4. **Problem:** Prove that the class of context-free languages is closed under unions, concatenations, and the Kleene star operation, i.e. if the languages $L_1, L_2 \subseteq \Sigma^*$ are context-free, then so are the languages $L_1 \cup L_2$, $L_1 L_2$ and $L_1^*$.

**Solution:** Let $L_1$ and $L_2$ be context-free languages that are defined by grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$. In addition we require that $(V_1 - \Sigma_1) \cap (V_2 - \Sigma_2) = \emptyset$. That is, the grammars may not have any common nonterminals. Since the nonterminals may be renamed if necessary, this is not an essential limitation.

- **Union:** Let $S$ be a new nonterminal and $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1 | S_2\}, S)$. Now $L(G) = L(G_1) \cup L(G_2) = L_1 \cup L_2$. This holds, since the initial symbol $S$ may derive only $S_1$ or $S_2$, and they in turn may derive only strings that belong to the respective languages. (If the sets of nonterminals were not disjoint, this would not hold).

- **Concatenation:** The language $L_1 L_2$ is defined by the following grammar: $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1 S_2\}, S)$. $L(G) = L_1 L_2$.

- **Kleene star:** The language $L_1^*$ is defined by the following grammar: $G = (V_1 \cup \{S\}, \Sigma_1, R_1 \cup \{S \rightarrow \epsilon | SS_1\}, S)$. $L(G) = L_1^*$.

5. **Problem:** Design a context-free grammar describing the syntax of simple “programs” of the following form: a program consists of nested for loops, compound statements enclosed by `begin-end` pairs and elementary operations `a`. Thus, a “program” in this language looks something like this:

```plaintext
a;
for 3 times do
begin
  for 5 times do a;
  a; a
end.
```

For simplicity, you may assume that the loop counters are always integer constants in the range 0, . . ., 9.

**Solution:** The context-free grammars of programming languages are most often defined so that the alphabet consists of all syntactic elements (lexemes) that occur in the language. In this case numbers, `a`, and reserved words are lexemes. We divide the parsing of a program into two parts:

(a) The program text is transformed into a string of lexemes using a finite state automaton;

(b) The parse tree of the lexeme string is constructed.

The given grammar can be formalized in many ways, this is one possible interpretation:

$$G = (V, \Sigma, P, C)$$

$$V = \{C, S, N, \text{begin, do, end, for, times}, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, a\}$$

$$\Sigma = \{\text{begin, do, end, for, times}, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, a\}$$
Here the nonterminal \( S \) denotes a statement, \( C \) a compound statement, and \( N \) a number. The rules of the grammar are defined as follows:

\[
P = \{ C \rightarrow S \mid S; C \mid S \rightarrow a \mid \text{begin } C \text{ end } \mid \text{for } N \text{ times do } S \mid N \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \}\]

For example, the program in the problem text has the following parse tree:

![Parse Tree Image]

6. **Problem**: In the modern WWW-page description language XML designers can construct their own “data type definitions” (abbr. DTD), which are essentially context-free grammars describing the structure of the text or other data displayed on the page. Acquaint yourself with the notation used in this XML/DTD description language and give a context-free grammar corresponding to the following XML/DTD description:

```xml
<!DOCTYPE Book [ 
  <!ELEMENT Book (Title, Chapter+)>
  <!ATTLIST Book Author CDATA #REQUIRED>
  <!ELEMENT Title (#PCDATA)>
  <!ELEMENT Chapter (#PCDATA)>
  <!ATTLIST Chapter id ID #REQUIRED>
 ]>
```

**Solution:**

The DTD-description defines the structure for a book. There are two kinds of things in the definition: *elements* and *attributes*. The idea is that the book itself consists of the elements and attributes add some meta-information to the elements.

In general, it is not possible to express the semantics of the attributes using only context-free grammars and we need stronger *attribute grammars* for them. However, we can capture the attribute syntax with the grammar.

First, we consider the only the structure the elements. The first element definition
<!ELEMENT Book (Title, Chapter+)>
tells us that a book contains a title and a sequence of chapters. The \texttt{+} sign tells us that there has to be at least one chapter. The next line:

<!ELEMENT Title (#PCDATA)>
tells us that a title is a sequence of character data. We will abstract the data away here, and define an alphabet symbol \texttt{data} to denote any possible data string. In a real implementation we would use a lexer to identify the data blocks so that the parser of the grammar could work on the abstracted level. Finally, the line:

<!ELEMENT Chapter (#PCDATA)>
tells us that a chapter is again character data.
With these definitions we can define the book structure with the following grammar:

\[
\begin{align*}
\text{Book} & \rightarrow \text{Title} \ \text{Chapters} \\
\text{Title} & \rightarrow \text{data} \\
\text{Chapters} & \rightarrow \text{Chapter} \ \text{Chapters} \mid \text{Chapter} \\
\text{Chapter} & \rightarrow \text{data} \\
\end{align*}
\]

Now we extend this grammar to coincide with the XML syntax. A syntactic element \texttt{A} starts with an opening tag \texttt{<A>} and ends with the corresponding closing tag \texttt{</A>}. When we add these to the grammar, we get:

\[
\begin{align*}
\text{Book} & \rightarrow \langle \text{Book} \rangle \ \text{Title} \ \text{Chapters} \ \langle /\text{Book} \rangle \\
\text{Title} & \rightarrow \langle \text{Title} \rangle \ \text{data} \ \langle /\text{Title} \rangle \\
\text{Chapters} & \rightarrow \text{Chapter} \ \text{Chapters} \mid \text{Chapter} \\
\text{Chapter} & \rightarrow \langle \text{Chapter} \rangle \ \text{data} \ \langle /\text{Chapter} \rangle \\
\end{align*}
\]

The syntax for attributes in XML is that we add them inside the opening tag. An attribute consists of a name-value pair \texttt{name = value}:

\[
\begin{align*}
\text{Book} & \rightarrow \langle \text{Book} \ \text{BookAttributes} \rangle \ \text{Title} \ \text{Chapters} \ \langle /\text{Book} \rangle \\
\text{Title} & \rightarrow \langle \text{Title} \rangle \ \text{data} \ \langle /\text{Title} \rangle \\
\text{Chapters} & \rightarrow \text{Chapter} \ \text{Chapters} \mid \text{Chapter} \\
\text{Chapter} & \rightarrow \langle \text{Chapter} \ \text{ChapterAttributes} \rangle \ \text{data} \ \langle /\text{Chapter} \rangle \\
\text{BookAttributes} & \rightarrow \text{author} = \text{data} \\
\text{ChapterAttributes} & \rightarrow \text{id} = \text{data} \\
\end{align*}
\]

\footnote{The symbols written with \texttt{\uline{}} are non-terminals while those in \texttt{\bf{bold}} are terminals.}