In this exercise sheet we work with the syntax and semantics definitions for the Boostan language.

Submit your solution by uploading it as PDF in ILIAS.

**Exercise 1: Boostan Grammar**

**4 Points**

In this exercise you should propose a syntax for the Boostan programming language. State a context-free grammar $G_{Boostan} = (\Sigma_{Boostan}, N_{Boostan}, P_{Boostan}, S_{Boostan})$ such that a word of the generated language is a program of (your version of) the Boostan language.

In the lecture slides we propose the grammar $G_I = (\Sigma_I, N_I, P_I, S_I)$ for integer expressions, where $\Sigma_I = \{-, +, \ast, \div, \%, (,), 0, \ldots, 9, a, \ldots, z, A, \ldots Z\}$, $N_I = \{X_{iexpr}, X_{num}, X_{num'}\}$, $S_I = X_{iexpr}$ and the following derivation rules.

$$
P_I = \left\{ \begin{array}{l}
X_{iexpr} \rightarrow (X_{iexpr}) \\
X_{iexpr} \rightarrow X_{iexpr} \\
X_{iexpr} \rightarrow X_{iexpr} + X_{iexpr} \\
X_{iexpr} \rightarrow X_{iexpr} - X_{iexpr} \\
X_{iexpr} \rightarrow X_{iexpr} \ast X_{iexpr} \\
X_{iexpr} \rightarrow X_{iexpr} \div X_{iexpr} \\
X_{iexpr} \rightarrow X_{iexpr} \% X_{iexpr} \\
X_{iexpr} \rightarrow X_{var} \\
X_{iexpr} \rightarrow X_{num} \\
X_{num} \rightarrow 0X_{num'} \ldots 9X_{num'} \\
X_{num'} \rightarrow 0X_{num'} \ldots 9X_{num'} | \varepsilon \\
X_{var} \rightarrow aX_{var'} \ldots zX_{var'} | AX_{var'} \ldots ZX_{var'} \\
X_{var'} \rightarrow aX_{var'} \ldots zX_{var'} | AX_{var'} \ldots ZX_{var'} | 0X_{var'} \ldots 9X_{var'} | \varepsilon
\end{array} \right\} \bigcup P_I
$$

Next, we proposed the grammar $G_B = (\Sigma_B, N_B, P_B, S_B)$ for Boolean expressions, where $\Sigma_B = \Sigma_I \cup \{!, \&\& \rightarrow, ==, >=, <=, <, >, <=, >=\}$, $N_B = N_I \cup \{X_{bexpr}\}$, $S_B = X_{bexpr}$ and the following derivation rules.

$$
P_B = \left\{ \begin{array}{l}
X_{bexpr} \rightarrow (X_{bexpr}) \\
X_{bexpr} \rightarrow !X_{bexpr} \\
X_{bexpr} \rightarrow X_{bexpr} \&\& X_{bexpr} \\
X_{bexpr} \rightarrow X_{bexpr} || X_{bexpr} \\
X_{bexpr} \rightarrow X_{bexpr} == X_{bexpr} \\
X_{bexpr} \rightarrow X_{iexpr} == X_{iexpr} \\
X_{iexpr} \rightarrow X_{iexpr} > X_{iexpr} \\
X_{iexpr} \rightarrow X_{iexpr} < X_{iexpr} \\
X_{iexpr} \rightarrow X_{iexpr} \rightarrow X_{iexpr}
\end{array} \right\} \bigcup P_I
$$

We propose that you use $\Sigma_{Boostan} = \Sigma_B \cup \{\text{while}, \text{if}, \text{else}, \{\}, ;, :=\}$ and your language should have the following properties.
• There should be a while statement, an if-then-else statement and an assignment statement.
• The concatenation of statements should be a statement.
• A program should be a statement and we do not need statements for declaring variables.

**Exercise 2: Derivation Tree**

Give a derivation tree for the grammar \( \mathcal{G}_1 \) and the word \( 15 + a + 4 \).

**Exercise 3: Semantics of Assignment**

In the lecture we defined the semantics for the assignment statement. In this exercise we use this definition and compute for statements the corresponding relations.

(a) Let \( V = \{x, y\} \), \( \mu(x) = \mathbb{Z} \) and \( \mu(y) = \mathbb{Z} \).
Write down the relation \( [x := x-y] \). Give a pair of states \( (s_1, s_2) \in [x := x-y] \), and give a pair of states \( (s_3, s_4) \notin [x := x-y] \).

(b) Let \( V = \{x, y\} \), \( \mu(x) = \{\text{true}, \text{false}\} \) and \( \mu(y) = \{\text{true}, \text{false}\} \).
Write down the relation \( [x := \neg y] \) by listing all elements of the relation explicitly.

In the next part of the lecture, we will continue to define the semantics of Boostan formally. For this purpose, we will use the **reflexive transitive closure** of a relation. The exercises below should make you familiar with that term.

Given a set \( X \), a **binary relation** over \( X \) is a subset of the Cartesian product \( X \times X \). We call a binary relation \( R \) reflexive if for all \( x \in X \) the pair \((x, x)\) is an element of \( R \). We call a binary relation \( R \) transitive if for all \( x, y, z \in X \) the following property holds:

If \((x, y) \in R \) and \((y, z) \in R \) then \((x, z) \in R \).

**Exercise 4: Reflexivity and Transitivity**

State for each of the following relations if the relation is reflexive and if the relation is transitive.

(a) The “strictly smaller” relation over integers, \( R_a = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x < y\} \)

(b) The relation \( \text{win}_{\text{RPS}} \) over the set \( \{\text{Rock}, \text{Paper}, \text{Scissors}\} \).
\( R_b = \{(\text{Rock}, \text{Scissors}), (\text{Scissors}, \text{Paper}), (\text{Paper}, \text{Rock})\} \)

(c) \( R_c = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid (x - y) \text{ mod } 42 = 0\} \)

(d) \( R_d = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid (x - y) = 2 \text{ or } (y - x) = 2\} \)
Given a binary relation $R$ over the set $X$, the reflexive transitive closure, denoted $R^*$, is the smallest relation such that $R \subseteq R^*$, $R^*$ is reflexive and $R^*$ is transitive. We note that in this context “smallest” means that there is no strict subset that has the same properties and that this minimum is indeed unique since reflexive and transitive relations are closed under intersection.

**Exercise 5: Reflexive Transitive Closure**  
2 Points  
Compute for each of the four relations above the reflexive transitive closure.