University of Nevada, Las Vegas Computer Science 456/656 Spring 2022

Assignment 4: Due Wednesday March 30 2022

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet. You will receive a message from our graduate assistant telling you how to turn in the assignment.

Throughout this assignment, you may assume that a language is recursively enumerable if and only if it is accepted by some machine. Recall that "L is recursively enumerable (RE)" means that there is a machine that enumerates L.

1. True/False/Open

   (a) **F** Every subset of a regular langugage is regular.

   (b) **O** If L_1 is $\mathcal{NP}$-complete and L_2 is $\mathcal{NP}$, there is a $\mathcal{P}$-TIME reduction of L_1 to L_2.

   (c) **T** If L_1 is $\mathcal{NP}$-complete and L_2 is $\mathcal{NP}$ and there is a $\mathcal{P}$-TIME reduction of L_1 to L_2, then L_2 is $\mathcal{NP}$-complete.

   (d) **O** If L is $\mathcal{NP}$-complete, there is no polynomial time algorithm which decides L.

   (e) **T** Every $\mathcal{NP}$ language is decidable.

   (f) **O** $\mathcal{NP} = \text{co-}\mathcal{NP}$.

   (g) **T** If L_1 is undecidable and there is a recursive reduction of L_1 to L_2, then L_2 is undecidable.

   (h) **F** The CF grammar equivalence problem is recursively enumerable.

   (i) **T** If a language L is decidable, then there must be a machine that enumerates L in canonical order.

   (j) **F** If there is a machine that enumerates a languaqge L, then L must be decidable.

   (k) **T** If there is a machine that accepts a languaqge L, then L must be recursively enumerable (RE).

   (l) **T** If a language L is decidable, there is a machine that enumerates L.

   (m) **T** If there is a machine that enumerates a languaqge L in canonical order, then L must be decidable.

   (n) **O** If $f : \mathcal{N} \to \mathcal{N}$ is a one-to-one and onto function, where $\mathcal{N}$ is the natural numbers (positive integers) we define the *inverse* of f to be a function $g : \mathcal{N} \to \mathcal{N}$ such that $f(g(n)) = n$ and $g(f(n)) = n$ for all $n \in \mathcal{N}$. There exists a one-to-one onto function $f : \mathcal{N} \to \mathcal{N}$ which can be computed in polynomial time whose inverse cannot be computed in polynomial time. (Such a function is called a *one-way* function.)

   (o) **F** There exists a recursive funtion T such that, for any provable statement P, there is a proof of P whose length does not exceed $T(n)$, where n is the length of P.

2. Consider the following CF grammar and LALR parser.

   1. $S \to i_2 S_3$
   2. $S \to i_2 S_3 e_4 S_5$
   3. $S \to w_6 S_7$
   4. $S \to a_8$

	ACTION					GOTO
	a	i	e	w	$	S
0	$s8$	$s2$		$s6$		1
1					halt	
2	$s8$	$s2$		$s6$		3
3			$s4$		$r1$	
4	$s8$	$s2$		$s6$		5
5			$r2$		$r2$	
6	$s8$	$s2$		$s6$		7
7			$r3$		$r3$	
8			$r4$		$r4$	

Walk through the computation of this parser where the input string is *iiwaeia*.

$\$_0$	:	$iiwaeia\$$		
$\$_0 i_2$	:	$iwaeia\$$	$s2$	
$\$_0 i_2 i_2$	:	$waeia\$$	$s2$	
$\$_0 i_2 i_2 w_6$	:	$aeia\$$	$s6$	
$\$_0 i_2 i_2 w_6 a_8$	:	$eia\$$	$s8$	
$\$_0 i_2 i_2 w_6 S_7$	:	$eia\$$	$r4$	4
$\$_0 i_2 i_2 S_3$	:	$eia\$$	$r3$	43
$\$_0 i_2 i_2 S_3 e_4$	:	$ia\$$	$s4$	43
$\$_0 i_2 i_2 S_3 e_4 i_2$	:	$a\$$	$s2$	43
$\$_0 i_2 i_2 S_3 e_4 i_2 a_8$	:	$	$s8$	43
$\$_0 i_2 i_2 S_3 e_4 i_2 S_3$	:	$	$r4$	434
$\$_0 i_2 i_2 S_3 e_4 S_5$	:	$	$r1$	4341
$\$_0 i_2 S_3$	:	$	$r2$	43412
$\$_0 S_1$	:	$	$r1$	434121

halt

3. Let L be a decidable language. Write a program in pseudo-code that enumerates L in canonical order.

   Let Σ be the alphabet of L. Let w_i be the i^{th} string in the canonical order of Σ^* The following program enumerates L in canonical order.

   For i from 1 to ∞
   If($w_i \in L$) write w_i

4. Let $L = \{\langle G_1\rangle\langle G_2\rangle : G_1, G_2$ are CF grammars that are not equivalent$\}$. Prove that L is recursively enumerable.[1] Assume that the terminal alphabet of both grammars is Σ. We need only give a program which enumerates L.

   Let $L_2 = \{\langle G_1\rangle\langle G_2\rangle : G_1, G_2$ are CF grammars$\}$.

   Note that $L \subseteq L_2$, and L_2 is decidable, in fact, it is $\mathcal{P}$-TIME, since all we have to do is check that both $\langle G_1\rangle$ and $\langle G_2\rangle$ describe CF grammars. Thus L_2 is recursively enumerable in canonical order. Consider the following program, P. For n from 1 to ∞

   For all $\langle G_1\rangle\langle G_2\rangle \in L_2$ of length no greater than n

   For all $w \in \Sigma^*$ of length no greater than n

   If($w \in L(G_1)$ and $w \notin L(G_2)$) write $\langle G_1\rangle\langle G_2\rangle$

   Else if($w \in L(G_2)$ and $w \notin L(G_1)$) write $\langle G_1\rangle\langle G_2\rangle$

   We need to show that P enumerates L. Suppose $u = \langle G_1\rangle G_2 \in L$. Let $i = |u|$. Let $w \in \Sigma^*$ be the shortest string over Σ which is generated by one of those grammars but not the other. (Such a string must exist, since the two grammars are not equivalent.) Let $j = |w|$. Let $n = \max\{i, j\}$. During the n^{th} iteration of the outer loop, u will be written. On the other hand, if G_1 and G_2 are equivalent, the string u will never be written. Thus P enumerates L.

5. Prove that the halting problem is undecidable.

   Recall that we can define $L(M)$, for any machine M, to be the set of all strings accepted by M, that is, all strings w such that M halts with input w. Let HALT $= \{\langle M\rangle w : w \in L(M)\}$. We prove that HALT is undecidable by contradiction. Assume HALT is decidable. Let DIAG $= \{\langle M\rangle : \langle M\rangle \notin L(M)\}$, the diagonal language. Note that $\langle M\rangle \in$ DIAG if and only if $\langle M\rangle\langle M\rangle \notin$ HALT. Thus, DIAG is decidable.

   Let M_{DIAG} be a machine which decides DIAG. Does M_{DIAG} accept its own encoding?

   By definition of DIAG, $\langle M_{\text{DIAG}}\rangle \in L(M_{\text{DIAG}})$ if and only if $\langle M_{\text{DIAG}}\rangle \notin$ DIAG. By the definition of M_{DIAG}, $\langle M_{\text{DIAG}}\rangle \in L(M_{\text{DIAG}})$ if and only if $\langle M_{\text{DIAG}}\rangle \in$ DIAG. This is a contradiction. We conclude that our assumption is false, that is, HALT is undecidable.

6. Given that 3-SAT is $\mathcal{NP}$-complete, prove, by reduction, that IND, the independent set problem, is also $\mathcal{NP}$-complete.

   We give a polynomial time reduction R of 3-SAT to IND. For any Boolean expression E in 3-CNF form, we define a graph G and a number k such that G has an independent set of size k if and only if E is satisfiable.

   Let $E = C_1 * C_2 * \cdots * C_k$ be a Boolean expression in 3-CNF form. Each clause C_i is the disjunction of three terms, each of which is a variable or the negation of a variable. We write $C_i = t_{i,1} + t_{i,2} + t_{i,3}$, where the $\{t_{i,m}\}$ are the terms. Let G be the graph whose vertices are $\{v_{i,m}\}$ for $1 \le i \le k$ and $m = 1, 2, 3$. There are two sets of edges of G, short edges and long edges. There is a short edge between $v_{i,m}$ and $v_{i,n}$ for any i if $n \ne m$. There is a long edge from $v_{i,m}$ to $v_{j,n}$ if $t_{i,m} * t_{j,n}$ is a contradiction: that is, if one of those terms is a variable x and the other is $!x$ This function (from a 3-CNF expressions to (G, k)) is clearly polynomial time.

[1] We know that L is not decidable, since the CF grammar equivalence problem is undecidable.

Suppose E is satisfiable. Pick a satisfying assignment of E. For each $i \in \{1, \dots k\}$, pick one term of C_i which is true under that assignment, a total of k terms altogether. Let S be the set of vertices of G corresponding to those terms. No two members of S are connected by a short edge, since the terms are in separate clauses. No two members of S are connected by a long edge, because the terms do not contradict. Therefore S is an independent set of vertices of G of order k.

Conversely, suppose S is an independent set of vertices of G of order k, and let T be the corresponding set of terms of E. Choose an assignment such that each member of T is true. Since no two members of T contradict, this can be done. There could be some variables which are not mentioned in T. These variables can be arbitrarily assigned true or false. Since there is at least one true term in each clause, each clause becomes true under that assignment, and thus E is satisfiable.