tape head never moves left. Suppose also that on the output tape, $M$ writes strings over some alphabet $\Sigma$, separated by a marker symbol $\#$.

We can define $G(M)$, the language generated by $M$, to be the set of $w$ in $\Sigma^*$ such that $w$ is eventually printed between a pair of $\#$'s on the output tape.

Note that unless $M$ runs forever, $G(M)$ is finite. Also, we do not require that words be generated in any particular order, or that any particular word be generated only once. If $L$ is $G(M)$ for some TM $M$, then $L$ is an r.e. set, and conversely.

The recursive sets also have a characterization in terms of generators; they are exactly the languages whose words can be generated in order of increasing size. These equivalences will be proved in turn.

Characterization of r.e. sets by generators

Lemma 7.1 If $L$ is $G(M_1)$ for some TM $M_1$, then $L$ is an r.e. set.

Proof Construct TM $M_2$ with one more tape than $M_1$. $M_2$ simulates $M_1$ using all but $M_2$'s input tape. Whenever $M_1$ prints $\#$ on its output tape, $M_2$ compares its input with the word just generated. If they are the same, $M_2$ accepts; otherwise $M_2$ continues to simulate $M_1$. Clearly $M_2$ accepts an input $x$ if and only if $x$ is in $G(M_1)$. Thus $L(M_2) = G(M_1)$. \qed

The converse of Lemma 7.1 is somewhat more difficult. Suppose $M_1$ is a recognizer for some r.e. set $L \subseteq \Sigma^*$. Our first (and unsuccessful) attempt at designing a generator for $L$ might be to generate the words in $\Sigma^*$ in some order $w_1, w_2, \ldots$, run $M_1$ on $w_1$, and if $M_1$ accepts, generate $w_2$. Then run $M_1$ on $w_2$, generating $w_2$ if $M_1$ accepts, and so on. This method works if $M_1$ is guaranteed to halt on all inputs. However, as we shall see in Chapter 8, there are languages $L$ that are r.e. but not recursive. If such is the case, we must contend with the possibility that $M_1$ never halts on some $w_i$. Then $M_1$ never considers $w_i$, $w_{i+1}, w_{i+2}, \ldots$, and so cannot generate any of these words, even if $M_1$ accepts them.

We must therefore avoid simulating $M_1$ indefinitely on any one word. To do this we fix an order for enumerating words in $\Sigma^*$. Next we develop a method of generating all pairs $(i, j)$ of positive integers. The simulation proceeds by generating a pair $(i, j)$ and then simulating $M_1$ on the $i$th word for $j$ steps.

We fix a canonical order for $\Sigma^*$ as follows. List words in order of size, with words of the same size in "numerical order." That is, let $\Sigma = \{a_0, a_1, \ldots, a_k\}$, and imagine that $a_i$ is the "digit" $i$ in base $k$. Then the words of length $n$ are the numbers $0$ through $k^n - 1$ written in base $k$. The design of a TM to generate words in canonical order is not hard, and we leave it as an exercise.

Example 7.9 If $\Sigma = \{0, 1\}$, the canonical order is $\epsilon$, 0, 1, 00, 01, 10, 11, 000, 001, \ldots.

Next consider generating pairs $(i, j)$ such that each pair is generated after some finite amount of time. This task is not so easy as it seems. The naive approach, $(1, 1), (1, 2), (1, 3), \ldots \ldots$ never generates any pairs with $i > 1$. Instead, we shall generate pairs in order of the sum $i + j$, and among pairs of equal sum, in order of increasing $i$. That is, we generate $(1, 1), (1, 2), (2, 1), (1, 3), (2, 2), (3, 1), (1, 4), \ldots$. The pair $(i, j)$ is the $\left\lfloor \frac{(i + j - 1)(i + j - 2)}{2} + 1 \right\rfloor$th pair generated. Thus this ordering has the desired property that there is a finite time at which any particular pair $(i, j)$ is generated.

A TM generating pairs $(i, j)$ in this order in binary is easy to design, and we leave its construction to the reader. We shall refer to such a TM as the pair generator in the future. Incidentally, the ordering used by the pair generator demonstrates that pairs of integers can be put into one-to-one correspondence with the integers themselves, a seemingly paradoxical result that was discovered by Georg Cantor when he showed that the rationals (which are really the ratios of two integers) are equinumerous with the integers.

Theorem 7.7 A language is r.e. if and only if it is $G(M_2)$ for some TM $M_2$.

Proof With Lemma 7.1 we have only to show how an r.e. set $L = L(M_1)$ can be generated by a TM $M_2$. $M_2$ simulates the pair generator. When $(i, j)$ is generated, $M_2$ produces the $i$th word $w_i$ in canonical order and simulates $M_1$ on $w_i$ for $j$ steps. If $M_1$ accepts on the $j$th step (counting the initial $\epsilon$ as step 1), then $M_2$ generates $w_i$.

Surely $M_2$ generates no word not in $L$. If $w$ is in $L$, let $w$ be the $i$th word in canonical order for the alphabet of $L$, and let $M_1$ accept $w$ after exactly $j$ moves. As it takes only a finite amount of time for $M_2$ to generate any particular word in canonical order or to simulate $M_1$ for any particular number of steps, we know that $M_2$ will eventually produce the pair $(i, j)$. At that stage, $w$ will be generated by $M_2$. Thus $G(M_2) = L$. \qed

Corollary If $L$ is an r.e. set, then there is a generator for $L$ that enumerates each word in $L$ exactly once.

Proof $M_2$ described above has that property, since it generates $w_i$ only when considering the pair $(i, j)$, where $j$ is exactly the number of steps taken by $M_1$ to accept $w_i$. \qed

Characterization of recursive sets by generators

We shall now show that the recursive sets are precisely those sets whose words can be generated in canonical order.