Sample Solution to Problem 4

Problem 4: Describe an algorithm for the following decision problem: Given a context-free grammar $G$ with terminal alphabet $\{a, b\}$, is there a string $w \in L(G)$ such that $w$ includes at least one occurrence of $a$ and at least one occurrence of $b$? Or, to put the question another way, is it the case that

$$L(G) \cap (a+b)^*a(a+b)^* \cap (a+b)^*b(a+b)^* \neq \emptyset?$$

Solution: Assume, without loss of generality, that $G = (V, \{a, b\}, P, S)$ has no useless symbols. (Recall that such symbols can be removed algorithmically.) For brevity, we will use $V_{ab}$ as an abbreviation for $V \cup \{a, b\}$.

Here is a sketch of the algorithm:

1. Compute $\Gamma_a = \{ X \in V_{ab} \mid \text{for some } \alpha, \beta \in V_{ab}, X \Rightarrow \alpha a \beta \}$. Informally, $\Gamma_a$ consists of those symbols that can produce an occurrence of $a$.
2. Compute $\Gamma_b = \{ X \in V_{ab} \mid \text{for some } \alpha, \beta \in V_{ab}, X \Rightarrow \alpha b \beta \}$. Informally, $\Gamma_b$ consists of those symbols that can produce an occurrence of $b$.
3. If there is a production $A \rightarrow Y_1 Y_2 \cdots Y_m \in P$ such that, for some $i$ and $j$ satisfying $i \neq j$, $Y_i \in \Gamma_a$ and $Y_j \in \Gamma_b$, answer YES; otherwise, answer NO.

Before describing how to carry out the first two steps, let us justify the third:

Lemma: There exists a string $w \in L(G) \cap (a+b)^*a(a+b)^* \cap (a+b)^*b(a+b)^*$ if and only if $G$ has a production $A \rightarrow Y_1 Y_2 \cdots Y_m$ where, for some $i$ and $j$ satisfying $i \neq j$, $Y_i \in \Gamma_a$ and $Y_j \in \Gamma_b$.

Proof: $(\Rightarrow)$ Suppose that $G$ has a production $A \rightarrow Y_1 Y_2 \cdots Y_m$ as described in the statement of the lemma. Then for some $u, v, y_k (i \neq k \neq j), y_{i,1}, y_{i,2}, y_{j,1}, y_{j,2} \in \{a, b\}^*$ satisfying $Y_k \Rightarrow y_k (i \neq k \neq j), Y_i \Rightarrow y_{i,1} a y_{i,2}$, and $Y_j \Rightarrow y_{j,1} a y_{j,2}$ (all of these being justified by our assumption that there are no useless symbols in $G$), we have

$$S \Rightarrow uAv \Rightarrow uY_1 Y_2 \cdots Y_i \cdots Y_j \cdots Y_m v \Rightarrow u y_1 y_2 \cdots y_{i,1} a y_{i,2} \cdots y_{j,1} b y_{j,2} \cdots y_m v$$

(Note that, although the above suggests $i < j$, that need not be the case.) It follows that $L(G)$ includes a string that has at least one occurrence of $a$ and at least one occurrence of $b$.

$(\Rightarrow)$ Suppose that $S \Rightarrow w$, where $w$ has at least one occurrence of $a$ and at least one occurrence of $b$. In a corresponding derivation tree (i.e., one having a root labeled $S$ and leaves that, when their labels are read from left to right, spell out $w$), find a leaf labeled $a$ and a leaf labeled $b$. Let $A$ be the label on the node that is the nearest common ancestor of those two leaves. Let the children of that node be labeled $Y_1, Y_2, \ldots, Y_m$, going from left to right. One of them, say the $i$-th, is an ancestor of the aforementioned leaf labeled $a$, and another, say the $j$-th, is an
ancestor of the leaf labeled $b$. But then $G$ must have a production $A \rightarrow Y_1 Y_2 \cdots Y_m$ such that $Y_i \in \Gamma_a$ and $Y_j \in \Gamma_b$, where $i \neq j$. QED

It remains to describe how to compute $\Gamma_a$ and $\Gamma_b$. As computing one is just like computing the other, it suffices to demonstrate how to compute $\Gamma_a$. Consider the following algorithm.

\begin{verbatim}
gamma := \{a\};
toExplore := \{a\};
do while toExplore \neq \emptyset
    X := toExplore.chooseOne();
    do for each $A \rightarrow \eta \in P$ such that $A \notin \gamma$ and $X$ occurs in $\eta$
        \begin{verbatim}
gamma := gamma \cup \{A\};
toExplore := toExplore \cup \{A\};
od;
\end{verbatim}
    toExplore := toExplore \setminus \{X\};
od;
\end{verbatim}

\textbf{Lemma:} Execution of the above results in $\gamma = \Gamma_a$.

\textbf{Proof:} ($\subseteq$) We show by induction on the number of loop iterations (of the outer loop) that

$$Inv: \text{toExplore} \subseteq \gamma \subseteq \Gamma_a$$

is an invariant of that loop.

Before the first loop iteration, we have $\text{toExplore} = \gamma = \{a\} \subseteq \Gamma_a$. This completes the basis. As an induction hypothesis, assume that, for some $n \geq 0$, $Inv$ holds after the $n$-th iteration. If, during the $(n + 1)$-st iteration, the symbol $A$ is inserted into $\gamma$, it can only be because there is a production $A \rightarrow \eta$ such that $X$ occurs in $\eta$, where $X \in \text{toExplore}$. By the induction hypothesis, $X \in \Gamma_a$, which implies (taking $\eta = \eta_1X\eta_2$) that

$$A \Rightarrow \eta_1X\eta_2 \Rightarrow \eta_1\alpha\beta\eta_2$$

for some $\alpha, \beta \in V_{ab}^*$. It follows that $A \in \Gamma_a$; hence, inserting $A$ into both $\text{toExplore}$ and $\gamma$ preserves the truth of $Inv$. Removing $X$ from $\text{toExplore}$ also preserves $Inv$. This completes the induction, thereby proving that $Inv$ is a loop invariant. As a consequence, $Inv$ will be true (if and) when the loop terminates, from which it follows that, at that moment, $\gamma \subseteq \Gamma_a$ holds, as was to be proved.

($\supseteq$) For $i \geq 0$, define $\Phi_i \subseteq V_{ab}$ as follows:

$$\Phi_0 = \{a\}$$

$$\Phi_{k+1} = \Phi_k \cup \{A \in V \mid A \rightarrow Y_1 Y_2 \cdots Y_m \in P \text{ and, for some } i, Y_i \in \Phi_k\} \ (k \geq 0)$$

We make the following observations, leaving the proofs to the reader:

1. $X \in \Phi_k$ if and only if there exists a derivation tree having a root labeled $X$ and a leaf labeled $a$ at distance $k$ or less from the root.
2. \( \Gamma_a = \Phi_0 \cup \Phi_1 \cup \Phi_2 \cup \cdots \)

To show that, upon completion of the algorithm, \( \gamma \) includes all the members of \( \Gamma_a \), we first suppose otherwise. Let \( j \) be the smallest value such that \( \Phi_j \not\subseteq \gamma \), and let \( X \in \Phi_j \) but \( X \notin \gamma \). It cannot be that \( j = 0 \), because \( \Phi_0 = \{ a \} \subseteq \gamma \). By definition of \( \Phi_j \), from \( X \in \Phi_j \) (and \( j > 0 \)) it follows that there is a production \( X \rightarrow Y_1Y_2\cdots Y_m \in P \) such that, for some \( i \), \( Y_i \in \Phi_{j-1} \). By our assumption that \( \Phi_{j-1} \subseteq \gamma \), it must be that \( Y_i \in \gamma \) holds upon termination of the loop. By examining the algorithm, we reason that \( Y_i \) must have been inserted into \texttt{toExplore} at some point during its execution. (After all, every symbol inserted into \( \gamma \) is immediately thereafter inserted into \texttt{toExplore}.) But that means that, during some subsequence iteration of the loop, \( Y_i \) was removed from \texttt{toExplore} and every symbol appearing on the left-hand side of any production in whose right side \( Y_i \) appears was inserted into \( \gamma \). (These actions occur during iterations of the nested loop.) In particular, that means that \( A \) (which, you will recall, is the left-hand side of a production \( A \rightarrow Y_1 \cdots Y_i \cdots Y_m \in P \) on whose right side \( Y_i \) appears) must have been inserted into \( \gamma \). This (together with the fact that symbols are never removed from \( \gamma \)) contradicts our assumption that \( A \notin \gamma \). This completes the proof that \( \gamma \supseteq \Gamma_a \).

Having proved both that \( \gamma \subseteq \Gamma_a \) and \( \gamma \supseteq \Gamma_a \) upon termination of the algorithm, we conclude that \( \gamma = \Gamma_a \) at that time.

It is left to prove only that the algorithm is guaranteed to terminate. Toward that end, we make these observations:

1. no symbol can be inserted into \texttt{toExplore} more than once (because \( \gamma \) includes every symbol that was ever a member of \texttt{toExplore} and a symbol already in the former cannot be inserted into the latter), and

2. on each iteration, one symbol is removed from \texttt{toExplore}.

It follows that the number of iterations of the outer loop is at most \(|V| + 1 \). (This allows for the possibility that every nonterminal symbol, plus \( a \), is inserted into \texttt{toExplore}.) This completes the proof that the algorithm is guaranteed to terminate. \( \text{QED} \)