that recognizes

$$L(\text{TEST\_COMPOSITE}) = \{n \mid n \text{ is a binary composite number}\}$$

can be combined from these components:

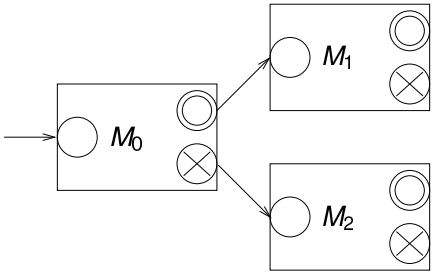







The combined machine accepts a binary number $n$ given as input if and only if there are binary numbers $p, q \geq 2$, for which $n = pq$ — that is if and only if $n$ is composite.

*Note.* A general way of expressing combining Turing machines:








**Theorem 4.3** If the formal language $L$ can be recognized by a nondeterministic Turing machine, it can also be recognized by a standard model deterministic Turing machine.

*Proof (idea).* Let

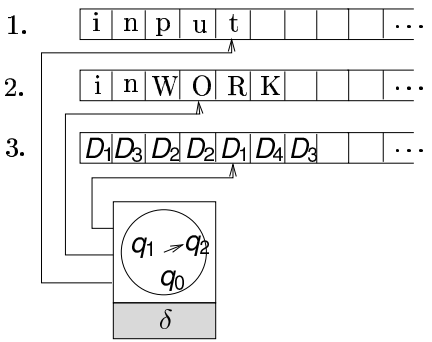
$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

be a nondeterministic Turing machine that recognizes $L$. The machine $M$ can be simulated by a three-tape deterministic machine $\widehat{M}$, that systematically examines the possible computations (configuration sequences) of $M$, until it finds an accepting one — if one exists. The machine $\widehat{M}$ can further be transformed to a standard model machine by the constructions in the previous Theorems.

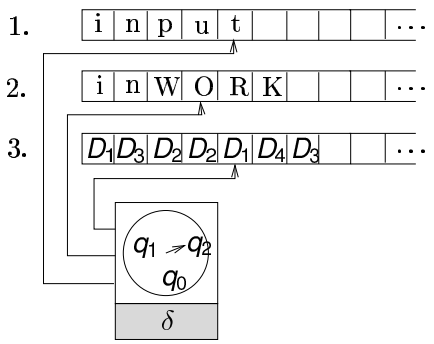




In more detail:
On tape 1 $\widehat{M}$ maintains a copy of the input and on tape 2 it simulates the tape of $M$. At the beginning of each computation $\widehat{M}$ copies the input from tape 1 to tape 2 and and wipes out any extra symbols left from a previous computation on tape 2.
On tape 3 $\widehat{M}$ keeps a record of "number" of the current computation.



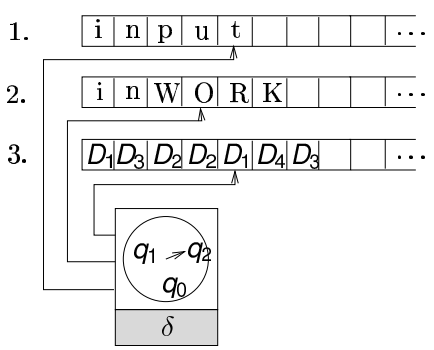
More exactly, let $r$ be the maximum number of values of the transition function from a given state for a given input symbol. Then $\widehat{M}$ generates strings of the special tape symbols $D_1, \ldots, D_r$ onto tape 3 in canonical order: $\varepsilon$, $D_1$, $D_2$, …, $D_r$, $D_1D_1$, $D_1D_2$, …, $D_1D_r$, $D_2D_1$, ….
For every generated string $\widehat{M}$ simulates a partial computation of $M$, where the nondeterministic choices are made in accordance to the symbol sequence on tape 3.





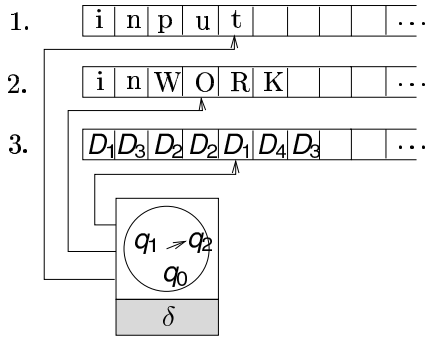



If for example tape 3 contains the string $D_1D_3D_2$, then in the first transition the machine makes choice 1, in the second choice 3, and in the third choice 2. If this computation did not lead to the accepting final state, the machine will generate the next sequence $D_1D_3D_3$ and start over.
If the sequnce is invalid, i.e., if in some configuration the number of the choice is larger than the number of choices available, the simulated computation is interrupted and the next sequence is generated.








Clearly this systematic exploration of the computations of $M$ leads $\widehat{M}$ to accept the input if and only if $M$ has a computation where the input is accepted. If no such computation exists, $\widehat{M}$ will never stop. □