

Solutions for Mid-Term
Course: CS 509
Foundations of Computer Science

Lecturer: Joe Kilian
Department of Computer Science
Rutgers, The State University of New Jersey

October 17, 2006

Problem 1. *For this problem, L_1 is an arbitrary context free language and L_2 is an arbitrary regular language. In class we learned that the intersection of a regular and a context free language is context free.*

- A. *Show that it is decidable whether a regular language contains all strings of the form $a^n b^n$.*
- B. *Show how to decide whether $L_1 \subseteq L_2$.*
- C. *Show that it is undecidable whether $L_2 \subseteq L_1$.*

Answer.

- A. Given a regular language L_2 , consider its complement $\overline{L_2}$ which is also a regular language. Let $L_{\text{test}} = \overline{L_2} \cap \{a^n b^n | n \in \mathbb{N}\}$ and it is context free. So $L_{\text{test}} = \phi$ if and only if L_2 contains all strings of form $a^n b^n$ and we could test the emptiness of L_{test} deterministically.
- B. As we did in part A, let $L_{\text{test}} = \{\overline{L_2} \cap L_1\}$. $L_{\text{test}} = \phi$ if and only if $L_1 \subseteq L_2$.
- C. Let $L_2 = \Sigma^*$. We know that ALLCFG, the special case, is undecidable, so is the above statement.

□

Problem 2. *Recalling Problem Set 2 (Sipser, 4.18): Let A and B be disjoint languages. We say that a language C separates A and B if $A \subseteq C$ and $B \subseteq \overline{C}$ (\overline{C} denotes the complement of C).*

- A. *Show that if A and B disjoint and either A or B is regular (or both), then there exists a regular language C that separates them.*
- B. *Give with proof an example of disjoint A and B such that*
- *A and B are context free.*
 - *Neither A nor B are regular.*
 - *There exists a regular language C that separates A and B .*
- C. *Give with proof an example of disjoint A and B such that*
- *A and B are context free.*
 - *There is no regular C that separates A and B .*

Answer.

- A. If A is regular, let $C = A$. If B is regular, let $C = \overline{B}$.
- B. Let $A = \{a^n b^n \mid n \in \mathbb{N}, n > 0\}$, $B = \{b^n c^n \mid n \in \mathbb{N}, n > 0\}$ and $C = a^* b^*$. So trivially $A \subseteq C$ and $B \subseteq \overline{C}$.
- C. Let $A = \{a^n b^m \mid n, m \in \mathbb{N}, n > m > 0\}$ and $A = \{a^n b^m \mid n, m \in \mathbb{N}, m \geq n > 0\}$. If such C exists, by its separation property, then $A = C \cap a^* b^* - \{\lambda\}$. Because regular languages are closed under intersection, if C is regular, so is A , which is apparently a contradiction.

□

Problem 3. *We consider the following two definition for a language L being decidable on a subset $S \subseteq \Sigma^*$:*

Definition 1. L is decidable on $S \subseteq \Sigma^*$ iff there exists a Turing machine M such that

- M halts on all input $x \in \Sigma^*$.
- For all $x \in S$, $x \in L$ iff M accepts on input x .

Definition 2. L is decidable on $S \subseteq \Sigma^*$ iff there exists a Turing machine M such that

- M halts on all input $x \in S$.
- For all $x \in S$, $x \in L$ iff M accepts on input x .

When are these definitions equivalent?

- A. Show that if S is decidable, then Definition 1 and 2 are equivalent.
- B. Give an example of a set S and language L that is decidable on S according to Definition 2 but not Definition 1

Answer.

- A. The direction from Definition 1 to 2 is trivial since $S \subseteq \Sigma^*$. For the other direction, if S is decidable, then there exists a decider M_S which halts on every input and accepts S . Combining it with M , we obtain a machine halting on all input $x \in \Sigma^*$ as follows: Run M_S on x first, if it accepts, then simulate M on x . Therefore according to our assumption in Definition 2, both of our conditions in Definition 1 are satisfied.
- B. (Due to Brian Thompson) Take S as the Halting problem and let $L = \{ \langle M \rangle \mid M \text{ halts with empty tape} \}$. Then L is decidable on H according to Definition 2, but not Definition 1. Otherwise it implies that there exists a Turing machine M' such that M' halts on every input and decide L correctly. Consider the machine M'' : run M' on $\langle M' \rangle$, if it halts with empty tape, write something on the tape; if not, erase everything on the tape. Therefore, $\langle M'' \rangle \in L$?

□

Problem 4. Let $<_l$ denote lexicographic ordering on strings: $a <_l b$ if $|a| < |b|$ (shorter strings come first) and equal-length strings are compared according to alphabetical order. Let L be an infinite recursively enumerable language and let M enumerate L . That is, M writes on its output tape the infinite sequence w_0, w_1, \dots , listing all the strings in L (every string in L is eventually output).

- A. Show that if w_0, w_1, \dots are in lexicographic order ($w_i <_l w_{i+1}$), then L is decidable.
- B. Show that there exists a Turing machine M' that outputs an infinite subset of L in lexicographic order. That is, it outputs an infinite sequence q_0, q_1, \dots , such that $q_i <_l q_{i+1}$ and $q_i \in L$ for all i .
- C. Conclude that every recursively enumerable language has an infinite recursive subset. Note that this part is easy despite being the last part. You may assume parts A and B when answering this part.

Answer.

- A. We give a decider M_1 for L as follows: On input x , at the i th step, run M and check the output w_i , there are three possibilities.
- If $w_i = x$, M_1 accepts;
 - If $w_i > x$, rejects;
 - Otherwise, $w_i < x$, continue to the next step.

Therefore, if $x \in L$, then $\exists j, w_j = x$ and M_1 accepts x . If $x \notin L$, since the enumeration from M is in increasing order and there are finite strings of order lower than x , $\exists j$ such that $w_{j-1} <_l x$ and $\forall k \geq j, w_k >_l x$ and M_1 would reject x with no doubt.

- B. M' looks like the following: Initially let $q_{\text{current}} = w_0$, at i th step, run M to get w_i , if $w_i <_l q_{\text{current}}$, ignore w_i , otherwise, let q_{current} be w_i and output w_i . Since there are infinite many strings within L , we could always find a string which has higher order than q_{current} . Thus, M' satisfies our requirements.
- C. From part B, we know that $L(M')$ is infinite subset of L and M' enumerate these strings in lexicographic order, then combining the conclusion from part A, $L(M')$ is decidable.

□

Problem 5. Denote by $\langle M \rangle$ the description of Turing machine M . Let $H(\langle M \rangle) = 1$ if M halts (on no input) and $H(\langle M \rangle) = 0$ otherwise. Define L by

$$L = \{(\langle M_1 \rangle, \langle M_2 \rangle) \mid H(\langle M_1 \rangle) \oplus H(\langle M_2 \rangle) = 1\}.$$

That is, L is the language of pairs of (description of) Turing machines such that exactly one of the Turing machines halts on no input.

- A. Show that $\text{HALT}_0 \leq_m L$.
- B. Show that $\overline{\text{HALT}_0} \leq_m L$.
- C. Show that $L \in \Sigma_2$. Bonus: show that it is not Σ_2 complete.

Answer.

- A. Given an instance i , we obtain a Turing machine M as follows: On every input x , run M_i on i , if it halts, accepts. It is easy to see that M accepts some input if and only if $i \in \text{HALT}_0$, which means $H(\langle M_{\text{test}} \rangle) = 0$ if and only if $i \in \text{HALT}_0$.

Let M_{test} be the machine which runs forever on every input, so $H(\langle M_{\text{test}} \rangle) = 1$. The output of our reduction is $(\langle M_{\text{test}} \rangle, \langle M \rangle)$ and trivially $(\langle M_{\text{test}} \rangle, \langle M \rangle) \in L$ if and only if $i \in \text{HALT}_0$.

- B. This time, let M_{test} be some machine which halts on λ (λ is not special, you could choose any string). The other parts of the construction work in the similar way.
- C. Given M , we create another Turing machine M' (It ignores the input) which runs M on all strings in the dovetailing manner and accepts as soon as M halts on some string. It is not hard to see that M' halts if and only if M halts on some input.

Therefore, we could decide L by making two oracle queries to the Halting problem, one for M'_1 and the other for M'_2 . So $L \in \Sigma_2$.

Bonus. It is known that the argument that the Halting problem is not decidable is relativizable and a Σ_2 -complete language is the r.e. complete language in the oracle(the Halting problem) version. By the conclusion of part C, L is impossible to be Σ_2 -complete since it is decidable with respect to the same oracle.

□