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 language is recursively enumerable.
   (b) T If $L_1$ is $NP$ and $L_2$ is $NP$-complete, there is a $P$-time reduction of $L_1$ to $L_2$.
   (c) T If $L_1$ is $NP$-complete and $L_2$ is $NP$ and there is a $P$-time reduction of $L_1$ to $L_2$, then $L_2$ is $NP$-complete.
   (d) O If $L$ is $NP$-complete, there is no polynomial time algorithm which decides $L$.
   (e) T Every $NP$ language is decidable.
   (f) O $NP = co-NP$.
   (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 $NC$.
   (u) T Every context-free language is in $NC$.
   (v) O The Boolean circuit problem is in $NC$. 
2. Consider the following CF grammar and LALR parser for an algebraic language. In this language, unary − has highest precedence, ∧ has precedence over *, which has precedence over + and binary −. Operators +, −, * are left associative, and ∧ is right associative.

(a) Fill in the missing entries in the tables:

<table>
<thead>
<tr>
<th>ACTION</th>
<th>id</th>
<th>+</th>
<th>−</th>
<th>*</th>
<th>∧</th>
<th>( )</th>
<th>$</th>
<th>GOTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>s15</td>
<td>s10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>s2</td>
<td>s4</td>
<td>s6</td>
<td>s8</td>
<td></td>
<td></td>
<td></td>
<td>halt</td>
</tr>
<tr>
<td>2</td>
<td>s15</td>
<td>s10</td>
<td></td>
<td></td>
<td>s12</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>r1</td>
<td>r1</td>
<td>s6</td>
<td>s8</td>
<td></td>
<td>r1</td>
<td>r1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>s15</td>
<td>s10</td>
<td></td>
<td>s8</td>
<td>s12</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>r2</td>
<td>r2</td>
<td>s6</td>
<td>s8</td>
<td></td>
<td>r2</td>
<td>r2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>s15</td>
<td>s10</td>
<td></td>
<td></td>
<td>s12</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>r3</td>
<td>r3</td>
<td>r3</td>
<td>s8</td>
<td></td>
<td>r3</td>
<td>r3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>s15</td>
<td>s10</td>
<td></td>
<td></td>
<td>s12</td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>r4</td>
<td>r4</td>
<td>r4</td>
<td>s8</td>
<td></td>
<td>r4</td>
<td>r4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>s15</td>
<td>s10</td>
<td></td>
<td></td>
<td>s12</td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>r5</td>
<td>r5</td>
<td>r5</td>
<td>r5</td>
<td></td>
<td>r5</td>
<td>r5</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>s15</td>
<td>s10</td>
<td></td>
<td></td>
<td>s12</td>
<td></td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>s2</td>
<td>s4</td>
<td>s6</td>
<td>s8</td>
<td></td>
<td></td>
<td>s14</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>r6</td>
<td>r6</td>
<td>r6</td>
<td>r6</td>
<td></td>
<td>r6</td>
<td>r6</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>r7</td>
<td>r7</td>
<td>r7</td>
<td>r7</td>
<td></td>
<td>r7</td>
<td>r7</td>
<td></td>
</tr>
</tbody>
</table>

(b) Walk through the computation of the parser, where the input string is \((u + v * x) \land y \land -z\)
3. Which of these problems, or languages, are known to be \( \mathcal{NP} \)-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 \( i \geq 0 \) \( uv^i xy^i z \in L \)

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 \( e = C_1 \ast C_2 \ast \cdots \ast C_k \) 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 \( t_{i,j} \ast t_{i',j'} \) 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 \( \{a^nb^nc^nd^n : n \geq 1\} \).

There are many correct answers. Here is one.

\[
S \rightarrow abcd \\
S \rightarrow Aabcd \\
Aa \rightarrow aA \\
Aa \rightarrow AaA \\
aAb \rightarrow aabbA \\
Ab \rightarrow bA \\
Ac \rightarrow cA \\
cAd \rightarrow ccdd
\]

Here is a derivation.

\[
S \Rightarrow Aabcd \Rightarrow AaAbcd \Rightarrow AaabAcd \Rightarrow Aaabbcddd \Rightarrow aaAbbcddd \Rightarrow aaabbAbcddd \Rightarrow aaabbbcAccdd \Rightarrow aaabbbbcAdd \Rightarrow aaabbbccAdd \Rightarrow aaabbbccddd
\]

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

There are many correct answers. Here is one. \( L, R, D \) stand for “left,” “right” and “double.”

\[
S \rightarrow aa \\
S \rightarrow LaaR \\
L \rightarrow LaD \\
Da \rightarrow aaD \\
DR \rightarrow aR \\
L \rightarrow a \\
R \rightarrow a
\]

Here is a derivation.

\[
S \Rightarrow LaaR \Rightarrow LaDaR \Rightarrow LaaaDaR \Rightarrow LaaaaaDR \Rightarrow LaaaaaaR \Rightarrow LaaaDaaaaaR \Rightarrow LaaaaaaaDaR \Rightarrow LaaaaaaaaaDaR \Rightarrow LaaaaaaaaaaaDaR \Rightarrow LaaaaaaaaaaaaDaR \Rightarrow LaaaaaaaaaaaaaDaR \Rightarrow LaaaaaaaaaaaaaaaDaR \Rightarrow \ldots \Rightarrow a^{16}
\]

9. Suppose a machine \( M \) accepts a language \( L \) over an alphabet \( \Sigma \). Prove that \( L \) is recursively enumerable.

Let \( w_1, w_2, \ldots \) be an enumeration of \( \Sigma^* \) 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 \( w \in L, w = w_i \), and \( M \) accepts \( w \) within \( t \) steps. Let \( t = \max_i j \). Then \( w \) will be be written 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 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 $P = NP$ 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 $P = NP$, 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 $\text{DIAG} = \{ \langle M \rangle : \langle M \rangle \notin$ HALT$\}$ Then $\text{DIAG}$ is decidable since HALT is decidable. Since $\text{DIAG}$ is decidable, it is accepted by some machine $M_{\text{DIAG}}$.

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

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

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

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

Here is a derivation of $aaabbbc$.c

$S \Rightarrow Aabc \Rightarrow AAabc \Rightarrow AaAabc \Rightarrow AaabBc \Rightarrow Aaabbc \Rightarrow aaAbbc \Rightarrow aaabBcc \Rightarrow aaabBb \Rightarrow aaabbBc \Rightarrow aaabbb$