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

Answers to Assignment 5

Do not turn this assignment in.

1. True/False/Open

   (a) **F** Every subset of a recursively enumerable langugage is recursively enumerable.

   (b) **T** If L_1 is $\mathcal{NP}$ and L_2 is $\mathcal{NP}$-complete, 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 RE.

   (i) **T** The CF grammar equivalence problem is co-RE.

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

   (k) **T** If a language L is decidable, the complement of L is decidable.

   (l) **T** If a language L is undecidable, the complement of L is undecidable.

   (m) **F** If a language L is recursively enumerable, the complement of L is recursively enumerable.

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

   (o) **T** Every language has a canonical order enumeration.

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

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

   (r) **T** If a language L is decidable, there is a machine that enumerates L in canonical order.

   (s) **O** There exist infinitely many one-way functions.

   (t) **T** Every regular language is in $\mathcal{NC}$.

   (u) **T** Every context-free language is in $\mathcal{NC}$.

   (v) **O** The Boolean circuit probolem is in $\mathcal{NC}$.

2. Consider the following CF grammar and LALR parser for an algebraic language. In this language, unary $-$ has highest precedence, $\wedge$ has precendence over $*$, which has precedence over $+$ and binary $-$. Operators $+$, $-$, $*$ are left associative, and $\wedge$ is right associative.

   1. $E \to E_{1,13} +_2 E_3$
   2. $E \to E_{1,13} -_4 E_5$
   3. $E \to E_{1,2,4,13} *_6 E_7$
   4. $E \to E_{1,3,5,7} \wedge_8 E_9$
   5. $E \to -_{10} E_{11}$
   6. $E \to (_{12} E_{13})_{14}$
   7. $E \to \mathbf{id}_{15}$

   (a) Fill in the missing entries in the tables:
   Row 2 columns **id**, $-$, (, E
   Row 3 columns $+$, $-$, $*$
   Row 5 columns $+$, $-$, $*$
   Row 9 columns $*$, $\wedge$

   (b) Walk through the computation of the parser, where the input string is $(u + v * x) \wedge y \wedge -z$

	ACTION								GOTO
	id	+	−	∗	∧	(	)	$	E
0	s15		s10			s12			1
1		s2	s4	s6	s8			halt	
2	s15		s10			s12			3
3		r1	r1	s6	s8		r1	r1	
4	s15		s10			s12			5
5		r2	r2	s6	s8		r2	r2	
6	s15		s10			s12			7
7		r3	r3	r3	s8		r3	r3	
8	s15		s10			s12			9
9		r4	r4	r4	s8		r4	r4	
10	s15		s10			s12			11
11		r5	r5	r5	r5		r5	r5	
12	s15		s10			s12			13
13		s2	s4	s6	s8		s14		
14		r6	r6		r6		r6	r6	
15		r7	r7		r7		r7	r7	

$\$_0$	:	$(u+v*x)\wedge y\wedge -z\$$		
$\$_0(_{12}$	:	$u+v*x)\wedge y\wedge -z\$$	$s12$	
$\$_0(_{12}\mathbf{id}_{15}$	:	$+v*x)\wedge y\wedge -z\$$	$s15$	
$\$_0(_{12}E_{13}$	:	$+v*x)\wedge y\wedge -z\$$	$r7$	7
$\$_0(_{12}E_{13}+_2$	:	$v*x)\wedge y\wedge -z\$$	$s2$	7
$\$_0(_{12}E_{13}+_2\mathbf{id}_{15}$	:	$*x)\wedge y\wedge -z\$$	$s15$	7
$\$_0(_{12}E_{13}+_2E_3$	:	$*x)\wedge y\wedge -z\$$	$r7$	77
$\$_0(_{12}E_{13}+_2E_3*_6$	:	$x)\wedge y\wedge -z\$$	$r7$	77
$\$_0(_{12}E_{13}+_2E_3*_6\mathbf{id}_{15}$	:	$)\wedge y\wedge -z\$$	$s15$	77
$\$_0(_{12}E_{13}+_2E_3*_6E_7$	:	$)\wedge y\wedge -z\$$	$r7$	777
$\$_0(_{12}E_{13}+_2E_3$	:	$)\wedge y\wedge -z\$$	$r3$	7773
$\$_0(_{12}E_{13}$	:	$)\wedge y\wedge -z\$$	$r1$	77731
$\$_0(_{12}E_{13})_{14}$	:	$\wedge y\wedge -z\$$	$s14$	77731
$\$_0E_1$	:	$\wedge y\wedge -z\$$	$r6$	777316
$\$_0E_1\wedge_8$	:	$y\wedge -z\$$	$s8$	777316
$\$_0E_1\wedge_8\mathbf{id}_{15}$	:	$\wedge - z\$$	s_{15}	777316
$\$_0E_1\wedge_8E_9$	:	$\wedge - z\$$	$r7$	7773167
$\$_0E_1\wedge_8E_9\wedge_8$	:	$-z\$$	$s8$	7773167
$\$_0E_1\wedge_8E_9\wedge_8-_{10}$	:	$z\$$	$s10$	7773167
$\$_0E_1\wedge_8E_9\wedge_8-_{10}\mathbf{id}_{15}$	:	$\$$	$s15$	7773167
$\$_0E_1\wedge_8E_9\wedge_8-_{10}E_{11}$	:	$\$$	$r7$	77731677
$\$_0E_1\wedge_8E_9\wedge_8E_9$	:	$\$$	$r5$	777316775
$\$_0E_1\wedge_8E_9$	:	$\$$	$r4$	7773167754
$\$_0E_1$	:	$\$$	$r4$	77731677544

HALT

3. Which of these problems, or languages, are known to be xxx $\mathcal{NP}$-complete?

   (a) The firehouse problem: given a graph G and a number K, does there exist a set F of vertices of G of cardinality K such that every vertex of G is either a member of F or adjacent to a member of F?
   Known to be $\mathcal{NP}$-complete

   (b) The **bounded** subset sum problem. Given an integer K and a list of n positive integers, $x_1, \ldots x_n$, such that $x_i \leq 100$ for each i, does there exist a sublist whose sum is exactly K?
   Polynomial time, not known to be $\mathcal{NP}$-complete

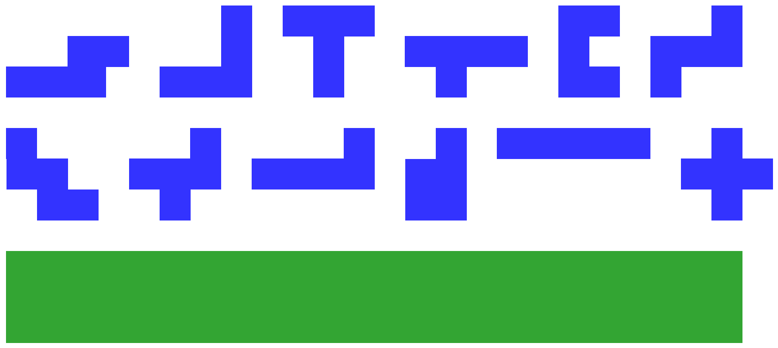
   (c) Boolean satisfiability.
   Known to be $\mathcal{NP}$-complete

   (d) The Boolean circuit problem.
   Polynomial time, not known to be $\mathcal{NP}$-complete

(e) 2-SAT, the set of all satisfiable Boolean expressions in 2-CNF form.
Polynomial time, not known to be $\mathcal{NP}$-complete

(f) The set of all configurations of RUSH-HOUR, for any size board, from which it is possible to win.
$\mathcal{P}$-SPACE complete, not known to be $\mathcal{NP}$-complete

(g) The tiling problem: given a finite set of small polygons and one large polygon, is it possible to place all the small polygons so as to exactly cover the large polygon?
Known to be $\mathcal{NP}$-complete. The partition problem can be reduced to the tiling problem in polynomial time.
Here is an example instance of this problem.



4. State the pumping lemma for context-free languages.

For any context free language L, there is a number p such that, for any string $w \in L$ of length at least p, there are strings u, v, x, y, z such that the following four statements hold:
1. $w = uvxyz$
2. $|vxy| \leq p$
3. v and y are not both empty
4. For any $i \geq 0$ $uv^ixy^iz \in L$

5. Give a polynomial time reduction of the subset sum problem to the partition problem.

Suppose $x_1, x_2, \ldots x_n; K$ is an instance of the subset sum problem. Let $S = \sum i = 1^n$. If $K > S$, there is clearly no solution, so we can assume $K \leq S$. We reduce that instance to the following instance of the partition problem: $x_1, x_2, \ldots x_n, K + 1, S - K + 1$.

6. Give a polynomial time reduction of 3-SAT to the independent set problem.

Suppose $e = C_1 * C_2 * \cdots * C_k$ is a Boolean expression in 3-CNF form. For each $i \in \{1 \ldots n\}$, let $C_i = t_{i,1} + t_{i,2} + t_{i,3}$.

Let (G, k) be the instance of the independent set problem, where $G = (V, E)$, where $V = \{v_{i,j} : 1 \leq i \leq k, j = 1, 2, 3\}$ and where E is the union of the following two sets:

(a) Short edges. There is a short edge from $v_{i,1}$ to $v_{i,2}$ and $v_{i,3}$ and from $v_{i,2}$ to $v_{i,3}$.

(b) Long edges. There is a long edge from $v_{i,j}$ to $v_{i',j'}$ if $t_{i,j} * t_{i',j'}$ is a contradiction.

G has an independent set of order k if and only if e is satisfiable.

7. Give a context-sensive grammar for the language $\{a^n b^n c^n d^n \ : \ n \geq 1\}$.

   There are many correct answers. Here is one.

   $S \to abcd$
   $S \to Aabcd$
   $Aa \to aA$
   $Aa \to AaA$
   $aAb \to aabbA$
   $Ab \to bA$
   $Ac \to cA$
   $cAd \to ccdd$

   Here is a derivation.
   $S \Rightarrow Aabcd \Rightarrow AaAbcd \Rightarrow AaabbAcd \Rightarrow AaabbcAd \Rightarrow Aaabbccdd \Rightarrow aAabbccdd \Rightarrow aaAbbccdd \Rightarrow aaabbAbccdd \Rightarrow aaabbbAccdd \Rightarrow aaabbbcAcdd \Rightarrow aaabbbccAdd \Rightarrow aaabbbcccddd$

8. Give a context-sensitive grammar for the language $\{a^n \ : \ n$ is a power of $2\}$

   There are many correct answers. Here is one. L, R, D stand for "left," "right" and "double." $S \to a$
   $S \to aa$
   $S \to LaaR$
   $L \to LaD$
   $Da \to aaD$
   $DR \to aR$
   $L \to a$
   $R \to a$
   Here is a derivation.
   $S \Rightarrow LaaR \Rightarrow LaDaaR \Rightarrow LaaaDaR \Rightarrow LaaaaaDR \Rightarrow LaaaaaaR \Rightarrow LaDaaaaaaR \Rightarrow LaaaDaaaaaR \Rightarrow LaaaaaDaaaaR \Rightarrow LaaaaaaaDaaaR \Rightarrow LaaaaaaaaaDaaR \Rightarrow LaaaaaaaaaaaDaR \Rightarrow LaaaaaaaaaaaaaDR \Rightarrow LaaaaaaaaaaaaaaR \Rightarrow aaaaaaaaaaaaaaaR \Rightarrow aaaaaaaaaaaaaaaa = a^{16}$

9. Suppose a machine M accepts a language L over an alphabet Σ. Prove that L is recursively enumerable.

   Let $w_1, w_2, \ldots$ be an enumeration of Σ^* in canonical order.

   The following program enumerates L.

```
For all $t$ from 1 to infinity.
  For all $n$ from 1 to $t$
    for all $i$ from 1 to $t$
      if $M$ accepts $w_i$ in at most $t$ steps
        write $w_i$
```

   For all $w \in L$, $w = w_i$ for some i, and M accepts w within j steps. Let $t = maxi, j$. Then w will be be writen during the t^{th} iteration of the outer loop. If $w \notin L$, then w is not accepted by M and will never be written.

10. Suppose a language L is recursively enumerable. Prove that L is accepted by some machine.

    Let $w_1, w_2, \ldots$ be the enumeration of L written by some machine. The following program accepts L.

    ```
    Read $w$. For all $i$ from 1 to infinity
      if $w = w_i$    Accept
    ```

11. State the Church-Turing thesis, and explain why it is important.

    Machine is equivalent to some Turing machine. That is, every machine can be emulated by some Turing machine. This is important because Turing machines are very simple, and if we can prove that there there is no Turing machine which can do a certain task, we know that no machine can do that task.

12. Why is the question of whether $\mathcal{P} = \mathcal{NC}$ important?

    Because parallel machines are becoming more common, as there is a theoretical limit on the power of a single processor, which we will soon reach, or have already reached. We would like to take a problem that could be worked by one processor in some very large amount of time, so large that it is impractical for a one processor machine, and run it on a parallel machine by having each processor do one part of the work, and then solve the problem faster.

    If $\mathcal{P} = \mathcal{NC}$, this can always be done if we have enough processors.

13. Prove that the halting problem is undecidable.

    Here is the proof I gave in class. However, I will accept any correct proof.

    Assume that HALT is decidable. Recall that $\langle M \rangle w \in$ HALT if and only if M accepts w.

    Let DIAG $= \{\langle M \rangle : \langle M \rangle \langle M \rangle \notin \text{HALT}\}$ Then DIAG is decidable since HALT is decidable. Since DIAG is decidable, it is accepted by some machine M_{DIAG}.

    We now ask whether $\langle M_{\text{DIAG}} \rangle \in$ DIAG.

    $\langle M_{\text{DIAG}} \rangle \in$ DIAG if and only if M_{DIAG} accepts $\langle M_{\text{DIAG}} \rangle$, by definition of M_{DIAG}. On the other hand, $\langle M_{\text{DIAG}} \rangle \in$ DIAG if and only if M_{DIAG} does not accept $\langle M_{\text{DIAG}} \rangle$, by definition of DIAG. Contradiction. Thus, HALT is not decidable.

I announced in class that I would post a CS grammar for $L = \{a^n b^n c^n : n \geq 1\}$. Here it is.

$S \to abc$
$S \to Aabc$
$A \to AA$
$Aa \to aA$
$Ab \to abB$
$bBc \to bbcc$

Here is a derivation of *aaabbbccc*.

$S \Rightarrow Aabc \Rightarrow AAabc \Rightarrow AaAbc \Rightarrow AaabBc \Rightarrow Aaabbcc \Rightarrow aAabbcc \Rightarrow aaAbbcc \Rightarrow aaabBbcc \Rightarrow aaabBbcc \Rightarrow aaabbBcc \Rightarrow aaabbbccc$