CS481F01 Final Solutions

A. Demers

18 Dec 2001

The solutions here were typeset during the exam. I didn't quite finish, which suggests – as usual – the exam was longer than I intended it to be.

1. (20 points) *Post's Correspondence Problem* (PCP) is the following: You are given a finite collection of pairs of strings

$$\{\ \langle x_1, y_1\rangle,\ \langle x_2, y_2\rangle,\ \ldots,\ \langle x_n, y_n\rangle\ \}$$

and are asked whether

$$(\exists n, i_1, i_2, \ldots, i_n)(\ x_{i_1} x_{i_2} \ldots x_{i_n} \ = \ y_{i_1} y_{i_2} \ldots y_{i_n}\ )$$

that is, whether there is a way to chose a finite sequence of pairs (possibly with repetitions) so that corresponding strings concatenate to the same result.

For example, the instance

$$\langle 00, 0\rangle,\ \langle 10, 1\rangle,\ \langle 1, 0001\rangle$$

has solution

$$n = 4,\ i_1 = 2,\ i_2 = 1,\ i_3 = 1,\ i_4 = 3$$

since

$$x_2 x_1 x_1 x_3 \ = \ 10\ 00\ 00\ 1 \ = \ 1\ 0\ 0\ 0001 \ = \ y_2 y_1 y_1 y_3$$

while you can verify that the instance

$$\langle 10, 1\rangle,\ \langle 10, 01\rangle,\ \langle 1, 11\rangle$$

has no solution.

Prove PCP is not decidable.

Answer 1: PCP is a well known problem. Proofs of its undecidability "abound in the literature" – see, for example, Hopcroft and Ullman, *Introduction to Automata Theory, Languages and Computation*, p. 193 ff. (the page number may be wrong – I don't own the latest edition). This text was on reserve in the Engineering Library for this course.

A number of people tried to solve this problem using Rice's Theorem. I don't know a correct way to do this. The incorrect attempts were of the following form.

> Any instance of PCP can be represented as a string
>
> $$\sharp x_1 \sharp y_1 \sharp x_2 \sharp y_2 \ \ldots \ \sharp x_n \sharp y_n \sharp$$
>
> Let the property $P(L)$ be "L is $\{w\}$, where w is the encoding of an instance of PCP that has a solution." Clearly P is a nontrivial property of sets, so Rice's Theorem applies.

The problem is, Rice's Theorem doesn't tell us anything interesting about decidability of PCP. What Rice's Theorem tells us is

$$\{ \ i \mid P(L(M_i)) \ \} \ \text{ is not recursive.}$$

What we're interested in is

$$\begin{aligned} L_{PCP} &= \{ \ w \mid w \text{ encodes a PCP instance with a solution } \} \\ &= \{ \ w \mid P(\{w\}) \ \} \end{aligned}$$

Rice's Theorem *does not* imply that L_{PCP} is undecidable.

Consider the property $Q(L)$ given by "L is $\{w\}$, for some w such that length(w) is prime." Clearly this is a nontrivial property of sets, so by Rice's Theorem

$$\{ \ i \mid Q(L(M_i)) \ \} \ \text{ is not recursive.}$$

But just as clearlly

$$\{ \ w \mid Q(\{w\}) \ \} \quad \text{is recursive.}$$

It was an interesting idea, though.

2. (24 points – each part 2 points for answer, 4 points for justification) For this problem, define

$$M_i(x) \prec M_j(y)$$

if $M_i(x)$ halts in fewer steps than $M_j(y)$. We do not specify whether $M_i(x)$ accepts or rejects, and we allow the possibility that $M_j(y)$ *never* halts.

Consider the languages

$$\begin{array}{lrcl}
(a) & L_a & = & \{ \langle i,j,x\rangle \mid M_i(x) \prec M_j(x) \} \\
(b) & L_b & = & \{ \langle i,j\rangle \mid (\exists x)(\ M_i(x) \prec M_j(x)\ ) \} \\
(c) & L_c & = & \{ \langle i,j\rangle \mid (\forall x)(\ M_i(x) \prec M_j(x)\ ) \} \\
(d) & L_d & = & \{ \langle i,j\rangle \mid (\exists n \forall x)(\ (M_i(x) \prec M_j(x)) \ \Rightarrow\ (|x| \le n)\ ) \}
\end{array}$$

For each of these languages, tell where it sits in the Arithmetic Hierarchy; e.g.

Δ_1^0 (recursive)
Σ_1^0 (r.e. but not recursive)
Π_1^0 (co-r.e. but not recursive)
Δ_2^0
(etc.)

Justify your answers.

Answer a: This set is r.e., Σ_1^0. Given input $\langle i,j,x\rangle$ we can first simulate $M_i(x)$ until it halts. If it never halts, $M_i(x) \prec M_j(x)$ is necessarily false, so it's okay if we loop in this phase. If $M_i(x)$ halts after n steps, we then simulate $M_j(x)$ for up to $n+1$ steps. If $M_j(x)$ is still running after $n+1$ steps, we accept.

Answer b: Again, the set is r.e., Σ_1^0. On input $\langle i,j\rangle$ we enumerate all pairs $\langle x,n\rangle$. For each pair, if $M_i(x)$ halts within n steps and $M_j(x)$ does not, we accept.

Answer c: This set is Π_2^0. You can characterize it by

$$(\forall x)(\exists t)(\ M_i(x) \text{ halts in } t \text{ steps and } M_j(x) \text{ does not }\ )$$

showing that it is in Π_2^0. To show that it is *properly* in Π_2^0, observe you can reduce TOTAL (the set of total TM indices) to this language by letting M_j be a machine that loops on every input (and modifying M_i if necessary so it accepts if and only if it halts).

Answer d: By a similar argument, this language is Σ_2^0, As above, $\prec$ is basically a single existential. Rewrite

$$(\ (M_i(x) \prec M_j(x)) \ \Rightarrow \ (|x| \leq n)\ )$$

as

$$(\ \neg(M_i(x) \prec M_j(x)) \ \vee \ (|x| \leq n)\ )$$

so the existential is inside a negation, and becomes a universal. Now the specification of L_d is

$$(\exists n)(\forall x)(\forall t)(\ \ldots\ )$$

We can combine the pair of adjacent universals so the specification becomes

$$(\exists n)(\forall x, t)(\ \ldots\ )$$

putting L_d in Σ_2^0; Again, as above, you can reduce FINITE (the set of indices of TM's that recognize finite languages) to this set by choosing M_j to be an everywhere-looping machine.

3. (20 points) Let

$$F_1,\ F_2,\ F_3,\ \ldots$$

be an effective enumeration of the primitive recursive function definitions. Using it, describe a total TM-computable function that is *not* primitive recursive. Justify your answer.

Answer: As the hint suggests, this is just a diagonalization. Invoke Church's Thesis to argue that a TM can simulate a primitive recursive function definition. That is, the function

$$(i, n) \ \mapsto \ F_i(n)$$

is a total TM-computable function. Now construct a machine M which, given input n, computes

$$(n) \ \mapsto \ F_n(n) + 1$$

Clearly M is total, but the function it computes cannot be F_i for any i.

4. (25 points) Language L is said to be *bounded* if

$$(\exists k)(\exists w_1, w_2, \ldots, w_k)(\ L \ \subseteq \ w_1^* w_2^* \ldots w_k^* \ )$$

Define

$$N(L, m) \ = \ |\{ \ w \in L \mid \text{length}(w) \leq m \ \}|$$

that is, $N(L, m)$ is the number of strings in L of length at most m.

(a) (8 points) Show that if L is bounded then there exists a polynomial $p(m)$ such that $N(L, m) \leq p(m)$ (that is, $N(L, m)$ is bounded by some polynomial in m).

Answer a: Suppose

$$w \ = \ w_1^{i_1} w_2^{i_2} \ \ldots \ w_k^{i_k}$$

We can assume without loss of generality that for all j the string w_j is not the empty string – otherwise we could just leave it out of the specification, and the language would remain bounded by the remaining $k-1$ strings. In that case, each of i_1, …, i_k can be *at most* the length of w. Thus, there are fewer than $|w|^k$ possible choices for i_1 thru i_k, so $N(L, m) \leq m^k$ as desired.

(b) (9 points) Give examples of languages that are

(1) Regular but not bounded.
(2) Bounded and context-free but not regular; and
(3) Bounded but not context-free.

Justify your answers.

Answer b: For part (1), we can simply use Σ^*, since

$$N(\ \Sigma^*, \ m\ ) \ = \ |\Sigma|^m$$

which grows faster than any polynomial in m (provided Σ has at least two letters).

For part (2), our old friend

$$\{\ 0^n10^n \mid n \geq 0\ \}$$

is clearly bounded by $0^*1^*0^*$.

For part (3), we can choose a language over 0^* that is not ultimately periodic, and argue by Parikh's Theorem (I got the name right this time) that it cannot be context-free. Thus, a set like

$$\{\ 0^p \mid p \text{ is prime}\ \}$$

will do.

(c) (8 points) Given a TM description M, is it decidable whether $L(M)$ is bounded? Justify your answer.

Answer c: We have given examples above of r.e. sets that are bounded, and of r.e. sets that are not bounded. Thus, "boundedness" is a nontrivial property of the r.e. sets, so the result is immediate by Rice's Theorem.

(d) (20 points extra credit) (This is *not easy* – don't tackle it unless you have time left at the end!) Given a right-linear grammar R, is it decidable whether $L(R)$ is bounded? Justify.

Answer d: Assume wlog that the grammar has no useless nonterminals – every nonterminal is reachable from the start symbol and generates at least one terminal string. Also we'll assume the alphabet is $\{0, 1\}$.

Suppose there exists a nonterminal A and a pair of strings w and x such that

$$A \ \to^* \ 0wA \qquad \text{and} \qquad A \ \to^* \ 1xA$$

In this case, by part (a), $L(R)$ cannot be bounded, since

$$N(L(R), m) \ \geq \ 2^q \qquad\qquad q \ = \ m/(1 + \max(|w|, |x|))$$

which grows faster than any polynomial in m.

Suppose a pair of derivations like the above cannot exist. Then for any A and any pair of strings u and v,

$$(A \;\to^*\; uA \;\wedge\; A \;\to^*\; vA) \quad\Rightarrow\quad (u \;\prec\; v)$$

where we use $\prec$ to mean "is a prefix of" and we assume wlog that u is shorter than v.

Now choose any nonterminal A and let

$$g \;=\; \text{gcd}(\;\{\;\text{length}(z) \mid A \;\to^*\; zA\;\}\;)$$

Let

$$u_A \;=\; \text{the first } g \text{ symbols of } z \qquad \text{where } A \;\to^*\; zA$$

This is well-defined, since for any two such z one must be a prefix of the other. You can show that

$$(A \;\to^*\; zA) \quad\Rightarrow\quad z \;\in\; u_A^*$$

There is such a u_A for each nonterminal A.

Now, consider any derivation in R. It starts with S. It generates a string in u_S^* up to the *last* use of S in the derivation. It then generates either a 0 or a 1, and a new nonterminal A. It then generates a string in u_A^* until the last use of A. The derivation continues in this fashion, possibly for every nonterminal in the grammar. But no nonterminal is used more than once in this way. Eventually the derivation ends with a use of a rule of the form

$$B \;\to\; 0 \qquad \text{or} \qquad B \;\to\; 1$$

Suppose the nonterminals are A, B, $\ldots$, Z. The above argument shows the language must be bounded by

$$(\; u_A^* u_B^* \ldots u_Z^* 0^* 1^* \;)^n$$

where n is the number of nonterminals in the grammar. Note most of the uses of * are expanded 0 times.

To test whether $L(R)$ is bounded, it suffices to test the condition given above, that is, whether there is a nonterminal A and strings w and x such that

$$A \;\to^*\; 0wA \qquad \text{and} \qquad A \;\to^*\; 1xA$$

Since w and x can always be chosen to be no longer than the number of nonterminals, this property is decidable.

5. (54 points – each part 2 points for answer, 4 points for justification) For this question, we use the notation

$A, A_1, \ldots$	regular sets
$L, L_1, \ldots$	context-free languages
$D, D_1, \ldots$	deterministic CFLs
$M, \ldots$	Turing Machine descriptions

We use the symbol "$\sharp$" as a separator symbol not otherwise in any of the languages.

For each of the following sets, tell whether it is necessarily

(1) regular,
(2) a deterministic CFL,
(3) a CFL,
(4) co-CFL, the complement of a CFL,
(5) recursive,
(6) r.e. (i.e. Σ_1^0), or
(7) co-r.e (i.e. Π_1^0).

The sets are

(a) $AA = \{ xy \mid x \in A \wedge y \in A \}$
(b) $\{ x\sharp x \mid x \in A \}$
(c) $\{ x \mid x^r x x^r \in A \}$
(d) $\{ x \mid (\exists y)(xy \in A \wedge y \in L) \}$
(e) $\{ x \mid (\exists y)(x \in A \wedge xy \in L) \}$
(f) $D_1 \cap D_2$
(g) $L_1 \cap L_2$
(h) $\mathrm{ValComps}_{M,x}$
(i) $\mathrm{ValComps}_M = \bigcup_x \mathrm{ValComps}_{M,x}$

Justify your answers briefly.

Answers:

(a) – regular (1)
(b) – co-CFL (4)
(c) – regular (1)
(d) – regular (1)
(e) – CFL (3)

(f) – co-CFL (4)
(g) – recursive (5)
(h) – regular (1)
(i) – co-CFL (4)

Justifications: (a) From lecture, regular sets are closed under concatenation.

(b) We showed in lecture that

$$D \;=\; \{\; w\sharp w \mid w \in \Sigma^* \;\}$$

is co-CFL. Then the language L_b is simply

$$L_b \;=\; D \;\cap\; (A \cdot \{\sharp\} \cdot A)$$

so

$$\overline{L_b} \;=\; \overline{D \;\cap\; (A \cdot \{\sharp\} \cdot A)} \;=\; \overline{D} \;\cup\; \overline{(A \cdot \{\sharp\} \cdot A)}$$

This is the union of a CFL and a regular set, and thus is a CFL.

(c) A slick proof that this language is regular uses a 2-way DFA – remember those? We proved they recognize only regular languages. A 2-way DFA can recognize

$$\{\; x \mid x^r x x^r \in A \;\}$$

by first moving its head to the right end of the input, then doing three scans of the input tape: right-to-left, then left-to-right, then right-to-left again, while simulating a DFA that recognizes A.

(d) We proved this in a homework for the case where L is an *arbitrary* set.

(e) The specified language L_c is context-free: it is the set of prefixes of the intersection of a CFL with a regular set, and both these operations preserve CFLs. To show that L_c is not in general regular or a DCFL, it is sufficient to let A be Σ^* and L be some prefix-closed CFL that is not a DCFL; the set

$$L \;=\; \{0^i 1^j 2^k \mid i,j,k \geq 0 \;\wedge\; (i \geq j \vee igeqk) \;\}$$

is sufficient.

(f) Since DCFLs are closed under complement, we get

$$D_1 \;\cap\; D_2 \;\;=\;\; \overline{\overline{D_1} \;\cup\; \overline{D_2}}$$

Since the union of two DCFLs is in general a (nondeterministic) CFL, the result follows.

(g) We know it is undecidable whether the intersection of two CFLs is empty, but that does not tell us much about the complexity of the intersection – consider parts (h) and (i). Obviously a CFL is recursive, and the recursive sets are closed under intersection, so L_g is recursive. To show it is not a CFL, we can easily choose L_1 and L_2 so their intersection is

$$L_1 \;\cap\; L_2 \;=\; \{\; a^i b^i c^i \mid i \geq 0 \;\}$$

which is not a CFL. To show L_g is not co-CFL, let L_1 be Σ^* and L_2 be a CFL whose complement is not a CFL (for example, $\{w\sharp x \mid w \neq x\ \}$).

(h) Since $ValComps_{M,x}$ is either empty (if M does not accept x) or a single string (representing the accepting computation if M does accept x), it is always regular. It's just undecidable *what* regular set it is ...

(i) In lecture (and in the text) we proved $ValComps_{M,x}$ is co-CFL. The proof goes over almost completely unchanged for $ValComps_M$. For the version in the text, we replace condition (3) on p. 252 by "α_0 represents *some* start configuration of M."