Answers

The entire examination is 180 points.

1. Give a context-free grammar for each of these languages. As you know, it is undecidable whether an answer is correct, but I did the best I could.

(a) [20 points] The language of all strings of the form $a^i b^j$ such that $i > j$. I gave full credit for any correct answer, provided I could prove that it’s correct. There are several equally nice answers. Here are two of them. The first is ambiguous, the second unambiguous.

\[
S \rightarrow aSb \\
S \rightarrow aS \\
S \rightarrow a \\
S \rightarrow aSb \\
S \rightarrow aA \\
A \rightarrow aA \\
A \rightarrow \varepsilon
\]

(b) [20 points] The language of all strings over $\Sigma = \{a, b\}$ which have equal numbers of each symbol, such as $ab$, $ba$, $aababb$, $bbabbaaa$. I should have given a few more examples, such as $\varepsilon$ and $aabbbbaa$. Most people’s grammars did not generate $aabbbbaa$. Here are two answers. The first is ambiguous, the second unambiguous.

\[
S \rightarrow aSb \\
S \rightarrow bSa \\
S \rightarrow SS \\
S \rightarrow \varepsilon
\]

(c) [20 points] The language of all strings of the form $a^i b^j$ such that $i \neq j$. Hint: You will need two variables besides the start symbol: one for the case $i < j$, the other for the case $i > j$. This is actually not much harder than (a). The first answer is ambiguous, the second unambiguous.

\[
S \rightarrow A|B \\
A \rightarrow aAb \\
A \rightarrow Aa \\
A \rightarrow a \\
B \rightarrow aBb \\
B \rightarrow bB \\
B \rightarrow b
\]
(d) [20 points] The language consisting of all binary strings of length at least 2, such that the second to the last symbol is 0.

Why is this hard?

\[ S \rightarrow 0S \]
\[ S \rightarrow 1S \]
\[ S \rightarrow 00 \]
\[ S \rightarrow 01 \]

2. Give an unambiguous context-free grammar for each of these languages.

(a) [20 points] The Dyck language over the alphabet \{a, b\}. That is, use a, b instead of left and right parentheses. There are two very simple standard answers.

\[ S \rightarrow aSbS \]
\[ S \rightarrow \varepsilon \]

(b) [20 points] The language of all valid C++ expressions using only the operators *, /, +, −, using only the variables x, y, and z, and using parentheses. The operator − is used for both subtraction and negation. Do not consider unary +. Your grammar must respect the semantics of C++ expressions.

I didn’t mention parentheses, and this caused confusion. I decided not to grade it, so there are only 160 possible points. Here is the answer I wanted. The start symbol is E.

\[ E \rightarrow E + T \]
\[ E \rightarrow E - T \]
\[ E \rightarrow T \]
\[ T \rightarrow T \ast F \]
\[ T \rightarrow T / F \]
\[ T \rightarrow F \]
\[ F \rightarrow -F \]
\[ F \rightarrow x \]
\[ F \rightarrow y \]
\[ F \rightarrow z \]
\[ F \rightarrow (E) \]
(c) [20 points] The language consisting of all alphanumeric identifiers using no upper case letters. Such an identifier is a non-empty string of lower case Roman letters and Arabic digits, where the first symbol is a letter.

This is a regular language, and is generated by a left-regular grammar, and also by a right-regular grammar. The grammar below would be considered to be right-regular if $R$ and $A$ where single symbols.

Since the strings are identifiers, I will let the start symbol be $I$.

$I \to RT$
$T \to RT$
$T \to AT$
$T \to \varepsilon$
$R \to a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z$
$A \to 0|1|2|3|4|5|6|7|8|9$

(d) [20 points] $R_{a,b}$, the language consisting of all regular expressions over the alphabet $\Sigma = \{a,b\}$. Recall that the terminal alphabet of $R_{a,b}$ is $\{a,b,\varepsilon,\emptyset,+,\ast,()\}$ Your grammar must respect the semantics of regular expressions.

We had regular expressions at the beginning of the course. The $\ast$ is not a binary operator; rather it is the unary operator for Kleene closure. Concatenation does not have symbol, but is indicated by concatenation. Since $R_{a,b}$ is an algebraic language, I will use $E$ for the start symbol. $E$ is for expression, $T$ is for term, and $F$ is for factor, just as in Problem 2b.

The first of these grammars is ambiguous, the second unambiguous.

\[
\begin{align*}
E & \to E + T \\
E & \to T \\
E & \to EE \\
E & \to E^* \\
E & \to a \\
E & \to b \\
E & \to \varepsilon \\
E & \to \emptyset \\
E & \to (E)
\end{align*}
\]
3. [20 points] Given that the knapsack problem is \( \mathcal{NP} \)-complete, show that the partition problem is \( \mathcal{NP} \)-complete by using reduction.

Most people who gave a reduction did not prove that it was a reduction. That’s part of the problem.

I will give a reduction which is easier than the one I gave in class, since it has only one case instead of two.

An instance of the knapsack problem consists of a set \( \mathcal{X} \) of “items,” each having a positive integral weight, together with a positive integral “knapsack size,” \( K \). The answer to that instance is “yes,” if and only if there is a subset of \( \mathcal{X} \) whose total weight is \( K \).

An instance of the partition problem is a set \( \mathcal{Y} \) of items, each having a positive integral weight. The answer to that instance is “yes,” if and only if \( \mathcal{Y} \) can be partitioned into two subsets of equal total weight.

We now define the reduction \( R \). If \( I = (\mathcal{X}, K) \) is an instance of the knapsack problem, Let \( n \) be the number of weights, \( x_i \) the weight of the \( i \)th item, and \( S = \sum_{i=1}^{n} x_i \). Let \( k \) be a new item of weight \( 2K \), and \( s \) a new item of weight \( S \). We let \( R(I) = J \), where \( J \) consists of the set \( \mathcal{Y} = \mathcal{X} \cup \{k, s\} \). We now need to prove that this function is a reduction. Note that the total weight of the items of \( \mathcal{Y} \) is \( 2K + 2S \).

If \( I \) has a solution, there is a set \( \mathcal{S} \subseteq \mathcal{X} \) whose total weight is \( K \). Then \( \mathcal{T} = \mathcal{S} \cup \{s\} \) has total weight \( K + S \), which is half the weight of \( \mathcal{Y} \). Then \( \mathcal{Y} \) can be partitioned into sets \( \mathcal{T} \) and its complement, which each have total weight \( K + S \), hence \( J \) has a solution.

Conversely, suppose \( J \) has a solution. Then \( \mathcal{Y} \) is the union of disjoint subsets \( \mathcal{S}_1 \) and \( \mathcal{S}_2 \), each of which has weight \( K + S \).

Since the combined weight of \( k \) and \( s \) is \( 2K + S > K + S \), those two weights cannot be in the same subset.

Without loss of generality, \( s \in \mathcal{S}_1 \). Let \( \mathcal{T} \) be the set consisting of the remaining elements of \( \mathcal{S}_1 \). Since the weight of \( s \) plus the weight of \( \mathcal{T} \) is the weight of \( \mathcal{S}_1 \), which is \( S + K \), we know that \( \mathcal{T} \) has total weight \( K \). Furthermore, \( \mathcal{T} \subseteq \mathcal{X} \), hence is a solution to \( I \).

In conclusion, we have shown that an instance \( I \) of the knapsack problem has a solution if and only if \( R(J) \), an instance of the partition problem, has a solution.

We know that the knapsack problem is \( \mathcal{NP} \)-complete, and clearly the partition problem is \( \mathcal{NP} \), thus the partition problem is \( \mathcal{NP} \)-complete.