§3. The Turing machine. In this note we introduce Turing's formal model of computing; there are many possible. Informally, a *Turing machine* consists of a tape, a head which 'scans' the tape, and a finite control. The tape is a linear sequence of squares, each of which contains one of a finite number of distinct symbols. The tape has a definite left hand end, but is unbounded to the right. There is a special symbol called the *blank symbol*; at any instant during the computation of a Turing machine, all but a finite number of tape squares contain the blank symbol. (So the existence of an unbounded tape should not worry us unduly: it represents a *potential* rather than *actual* infinity.) At any time instant, the head *scans* (or is positioned over) a particular tape square.

The Turing machine (TM) starts with some input written on the squares of the tape, and with the head positioned over the leftmost tape square. The machine then proceeds in a sequence of *moves* or *transitions*. A single move of the Turing machine is determined by the symbol s scanned by the tape head, and the current state q of the finite control. For some pairs (q, s) no move is defined; in this case the machine halts and is deemed to have *rejected* its input unless q is the final state as described below. If a move *is* defined for the pair (q, s), the machine undergoes three changes:

1. the finite control assumes a new state;
2. the head prints a symbol on the scanned tape square, overwriting whatever was previously there;
3. the head is moved one square left or right.

A certain state of the finite control is designated the *accepting* or *final state*; when the machine enters this state it immediately halts and is deemed to have *accepted* its input.

To avoid any possible misunderstandings, and to emphasize the point that the computational process described above is well defined and perfectly mechanical, we shall now present a more formal definition of the Turing machine. (As the course progresses, and we become more familiar with the model, much of this formality may safely be dropped. There is no intrinsic merit in formality.)

A Turing machine is formally described by a septuple $M = (Q, \Gamma, \Sigma, \flat, q_I, q_F, \delta)$, where

$$\begin{aligned} Q &\text{ is a finite set of } \textit{states}, \\ \Gamma &\text{ is a finite } \textit{tape alphabet}, \\ \Sigma \subset \Gamma &\text{ is the } \textit{input alphabet}, \\ \flat \in \Gamma - \Sigma &\text{ is the } \textit{blank symbol}, \\ q_I \in Q &\text{ is the } \textit{initial state}, \\ q_F \in Q &\text{ is the } \textit{final state}, \end{aligned}$$

and δ is the *transition function*, which is a partial function mapping $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$. ($L$ and R can be interpreted as left and right, respectively.)

The disposition of the machine M at any instant is completely described by the sequence $\alpha q \beta$, where $q \in Q$ is the current state of M, $\alpha \in \Gamma^*$ is the finite sequence of tape symbols reading from the left hand end of the tape up to but not including the scanned symbol, and β is the infinite sequence of tape symbols starting at the scanned symbol and reading to the right. Note that the sequence α may be empty (corresponding to the head being positioned over the leftmost tape square). We shall temporarily refer to the infinite sequence $\alpha q \beta$ so derived as the *configuration* [9] *of* M.

We now define a *move* of the machine M. Let

$$x_1, x_2, \ldots, x_{i-1}, q, x_i, x_{i+1}, x_{i+2}, \ldots$$

be a configuration of M. If the (partial) function δ is not defined on the pair (q, x_i) then the machine halts. If $\delta(q, x_i)$ is defined, and is equal to (q', y, R), then the new configuration of M after a single move is

$$x_1, x_2, \ldots, x_{i-1}, y, q', x_{i+1}, x_{i+2}, \ldots .$$

Finally suppose $\delta(q, x_i) = (q', y, L)$. If $i = 1$, the machine M halts and rejects. (The head is not allowed to drop off the left hand end of the tape.) Otherwise the new configuration of M is

$$x_1, x_2, \ldots, x_{i-2}, q', x_{i-1}, y, x_{i+1}, x_{i+2}, \ldots .$$

Now observe that the infinite sequence $\gamma = \alpha q \beta$, which we have been referring to as a configuration of M, contains only a finite number of non-blank symbols. Thus, γ may be truncated to a finite sequence, without any loss of information, simply by stripping away all trailing blanks. From now on, the term *configuration* will be used exclusively to refer to the *finite* sequence obtained from γ in this way.

If γ_0 and γ_1 are configurations of M, then we write $\gamma_0 \vdash \gamma_1$ if γ_1 follows from γ_0 via a single move, and $\gamma_0 \vdash^* \gamma_1$ if γ_1 can be reached from γ_0 via some finite (possibly empty) sequence of moves. The *language accepted by* M, denoted by $L(M)$, is the set of all words $x \in \Sigma^*$ such that M, when run with x as input, eventually enters the accepting state. Formally,

$$L(M) = \{x \in \Sigma^* : q_I x \vdash^* \alpha q_F \beta, \text{ where } \alpha, \beta \in \Gamma^*\}.$$

The *computation of* M *on input* x is the sequence of configurations $\gamma_0, \gamma_1, \gamma_2, \ldots$, where $\gamma_0 = q_I x$ and $\gamma_i \vdash \gamma_{i+1}$ for each i. A computation may be finite (terminating) or infinite (non-terminating). This completes the formal definition of a Turing machine.

It is convenient to establish some conventions for writing down the transition function of a Turing machine M. Two conventions are commonly employed. In the first, the transition function δ is presented as a list of quintuples. For each pair $(q, s) \in Q \times \Gamma$ there is at most one quintuple of the form (q, s, q', s', d) where $(q', s', d) \in Q \times \Gamma \times \{L, R\}$.

[9] These 'configurations' should not be confused with the 'm-configurations' defined in the extract from Turing's paper given in NOTE 2. The latter correspond instead to states, $q \in Q$, of what we have called the 'finite control'. Hence there are only a finite number of possible m-configurations but infinitely many configurations.

If no such quintuple exists, then $\delta(q, s)$ is undefined; otherwise, $\delta(q, s) = (q', s', d)$. The second convention represents the transition function as a directed multigraph[10] on vertex set Q, whose edges are labelled by triples. The presence of a directed edge from vertex q to vertex q' with label $(s, s', d) \in \Gamma \times \Gamma \times \{L, R\}$ denotes the fact that $\delta(q, s) = (q', s', d)$. Again, for each q and s there can be at most one such edge.

EXAMPLE Consider the machine M_{palin} with $Q = \{q_0, q_1, \ldots, q_6\}$, $q_I = q_0$, $q_F = q_6$, $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, $\Gamma = \{\mathtt{0}, \mathtt{1}, \text{ƀ}\}$, and with transition function specified by the set of quintuples

$$\begin{array}{lll}
\{\ (q_0, \text{ƀ}, q_6, \text{ƀ}, R), & (q_0, \mathtt{0}, q_1, \text{ƀ}, R), & (q_0, \mathtt{1}, q_3, \text{ƀ}, R), \\
\ \ (q_1, \text{ƀ}, q_2, \text{ƀ}, L), & (q_1, \mathtt{0}, q_1, \mathtt{0}, R), & (q_1, \mathtt{1}, q_1, \mathtt{1}, R), \\
\ \ (q_2, \text{ƀ}, q_6, \text{ƀ}, R), & (q_2, \mathtt{0}, q_5, \text{ƀ}, L), & \\
\ \ (q_3, \text{ƀ}, q_4, \text{ƀ}, L), & (q_3, \mathtt{0}, q_3, \mathtt{0}, R), & (q_3, \mathtt{1}, q_3, \mathtt{1}, R), \\
\ \ (q_4, \text{ƀ}, q_6, \text{ƀ}, R), & & (q_4, \mathtt{1}, q_5, \text{ƀ}, L), \\
\ \ (q_5, \text{ƀ}, q_0, \text{ƀ}, R), & (q_5, \mathtt{0}, q_5, \mathtt{0}, L), & (q_5, \mathtt{1}, q_5, \mathtt{1}, L)\ \}.
\end{array}$$

The language accepted by M_{palin} is the set of all palindromes in $\{\mathtt{0}, \mathtt{1}\}^*$. A typical accepting computation of M_{palin} is the following, where the input is the palindrome $\mathtt{0110}$:

$$\begin{array}{lllllllllll}
q_0\mathtt{0110} & \vdash & \text{ƀ}q_1\mathtt{110} & \vdash & \text{ƀ}\mathtt{1}q_1\mathtt{10} & \vdash & \text{ƀ}\mathtt{11}q_1\mathtt{0} & \vdash & \text{ƀ}\mathtt{110}q_1 & \vdash & \\
\text{ƀ}\mathtt{11}q_2\mathtt{0} & \vdash & \text{ƀ}\mathtt{1}q_5\mathtt{1} & \vdash & \text{ƀ}q_5\mathtt{11} & \vdash & q_5\text{ƀ}\mathtt{11} & \vdash & \text{ƀ}q_0\mathtt{11} & \vdash & \\
\text{ƀƀ}q_3\mathtt{1} & \vdash & \text{ƀƀ}\mathtt{1}q_3 & \vdash & \text{ƀƀ}q_4\mathtt{1} & \vdash & \text{ƀ}q_5 & \vdash & \text{ƀƀ}q_0 & \vdash & \text{ƀƀƀ}q_6.
\end{array}$$

A typical rejecting computation of M_{palin} is the following, where the input is the non-palindrome $\mathtt{0111}$:

$$q_0\mathtt{0111} \vdash \text{ƀ}q_1\mathtt{111} \vdash \text{ƀ}\mathtt{1}q_1\mathtt{11} \vdash \text{ƀ}\mathtt{11}q_1\mathtt{1} \vdash \text{ƀ}\mathtt{111}q_1 \vdash \text{ƀ}\mathtt{11}q_2\mathtt{1}.$$

EXERCISES (i) Present the transition function of M_{palin} as a directed multigraph. (ii) Follow the computation of M_{palin} on a palindrome of odd length and a non-palindrome of odd length. (iii) Give an informal inductive argument that $L(M_{\text{palin}})$ is the language of all palindromes over the alphabet $\{\mathtt{0}, \mathtt{1}\}$.

We have so far viewed Turing machines as accepters of languages over the alphabet Σ or, equivalently, as computers of predicates on Σ^*. We can also use Turing machines to compute more general functions, from Σ^* to Σ^*. To do this we merely regard whatever appears on the tape when the machine halts as the result of the computation. A Turing machine which is used in this mode is called a *transducer*. The function computed by a transducer is more properly a *partial* function since the transducer may not halt on all inputs. (Recall that, as we saw in §1.3, the possibility of non-termination is an inherent feature of any general method of computing.)

[10] In a *multi*graph, several edges may share the same start and end vertices.

EXAMPLE Consider the machine M_{div3} with $Q = \{q_0, q_1, q_2, q_3\}$, $q_I = q_0$, $q_F = q_3$, $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, $\Gamma = \{\mathtt{0}, \mathtt{1}, \text{ƀ}\}$, and with transition function specified by the set of quintuples

$$\begin{array}{lll} \{\ (q_0, \text{ƀ}, q_3, \text{ƀ}, R), & (q_0, \mathtt{0}, q_0, \mathtt{0}, R), & (q_0, \mathtt{1}, q_1, \mathtt{0}, R), \\ \ (q_1, \text{ƀ}, q_3, \text{ƀ}, R), & (q_1, \mathtt{0}, q_2, \mathtt{0}, R), & (q_1, \mathtt{1}, q_0, \mathtt{1}, R), \\ \ (q_2, \text{ƀ}, q_3, \text{ƀ}, R), & (q_2, \mathtt{0}, q_1, \mathtt{1}, R), & (q_2, \mathtt{1}, q_2, \mathtt{1}, R)\ \}. \end{array}$$

On input $x \in \{\mathtt{0}, \mathtt{1}\}^*$, the machine M_{div3} computes $y = \lfloor x/3 \rfloor$, where x and y are both interpreted as binary integers. A typical computation of M_{div3} is the following, where the input is `10100`, i.e., 20 in binary:

$$\begin{array}{lllllll} q_0\mathtt{10100} & \vdash & \mathtt{0}q_1\mathtt{0100} & \vdash & \mathtt{00}q_2\mathtt{100} & \vdash & \\ \mathtt{001}q_2\mathtt{00} & \vdash & \mathtt{0011}q_1\mathtt{0} & \vdash & \mathtt{00110}q_2 & \vdash & \mathtt{00110}\text{ƀ}q_3. \end{array}$$

EXERCISE Provide an interpretation for the states q_0, q_1, and q_2.

Finally, the machine which is described in the final paragraph of the quotation of NOTE 2 is very close to the Turing machine defined here. The main difference is that the machine described by Turing can observe B squares simultaneously (i.e., has B heads) and can move each head through L squares in a single move. Clearly we have simplified the definitions by setting L and B to be 1. (The machine has one head which can move one square in a single move.) We shall argue later that no loss of generality attends this simplification: a machine with one head moving one square at a time can do the work of a machine with B heads moving L squares at a time (albeit more slowly).