§4. Bells and whistles. In defining a formal model of computation we inevitably make a number of essentially arbitrary design decisions. These decisions should not affect the class of languages which can be recognized within the model. One way of gaining confidence in a particular model is to demonstrate that extending the model in various ways does not add to the class of languages which can be recognized. We consider here four simple extensions to the Turing machine of Note 3, and show that none of these extensions adds extra power to the model in this sense. In carrying out this programme, we encounter the important technique of *simulation*: to show that two models have equivalent power, we demonstrate that the ostensibly more restricted model may simulate any computation within the more general model.

§4.1. Doubly infinite tape. A Turing machine, M, with doubly infinite tape is formally identical in its specification to the singly infinite version defined in Note 3. However a configuration of M is now of the form $\alpha q \beta$, where α is a sequence of tape symbols which is infinite to the left, and β is a sequence of tape symbols infinite to the right. (Formally, the elements of α are indexed by the negative integers: $\ldots, -3, -2, -1$.) The 'one step' relation $\vdash$ can be defined by analogy with the corresponding relation in Note 3. Observe that it is now impossible for the head to drop off the left hand end of the tape, as the tape no longer has a left hand end. All but a finite number of symbols in α and β are blanks, so, just as before, we can make the configurations of M finite by stripping away leading and trailing blanks. The initial configuration of M and the criterion for acceptance are unchanged from Note 3.

The generalization to a doubly infinite tape adds no extra power to our model of computation.

Lemma 4.1 *A language L is accepted by a Turing machine with doubly infinite tape if and only if L is accepted by a Turing machine with singly infinite tape.*

proof. The 'if' part is easy. Suppose L is accepted by a Turing machine $\widehat{M}$ with singly infinite tape. Then L is also accepted by the doubly infinite machine M that first writes a special symbol onto the tape just to the left of the input word and then behaves exactly as $\widehat{M}$. If the head of M ever re-encounters the special symbol, then M immediately halts and rejects.

Now we tackle the 'only if' part. Suppose L is accepted by the Turing machine $M = (Q, \Gamma, \Sigma, ß, q_I, q_F, \delta)$ with doubly infinite tape. We show how to construct a machine $\widehat{M} = (\widehat{Q}, \widehat{\Gamma}, \widehat{\Sigma}, \widehat{ß}, \widehat{q_I}, \widehat{q_F}, \widehat{\delta})$ with singly infinite tape which accepts L. Imagine that the squares of the doubly infinite tape of M are indexed by integers, with the input word starting at tape square 0. Fold the tape about the middle of square 0 so that squares i and $-i$, for all i, are brought into alignment. In essence the machine $\widehat{M}$ works by *simulating* the operation of M using the (singly infinite) folded tape.

Refining this idea, we can think of the tape of $\widehat{M}$ as having two *tracks*, the upper track representing the tape squares of M with negative index, and the lower track representing

$\cdots$	s_{-5}	s_{-4}	s_{-3}	s_{-2}	s_{-1}	s_0	s_1	s_2	s_3	s_4	s_5	$\cdots$

The tape of M

\$	s_{-1}	s_{-2}	s_{-3}	s_{-4}	s_{-5}	$\cdots$
s_0	s_1	s_2	s_3	s_4	s_5	$\cdots$

The tape of $\widehat{M}$

Figure 3: Simulation of a Turing machine with doubly infinite tape

those with positive index. The first tape square on the upper track contains a special symbol \$ which is not in the tape alphabet Γ of M. Figure 3 is intended to suggest the correspondence between the tapes of M and $\widehat{M}$. It should seem plausible that $\widehat{M}$ can be constructed so as to simulate the computation of M, move for move. Informally, the technique is the following. When the tape head of M is over a square with positive index i, the head of $\widehat{M}$ is over square i of *its* tape, reading from the lower track. When the tape head of M is over a square with negative index $-i$, the head of $\widehat{M}$ is also over square i, but this time reading from the upper track; in this mode, $\widehat{M}$ always moves its head in the *opposite* direction to the head of M.

On this occasion only, we shall flesh out the informal argument by providing a formal description of the simulating machine $\widehat{M}$. By doing this we shall conclusively demonstrate that a Turing machine with doubly infinite tape can be converted, in an entirely mechanical way, into an equivalent machine with only singly infinite tape. The states, tape alphabet and input alphabet of $\widehat{M}$ are as follows:

$$\begin{aligned}\widehat{Q} &= \{\widehat{q}_I, \widehat{q}_F\} \cup (Q \times \{0,1\}),\\ \widehat{\Gamma} &= \Gamma \times (\Gamma \cup \{\$\}),\\ \widehat{\Sigma} &= \Sigma \times \{\text{ƀ}\} \subset \widehat{\Gamma}.\end{aligned}$$

Aside from the initial and final states (which are special) the states of $\widehat{M}$ are pairs of the form $\langle q, 0\rangle$ or $\langle q, 1\rangle$, where q is a state of the simulated machine M. The interpretation of state $\langle q, 0\rangle$ (respectively, state $\langle q, 1\rangle$) is that the simulated machine is in state q and is scanning a tape square with positive (respectively, negative) index; i.e., the simulating machine $\widehat{M}$ is reading from the lower (respectively, upper) track of its tape. The tape symbols of $\widehat{M}$ are pairs of the form $\langle s_0, s_1\rangle$, where $s_0 \in \Gamma$ is to be interpreted as the symbol on the lower track of $\widehat{M}$'s tape, and s_1 as the symbol on the upper track. The input symbols are elements of $\widehat{\Gamma}$ in which the second component of the pair is the blank symbol. Note that there is an obvious correspondence between elements of $\widehat{\Sigma}$ and Σ[11]. Naturally

[11]Strictly speaking the machine $\widehat{M}$ does not accept the same language as M but rather the language $\{\langle x, \text{ƀ}\rangle \mid x \in L\}$. However we could extend the alhabet of $\widehat{M}$ by throwing in the symbols of Σ and adding a pre-processing phase to $\widehat{M}$ that checks the input string is from Σ^* and converts it to the 'paired' form required; after that the machine behaves as described in the proof.

enough, the blank symbol of $\widehat{M}$ is the pair $\widehat{b} = \langle b, b \rangle$.

Finally, the transition function $\widehat{\delta}$ of $\widehat{M}$ is defined by rules (1)–(5) below. The ordering of rules is significant: the earlier rules take precedence over the later ones. (There is a parallel here with function definition by patterns in the programming language ML.)

(1) Transitions from the initial state:

$$\widehat{\delta}(\widehat{q}_I, \langle s, b \rangle) = \begin{cases} (\langle q', 0 \rangle, \langle s', \$ \rangle, R), & \text{if } \delta(q_I, s) = (q', s', R); \\ (\langle q', 1 \rangle, \langle s', \$ \rangle, R), & \text{if } \delta(q_I, s) = (q', s', L). \end{cases}$$

[The special symbol \$ is written to mark the end of the tape while, simultaneously, the first move of M is simulated.]

(2) Transitions to the final state:

$$\widehat{\delta}(\langle q_F, \cdot \rangle, \langle s_0, s_1 \rangle) = (\widehat{q}_F, \langle s_0, s_1 \rangle, R).$$

[If M enters its accepting state, then $\widehat{M}$ enters its accepting state on the following move.]

(3) Transitions when M is scanning square 0:

$$\widehat{\delta}(\langle q, \cdot \rangle, \langle s, \$ \rangle) = \begin{cases} (\langle q', 0 \rangle, \langle s', \$ \rangle, R), & \text{if } \delta(q, s) = (q', s', R); \\ (\langle q', 1 \rangle, \langle s', \$ \rangle, R), & \text{if } \delta(q, s) = (q', s', L). \end{cases}$$

[If the head of M moves right the simulation is continued on the lower track, otherwise on the upper track.]

(4) Transitions when M is scanning a square with positive index:

$$\widehat{\delta}(\langle q, 0 \rangle, \langle s_0, s_1 \rangle) = \begin{cases} (\langle q', 0 \rangle, \langle s_0', s_1 \rangle, L), & \text{if } \delta(q, s_0) = (q', s_0', L); \\ (\langle q', 0 \rangle, \langle s_0', s_1 \rangle, R), & \text{if } \delta(q, s_0) = (q', s_0', R). \end{cases}$$

(5) Transitions when M is scanning a square with negative index:

$$\widehat{\delta}(\langle q, 1 \rangle, \langle s_0, s_1 \rangle) = \begin{cases} (\langle q', 1 \rangle, \langle s_0, s_1' \rangle, R), & \text{if } \delta(q, s_1) = (q', s_1', L); \\ (\langle q', 1 \rangle, \langle s_0, s_1' \rangle, L), & \text{if } \delta(q, s_1) = (q', s_1', R). \end{cases}$$

[Note that $\widehat{M}$ must move its head in the opposite direction to that of M.]

This completes the formal description of $\widehat{M}$. Note that the construction of $\widehat{M}$ from M is a purely mechanical process. □

Most people will agree that specifying simulations at this level of detail is quite tedious. Although it is instructive to carry out one or two proofs with this formality, little is gained from repeating the exercise many times. Indeed the formalism merely serves to obscure

the idea of the proof. In future, then, we shall present proofs more informally; our growing intuition about Turing machines will assure us that the details could, in principle, be filled in on request.

§4.2. Several tapes. A k-tape Turing machine has a finite control and k heads, each scanning a separate (singly infinite) tape. In a single transition, the machine performs the following sequence of actions, depending on the current state of the finite control, and on the k tape symbols scanned by the tape heads:

1. the finite control moves to a new state;
2. each tape head prints a new symbol on the tape square it currently scans;
3. each tape head moves (independently) one square left or right.

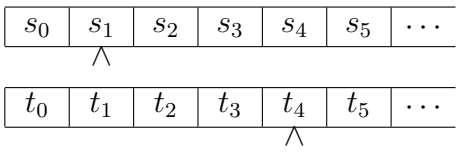
Initially, the input appears left justified on the first tape, and the other tapes are all blank. All tape heads are positioned at the leftmost square of each tape. A formal definition could be provided for multiple-tape Turing machines, but we shall not do so here.

LEMMA 4.2 *If a language L is accepted by a k-tape Turing machine for some k, then it is accepted by a single-tape Turing machine.*

PROOF. Let M be a k-tape Turing machine which accepts the language L. We shall construct a single tape Turing machine $\widehat{M}$ which simulates the operation of M. We shall imagine that $\widehat{M}$'s tape is divided into $2k$ tracks, two tracks for each tape of M. One track is used to record the contents of the corresponding tape of M. The other track is blank except that one square is marked by the special symbol $\wedge$; this *head marker* points out the square which is being scanned by the corresponding tape head of M. The first action of $\widehat{M}$ is to print a composite symbol, containing k of these head markers, onto the first square of its tape. [Questions: formally, what is the tape alphabet of $\widehat{M}$? the input alphabet? the blank symbol?] Figure 4 is intended to convey the arrangement of the single tape of $\widehat{M}$ and its correspondence with the k tapes of M.

In simulating a single move of M, the machine $\widehat{M}$ has to perform a sequence of moves. The current state of M is always recorded in $\widehat{M}$'s finite control. Each move of M is simulated by performing a sweep from the leftmost head marker to the rightmost, followed by a sweep from the rightmost head marker to the leftmost. In the left to right sweep, $\widehat{M}$ makes a note, in its finite control, of all k symbols scanned by the simulated machine M. By keeping a note of the number of head markers encountered during the sweep (again in the finite control) the sweep can be terminated as soon as the rightmost head marker has been reached.

The machine $\widehat{M}$ now has all the information necessary to simulate a single step of M; this it does during the right to left sweep. As each head marker is encountered, $\widehat{M}$ alters the tape symbol corresponding to that marker, and moves the marker left or right as appropriate. Again, by counting the head markers as they are encountered, the sweep can be terminated as soon as the leftmost marker has been reached. Finally, $\widehat{M}$ moves to a new state to reflect the new state of M. If the new state of M is an accepting state then $\widehat{M}$ accepts. □



The tapes of M

s_0	s_1	s_2	s_3	s_4	s_5	$\cdots$
$\flat$	$\wedge$	$\flat$	$\flat$	$\flat$	$\flat$	$\cdots$
t_0	t_1	t_2	t_3	t_4	t_5	$\cdots$
$\flat$	$\flat$	$\flat$	$\flat$	$\wedge$	$\flat$	$\cdots$

The tape of $\widehat{M}$

Figure 4: Simulation of a 2-tape machine

Lemma 4.2 will prove useful to us later. It is often easier to construct a Turing machine to perform a certain task if we allow ourselves to use a number of *work tapes*, each with a specified function. Lemma 2 assures us that any machine we construct in this way could, in principle, be transformed into a one-tape machine.

§4.3. Other variants. A two-dimensional Turing machine is one which has a two-dimensional array or *page* of squares, in place of the one-dimensional array of squares we have been calling a *tape*. The page has has a definite top-left corner, but is unbounded below and to the right. The transitions of the two-dimensional machine are similar to those of the conventional one-dimensional machine, except that the tape head can now move one square up, down, left, or right at each step, rather like a *crippled*[12] king in chess. The two-dimensional machine corresponds more closely to hand computation using a sheet of paper; however, it has no greater power than a one-dimensional machine:

Lemma 4.3 *If a language L is accepted by a two-dimensional Turing machine, then L is accepted by a one-dimensional Turing machine.*

Again the proof, which is omitted, involves simulating the computation of an arbitrary two-dimensional machine on a one-dimensional machine.

Finally, we might attempt to increase the power of the basic model by adding extra tape heads. This mirrors the ability of a human beings to shuffle the pages they are working on, and observe different symbols in widely separated parts of the manuscript simultaneously, or at least in quick succession. A k-head Turing machine is similar to the basic model except that it is equipped with k heads each moving independently of each other. Using a simulation similar to that employed in the proof of Lemma 4.2, it is not difficult to show the following:

[12]In chess, a king can also move along the diagonals.

LEMMA 4.4 *If a language L is accepted by a k-head Turing machine, then L is accepted by a one-head Turing machine.*

It is reassuring that the model we have chosen to work with is *robust*, i.e., is not affected by minor modifications to the definition. Indeed, we should have to discard the model if this turned out not to be the case.