CS481F01 Solutions 8

A. Demers

7 Dec 2001

1. Prob. 111 from p. 344 of the text. One of the following sets is r.e. and the other is not. Which is which?

(a) $\{\ i \mid L(M_i) \text{ contains at least 481 elements } \}$
(b) $\{\ i \mid L(M_i) \text{ contains at most 481 elements } \}$

Prove your answers.

Answer (a) This set is r.e. One proof: recall that the "membership" set

$$L_{\text{mbr}} \quad = \quad \{\ \langle j, k\rangle \mid M_j(k)\downarrow_a\ \}$$

is r.e. To test whether input i is in L_a), a machine can simply enumerate the elements of L_{mbr}, and accept after enumerating the 481st pair of the form $\langle i, k\rangle$.

Answer (b) This set is not r.e. Recall the "diagonal divergence" set

$$L_{\uparrow d} \quad = \quad \{\ j \mid M_j(j)\uparrow\ \}$$

This set is known not to be r.e., but we can easily reduce it to L_b. Given i, construct machine M_j such that $M_j(x)$ just simulates $M_i(i)$ until it halts. $M_j(x)$ accepts if $M_i(i)$; it loops otherwise. It is clear that computing j from i is a total TM-computable function. If $M_i(i)$ diverges, then $L(M_j)$ is empty, so j is in L_b. If $M_i(i)$ halts, then $L(M_j)$ is Σ^*, so j is not in L_b. Thus, we have shown

$$L_{\uparrow d} \quad \leq_m \quad L_b$$

so L_b cannot be r.e.

2. Suppose P is any property of *pairs* of r.e. sets. We define

$$L_P \quad = \quad \{ \ \langle i,j \rangle \mid P(L(M_i), L(M_j)) \ \}$$

We say such a property is *nontrivial* if it is neither identically true nor identically false; i.e.,

$$P \text{ nontrivial} \quad \Leftrightarrow \quad (\exists \langle i,j \rangle \in L_P) \ \wedge \ (\exists \langle i,j \rangle \notin L_P)$$

Prove the following extension of Rice's Theorem:

> No nontrivial property of pairs of r.e. sets is decidable.

Answer Following the Hint, we note that P is decidable iff $\neg P$ is decidable. Thus, we can assume without loss of generality that

$$P(\emptyset, \emptyset) \ = \ \textbf{false}$$

since this must be true of either P or $\neg P$, and proving either of these sets undecidable is equivalent.

Since we assume P is nontrivial, there must exist p and q such that

$$P(L(M_p, L(M_q)) \ = \ \textbf{true}$$

We reduce the halting problem to L_P as follows.

Given input $\langle i,j \rangle$, construct a machine M_m which, on input x, will

- Simulate $M_i(j)$ until it halts; then
- Simulate $M_p(x)$

Clearly if $M_i(j)$ halts then $L(M_m)$ will be exactly $L(M_p)$; but if $M_i(j)$ loops then $L(M_m)$ will be empty.

Analogously, given input $\langle i,j \rangle$, we can construct a machine M_n which, on input x, will

- Simulate $M_i(j)$ until it halts; then
- Simulate $M_q(x)$

As above, if $M_i(j)$ halts then $L(M_n)$ will be exactly $L(M_q)$; but if $M_i(j)$ loops then $L(M_n)$ will be empty.

Both the above constructions are total TM-computable functions. Thus, from the pair $\langle i, j\rangle$ a TM can compute a pair $\langle n, m\rangle$. Following the above argument, if $M_i(j)$ halts then

$$P(L(M_m), L(M_n)) \quad = \quad P(L(M_p), L(M_q)) \quad = \quad \textbf{true}$$

but if $M_i(j)$ loops

$$P(L(M_m), L(M_n)) \quad = \quad P(\emptyset, \emptyset) \quad = \quad \textbf{false}$$

This yields the reduction

$$L_{\text{mbr}} \ \leq_m \ L_P$$

so P cannot be decidable, as required.

3. Let L and L' denote CFLs (presented as CFGs), and let R denote a regular set (presented as a regular expression or right-linear grammar). Which of the following are decidable and which undecidable?

(a)	$L \ = \ R$
(b)	$L \ \subseteq \ R$
(c)	$L \ \supseteq \ R$
(d)	$L \ = \ L'$
(e)	$L \ \subseteq \ L'$
(f)	$L \ \supseteq \ L'$
(g)	$L \ = \ L\,L$

Prove your answers.

Answer (a) Assume we're enumerating the regular sets by right-linear grammars, and let R_i denote the i^{th} right-linear grammar in the enumeration. Now part (a) just asks whether the set

$$L_a \quad = \quad \{\langle i, j\rangle \mid L(G_i) = L(R_j)\ \}$$

is recursive.

Recall the set

$$\{ \ i \mid L(G_i) = \Sigma^* \ \}$$

is not recursive. For convenience, call this set L_u. We can reduce L_u to L_a almost trivially. Choose some n such that

$$L(R_n) \ = \ \Sigma^*$$

Then

$$L_u \ = \ \{ \ i \mid L(G_i) = L(R_n) \ \} \ \leq_m \ \{\langle i, j\rangle \mid L(G_i) = L(R_j) \ \}$$

The reduction is the simple function

$$i \ \mapsto \ \langle i, n\rangle$$

Answer (b) This one is decidable. Note

$$L \ \subseteq \ R \ \ \Leftrightarrow \ \ L \ \cap \ \overline{R} \ = \ \emptyset$$

We showed in lecture that CFLs are closed under intersection with regular sets. We also showed that it is decidable whether a CFG generates an empty language; i.e., the set

$$L_\emptyset \ = \ \{ \ i \mid L(G_i) = \emptyset \ \}$$

is recursive. To reduce L_b to $L_\emptyset$, given $\langle i, j\rangle$ we construct a CFG G_k such that

$$L(G_k) \ \ = \ \ L(G) \ \cap \ \overline{L(R_j)}$$

then ask whether k is in $L_\emptyset$.

Answer (c) Observe that, for *any* L whatsoever,

$$L \ \supseteq \ \Sigma^* \ \ \Leftrightarrow \ \ L \ = \ \Sigma^*$$

so L_c is not recursive by the same argument as for part (a).

Answer (d) Since Σ^* is regular, it is also context-free, and L_d is not recursive by the same argument as for part (a).

Answer (e) Since

$$L \subseteq L' \quad \Leftrightarrow \quad \overline{L'} \subseteq \overline{L}$$

parts (e) and (f) necessarily have the same answer.

Answer (f) L_f is not recursive by the same argument as for part (c).

Answer (g) This one required some inspiration. Consider the language

$$\text{NVC}_{i,j} \quad = \quad \overline{\text{ValComps}_{M_i,j}}$$

We showed in lecture that this language is a CFL, and given $\langle i, j \rangle$ we can effectively construct a grammar for $\text{NVC}_{i,j}$. Observe that

$$\text{NVC}_{i,j} \ = \ \Sigma^* \quad \Leftrightarrow \quad j \notin L(M_i)$$

That is, $\text{NVC}_{i,j}$ is Σ^* if $M_i(j)$ rejects, and is something smaller otherwise. Now, claim

$$\text{NVC}_{i,j}\ \text{NVC}_{i,j} \quad = \quad \Sigma^*$$

in all cases, whether $M_i(j)$ accepts or rejects. To see this, note that the empty string is not a valid computation. Thus, any w can be written

$$\begin{array}{ccll} w & = & \epsilon\ w & w \in \text{NVC}_{i,j} \\ & \text{or} & & \\ w & = & w_1\ w_2 & w_1, w_2 \in \text{NVC}_{i,j} \ \text{ if } w \notin \text{NVC}_{i,j} \end{array}$$

In the second case, a valid computation w can always be expressed as the concatenation of two strings w_1 and w_2, neither of which is itself a valid computation.

Thus, the function that maps a pair $\langle i, j \rangle$ to an index k such that

$$L(G_k) \quad = \quad \text{NVC}_{i,j}$$

yields a reduction

$$\overline{L_{\text{mbr}}} \ \leq_m \ L_g \ = \ \{\ i \mid l(G_i) \ = \ L(G_i)L(G_i)\ \}$$

proving that L_g is not recursive.