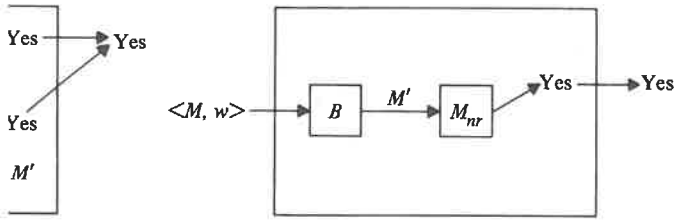


(b)

ıctions used in proof that L_{nr} is not r.e. (a) M'. (b) TM for $\bar{L}_u$.

cursive language if and only if M accepts w. M', which B must Fig. 8.10(a), and a TM to accept $\bar{L}_u$ given B and M_{nr}, is shown TM of Fig. 8.10(b) accepts $\langle M, w\rangle$ if and only if $L(M')$ is not ently, if and only if M does not accept w. That is, the TM ıd only if $\langle M, w\rangle$ is in $\bar{L}_u$. Since we have already shown that he assumption that M_{nr} exists is false. We conclude that L_{nr} is

---

ecursive index sets

show that we cannot decide if the set accepted by a Turing recursive. The technique of proof can also be used to show e if the set accepted is finite, infinite, regular, context free, has trings, or satisfies many other predicates. What then can we accepted by a TM? Only the trivial predicates, such as "Does . set?," which are either true for all TM's or false for all TM's. we shall discuss languages that represent properties of r.e. he languages are sets of TM codes such that membership of ge depends only on $L(M)$, not on M itself. Later we shall f TM codes that depend on the TM itself, such as "M has 27 e satisfied for some but not all of the TM's accepting a given

)f r.e. languages, each a subset of $(0 + 1)^*$. $\mathcal{S}$ is said to be a *nguages*. A set L has property $\mathcal{S}$ if L is an element of $\mathcal{S}$. For y of being infinite is $\{L \mid L \text{ is infinite}\}$. $\mathcal{S}$ is a *trivial* property if nsists of all r.e. languages. Let $L_{\mathcal{S}}$ be the set $\{\langle M\rangle \mid L(M)$ is

Theorem) Any nontrivial property $\mathcal{S}$ of the r.e. languages is

of generality assume that $\varnothing$ is not in $\mathcal{S}$ (otherwise consider ıtrivial, there exists L with property $\mathcal{S}$. Let M_L be a TM

accepting L. Suppose $\mathcal{S}$ were decidable. Then there exists an algorithm $M_{\mathcal{S}}$ accepting $L_{\mathcal{S}}$. We use M_L and $M_{\mathcal{S}}$ to construct an algorithm for L_u as follows. First construct an algorithm A that takes $\langle M, w\rangle$ as input and produces $\langle M'\rangle$ as output, where $L(M')$ is in $\mathcal{S}$ if and only if M accepts w ($\langle M, w\rangle$ is in L_u).

The design of M' is shown in Fig. 8.11. First M' ignores its input and simulates M on w. If M does not accept w, then M' does not accept x. If M accepts w, then M' simulates M_L on x, accepting x if and only if M_L accepts x. Thus M' either accepts $\varnothing$ or L depending on whether M accepts w.

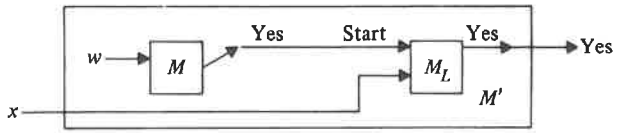


Fig. 8.11 M' used in Rice's theorem.

We may use the hypothetical $M_{\mathcal{S}}$ to determine if $L(M')$ is in $\mathcal{S}$. Since $L(M')$ is in $\mathcal{S}$ if and only if $\langle M, w\rangle$ is in L_u, we have an algorithm for recognizing L_u, a contradiction. Thus $\mathcal{S}$ must be undecidable. Note how this proof generalizes Example 8.2. □

Theorem 8.6 has a great variety of consequences, some of which are summarized in the following corollary.

Corollary The following properties of r.e. sets are not decidable:

a) emptiness,
b) finiteness,
c) regularity,
d) context-freedom.

Rice's Theorem for recursively enumerable index sets

The condition under which a set $L_{\mathcal{S}}$ is r.e. is far more complicated. We shall show that $L_{\mathcal{S}}$ is r.e. if and only if $\mathcal{S}$ satisfies the following three conditions.

1) If L is in $\mathcal{S}$ and $L \subseteq L'$, for some r.e. L', then L' is in $\mathcal{S}$ (the *containment property*).
2) If L is an infinite language in $\mathcal{S}$, then there is a finite subset of L in $\mathcal{S}$.
3) The set of finite languages in $\mathcal{S}$ is *enumerable*, in the sense that there is a Turing machine that generates the (possibly) infinite string $\text{code}_1\#\text{code}_2\#\ldots$, where code_i is a code for the ith finite language in $\mathcal{S}$ (in

any order). The code for the finite language $\{w_1, w_2, \ldots, w_n\}$ is just $w_1, w_2, \ldots, w_n$.

We prove this characterization with a series of lemmas.

Lemma 8.2 If $\mathscr{S}$ does not have the containment property, then $L_{\mathscr{S}}$ is not r.e.

Proof We generalize the proof that L_{nr} is not r.e. Let L_1 be in $\mathscr{S}$, $L_1 \subseteq L_2$, and let L_2 not be in $\mathscr{S}$. [For the case where $\mathscr{S}$ was the nonrecursive sets, we chose $L_1 = L_u$ and $L_2 = (\mathbf{0} + \mathbf{1})^*$.] Construct algorithm A that takes as input $\langle M, w\rangle$ and produces as output TM M' with the behavior shown in Fig. 8.12, where M_1 and M_2 accept L_1 and L_2, respectively. If M accepts w, then M_2 is started, and M' accepts x whenever x is in either L_1 or L_2. If M does not accept w, then M_2 never starts, so M' accepts x if and only if x is in L_1. As $L_1 \subseteq L_2$,

$$L(M') = \begin{cases} L_2 & \text{if } M \text{ accepts } w, \\ L_1 & \text{if } M \text{ does not accept } w. \end{cases}$$

Thus $L(M')$ is in $\mathscr{S}$ if and only if M does not accept w.

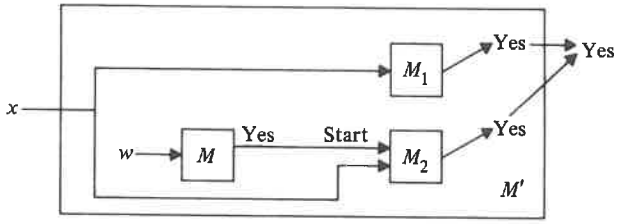
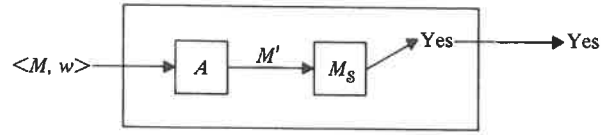


Fig. 8.12 The TM M'.

We again leave it to the reader to design the "compiler" A that takes $\langle M, w\rangle$ as input and connects them with the fixed Turing machines M_1 and M_2 to construct the M' shown in Fig. 8.12. Having constructed A, we can use a TM $M_{\mathscr{S}}$ for $L_{\mathscr{S}}$ to accept $\bar{L}_u$, as shown in Fig. 8.13. This TM accepts $\langle M, w\rangle$ if and only if M' accepts a language in $\mathscr{S}$, or equivalently, if and only if M does not accept w. As such a TM does not exist, we know $M_{\mathscr{S}}$ cannot exist, so $L_{\mathscr{S}}$ is not r.e. □

We now turn to the second property of recursively enumerable index sets.



Lemma 8.3 If $\mathscr{S}$ has an infinite language L such that no f
$\mathscr{S}$, then $L_{\mathscr{S}}$ is not r.e.

Proof Suppose $L_{\mathscr{S}}$ were r.e. We shall show that $\bar{L}_u$ wou
Let M_1 be a TM accepting L. Construct algorithm A to ta
input and produce as output a TM M' that accepts L if w
accepts some finite subset of L otherwise. As shown in Fig. 8
on its input x. If M_1 accepts x, then M' simulates M on w
fails to accept w after $|x|$ moves, then M' accepts x. We
algorithm A as an exercise.

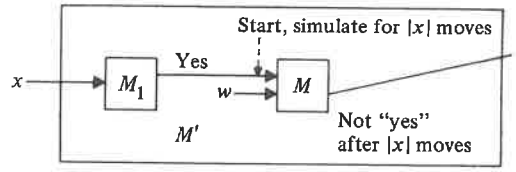


Fig. 8.14 Construction of M'.

If w is in $L(M)$, then M accepts w after some number o
$L(M') = \{x \mid x \text{ is in } L \text{ and } |x| < j\}$, which is a finite subset
$L(M)$, then $L(M') = L$. Hence, if M does not accept w, $L(M$
accepts w, $L(M')$, being a finite subset of L, is not in $\mathscr{S}$ by
lemma. An argument that is by now standard proves that if
Since the latter is not r.e., we conclude the former is not eit

Finally, consider the third property of r.e. index sets.

Lemma 8.4 If $L_{\mathscr{S}}$ is r.e., then the list of binary codes for tl
enumerable.

Proof We use the pair generator described in Section 7.7.
erated, we treat i as the binary code of a finite set, assumi
comma, 10 the code for zero, and 11 the code for one. We may
manner construct a TM $M^{(i)}$ (essentially a finite automaton)
the words in the finite language represented by i. We then s
ator for $L_{\mathscr{S}}$ for j steps. If it has printed $M^{(i)}$, we print the c
represented by i, that is, the binary representation of i itsel
limiter symbol #. In any event, after the simulation we retur
generator, which generates the pair following (i, j).

Theorem 8.7 $L_{\mathscr{S}}$ is r.e. if and only if

1) If L is in $\mathscr{S}$ and $L \subseteq L'$, for some r.e. L', then L' is in $\mathscr{S}$.
2) If L is an infinite set in $\mathscr{S}$, then there is some finite subse
3) The set of finite languages in $\mathscr{S}$ is enumerable.

The code for the finite language $\{w_1, w_2, \ldots, w_n\}$ is just w_1,

is characterization with a series of lemmas.

$^{\mathscr{S}}$ does not have the containment property, then $L_{\mathscr{S}}$ is not r.e.

alize the proof that L_{nr} is not r.e. Let L_1 be in $\mathscr{S}$, $L_1 \subseteq L_2$, and let
'or the case where $\mathscr{S}$ was the nonrecursive sets, we chose $L_1 = L_u$
*.] Construct algorithm A that takes as input $\langle M, w\rangle$ and pro-
TM M' with the behavior shown in Fig. 8.12, where M_1 and M_2
respectively. If M accepts w, then M_2 is started, and M' accepts
1 either L_1 or L_2. If M does not accept w, then M_2 never starts, so
ıd only if x is in L_1. As $L_1 \subseteq L_2$,

$$L(M') = \begin{cases} L_2 & \text{if } M \text{ accepts } w, \\ L_1 & \text{if } M \text{ does not accept } w. \end{cases}$$

$\mathscr{S}$ if and only if M does not accept w.

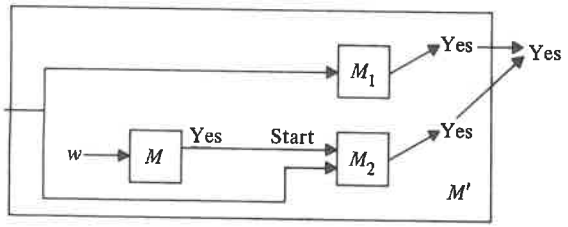


Fig. 8.12 The TM M'.

e it to the reader to design the "compiler" A that takes $\langle M, w\rangle$
ects them with the fixed Turing machines M_1 and M_2 to con-
vn in Fig. 8.12. Having constructed A, we can use a TM $M_{\mathscr{S}}$
$_u$, as shown in Fig. 8.13. This TM accepts $\langle M, w\rangle$ if and only
guage in $\mathscr{S}$, or equivalently, if and only if M does not accept w.
:s not exist, we know $M_{\mathscr{S}}$ cannot exist, so $L_{\mathscr{S}}$ is not r.e. □

to the second property of recursively enumerable index sets.

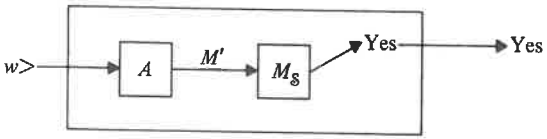


Fig. 8.13 Hypothetical TM to accept $\bar{L}_u$.

Lemma 8.3 If $\mathscr{S}$ has an infinite language L such that no finite subset of L is in $\mathscr{S}$, then $L_{\mathscr{S}}$ is not r.e.

Proof Suppose $L_{\mathscr{S}}$ were r.e. We shall show that $\bar{L}_u$ would be r.e. as follows. Let M_1 be a TM accepting L. Construct algorithm A to take a pair $\langle M, w\rangle$ as input and produce as output a TM M' that accepts L if w is not in $L(M)$ and accepts some finite subset of L otherwise. As shown in Fig. 8.14, M' simulates M_1 on its input x. If M_1 accepts x, then M' simulates M on w for $|x|$ moves. If M fails to accept w after $|x|$ moves, then M' accepts x. We leave the design of algorithm A as an exercise.

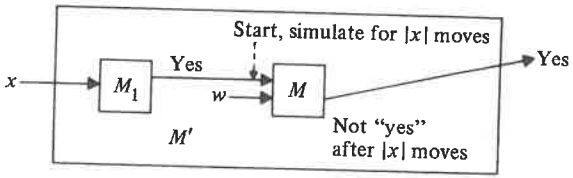


Fig. 8.14 Construction of M'.

If w is in $L(M)$, then M accepts w after some number of moves, say j. Then $L(M') = \{x \mid x \text{ is in } L \text{ and } |x| < j\}$, which is a finite subset of L. If w is not in $L(M)$, then $L(M') = L$. Hence, if M does not accept w, $L(M')$ is in $\mathscr{S}$, and if M accepts w, $L(M')$, being a finite subset of L, is not in $\mathscr{S}$ by the hypothesis of the lemma. An argument that is by now standard proves that if $L_{\mathscr{S}}$ is r.e., so is $\bar{L}_u$. Since the latter is not r.e., we conclude the former is not either. □

Finally, consider the third property of r.e. index sets.

Lemma 8.4 If $L_{\mathscr{S}}$ is r.e., then the list of binary codes for the finite sets in $\mathscr{S}$ is enumerable.

Proof We use the pair generator described in Section 7.7. When (i, j) is generated, we treat i as the binary code of a finite set, assuming 0 is the code for comma, 10 the code for zero, and 11 the code for one. We may in a straightforward manner construct a TM $M^{(i)}$ (essentially a finite automaton) that accepts exactly the words in the finite language represented by i. We then simulate the enumerator for $L_{\mathscr{S}}$ for j steps. If it has printed $M^{(i)}$, we print the code for the finite set represented by i, that is, the binary representation of i itself, followed by a delimiter symbol #. In any event, after the simulation we return control to the pair generator, which generates the pair following (i, j). □

Theorem 8.7 $L_{\mathscr{S}}$ is r.e. if and only if

1) If L is in $\mathscr{S}$ and $L \subseteq L'$, for some r.e. L', then L' is in $\mathscr{S}$.
2) If L is an infinite set in $\mathscr{S}$, then there is some finite subset L' of L that is in $\mathscr{S}$.
3) The set of finite languages in $\mathscr{S}$ is enumerable.

Proof The "only if" part is Lemmas 8.2, 8.3, and 8.4. For the "if" part, suppose (1), (2), and (3) hold. We construct a TM M_1 that recognizes $\langle M \rangle$ if and only if $L(M)$ is in $\mathscr{S}$ as follows. M_1 generates pairs (i, j) using the pair generator. In response to (i, j), M_1 simulates M_2, which is an enumerator of the finite sets in $\mathscr{S}$, for i steps. We know M_2 exists by condition (3). Let L_1 be the last set completely printed out by M_2. [If there is no set completely printed, generate the next (i, j) pair.] Then simulate M for j steps on each word in L_1. If M accepts all words in L_1, then M_1 accepts $\langle M \rangle$. If not, M_1 generates the next (i, j)-pair.

We use conditions (1) and (2) to show that $L(M_1) = L_{\mathscr{S}}$. Suppose L is in $L_{\mathscr{S}}$, and let M be any TM with $L(M) = L$. By condition (2), there is a finite $L' \subseteq L$ in $\mathscr{S}$ (take $L' = L$ if L is finite). Let L' be generated after i steps of M_2, and let j be the maximum number of steps taken by M to accept a word in L' (if $L' = \varnothing$, let $j = 1$). Then when M_1 generates (i, j), if not sooner, M_1 will accept $\langle M \rangle$.

Conversely, suppose M_1 accepts $\langle M \rangle$. Then there is some (i, j) such that within j steps M accepts every word in some finite language L' such that M_2 generates L' within its first i steps. Then L' is in $\mathscr{S}$, and $L' \subseteq L(M)$. By condition (1), $L(M)$ is in $\mathscr{S}$, so $\langle M \rangle$ is in $L_{\mathscr{S}}$. We conclude that $L(M_1) = L_{\mathscr{S}}$. □

Theorem 8.7 has a great variety of consequences. We summarize some of them as corollaries and leave others as exercises.

Corollary 1 The following properties of r.e. sets are not r.e.

a) $L = \varnothing$.

b) $L = \Sigma^*$.

c) L is recursive.

d) L is not recursive.

e) L is a *singleton* (has exactly one member).

f) L is a regular set.

g) $L - L_u \neq \varnothing$.

Proof In each case condition (1) is violated, except for (b), where (2) is violated, and (g), where (3) is violated. □

Corollary 2 The following properties of r.e. sets are r.e.

a) $L \neq \varnothing$.

b) L contains at least 10 members.

c) w is in L for some fixed word w.

d) $L \cap L_u \neq \varnothing$.

Problems about Turing machines

Does Theorem 8.6 say that everything about Turing machines i
The answer is no. That theorem has to do only with properties
accepted, not properties of the Turing machine itself. For exampl
"Does a given Turing machine have an even number of states?"
dable. When dealing with properties of Turing machines themselv
our ingenuity. We give two examples.

Example 8.4 It is undecidable if a Turing machine with alphabe
prints three consecutive 1's on its tape. For each Turing machine M
$\hat{M}_i$, which on blank tape simulates M_i on blank tape. However,
encode a 0 and 10 to encode a 1. If M_i's tape has a 0 in cell j, $\hat{M}_i$
$2j - 1$ and $2j$. If M_i changes a symbol, $\hat{M}_i$ changes the correspondi
the paired 0 to 1. One can easily design $\hat{M}_i$ so that $\hat{M}_i$ never has thr
1's on its tape. Now further modify $\hat{M}_i$ so that if M_i accepts, $\hat{M}$
consecutive 1's and halts. Thus $\hat{M}_i$ prints three consecutive 1's if a
accepts ϵ. By Theorem 8.6, it is undecidable whether a TM accep
predicate "ϵ is in L" is not trivial. Thus the question of whethe
Turing machine ever prints three consecutive 1's is undecidable.

Example 8.5 It is decidable whether a single-tape Turing machi
blank tape scans any cell four or more times. If the Turing machin
any cell four or more times, than every *crossing sequence* (sequen
which the boundary between cells is crossed, assuming states chan
head moves) is of length at most three. But there is a finite numb
crossing sequences of length three or less. Thus either the Turing r
within a fixed bounded number of tape cells, in which case finit
techniques answer the question, or some crossing sequence repeats
crossing sequence repeats, then the TM moves right with some easi
pattern, and the question is again decidable.

8.5 UNDECIDABILITY OF POST'S CORRESPONDENCE PROBLEM

Undecidable problems arise in a variety of areas. In the next three
explore some of the more interesting problems in language theory
techniques for proving particular problems undecidable. We begin
Correspondence Problem, it being a valuable tool in establishing oth
to be undecidable.

An instance of *Post's Correspondence Problem* (*PCP*) consists