Tutorial for Program Verification
Exercise Sheet 21

In this exercise sheet we work with termination and ranking functions.
Submit your solution by uploading it as PDF in ILIAS.

Exercise 1: Termination
2 Bonus Points

In the lecture, we discussed different properties of programs. Besides our definition of termination, we here give formal definitions of three other properties. In each case, we consider a program $P$ with a CFG $(\text{Loc}, \Delta, \ell_{\text{init}}, \ell_{\text{ex}})$.

(a) We say that $P$ can reach the exit location if there exists a finite execution, such that the first configuration $(\ell, s)$ is initial, and the last configuration is $(\ell_{\text{ex}}, s')$ for some state $s'$.

(b) We say that $P$ can stop if there exists a reachable configuration $(\ell, s)$ such that there exists no configuration $(\ell', s')$ and statement $st$ with $(\ell, st, \ell') \in \Delta$ and $(s, s') \in [st]$.

(c) We say that $P$ always reaches the exit location if there exist no infinite executions, and all finite executions end in a configuration $(\ell', s')$ where we either have a successor (i.e., there exists a configuration $(\ell'', s'')$ and statement $st$ with $(\ell', st, \ell'') \in \Delta$ and $(s', s'') \in [st]$) or we have that $\ell'$ is $\ell_{\text{ex}}$.

(d) We say that $P$ always stops (resp. $P$ terminates) if there exist no infinite executions.

In this exercise, you should give programs that differentiate between these definitions. In particular, for each of the following pairs, give a program such that one definition holds but the other does not. Explain which of the definitions holds and why.

(a) $P$ can reach the exit location vs. $P$ can stop

(b) $P$ can stop vs. $P$ always stops

Exercise 2: Ranking Functions
5 Points

For each of the following programs, state whether it (always) terminates or not. If it terminates, give a ranking function for each loop in the program. If it may not terminate, give an infinite execution of the program.
| 1  | while (x > 0) {          | 2  | while (x > 0) {          | 3  | while (x > 0) {          |
| 2  |     while (y > 0) {     | 4  |       if (y > 0) {       | 5  |       if (y > 0) {       | 6  |       if (y > 0) {       |
| 3  |       y := y - 1;      | 4  |           y := y - 1;   | 5  |           y := y - 1;   | 6  |           y := y - 1;   |
| 4  | } else {               | 5  | } else {                | 6  | } else {               | 7  | } else {               |
| 5  |       x := x - 1;      | 6  |             x := x - 1; | 7  |             x := x - 1; | 8  |             x := x - 1; |
| 6  | havoc y;               | 7  | havoc y;                | 8  | havoc y;               | 9  | havoc y;               |
| 7  | }                     | 8  | }                      | 9  | }                     |

Listing 1: Program P₁  
Listing 2: Program P₂  
Listing 3: Program P₃

**Hint:** For simple loops is often convenient to use a function whose range is \( \mathbb{N} \) and the strictly greater than relation \( > \) on natural numbers. For more complex loops, this is sometimes not sufficient but we can use instead a function \( f : S_{\mathbb{V},\mu} \to \mathbb{N} \times \ldots \times \mathbb{N} \) whose range are \( n \)-tuples of natural numbers and the *lexicographic order* \( >_{\text{lex}} \) that we define as follows.

\[(m_1, \ldots, m_n) >_{\text{lex}} (m'_1, \ldots, m'_n) \text{ iff there exists } i \in \{1, \ldots n\} \text{ such that } m_i > m'_i \]

and for all \( k \in \{1, \ldots i - 1\} \) the equality \( m_k = m'_k \) holds

If a function with that signature together with the order \( >_{\text{lex}} \) is a ranking function, it is often called a *lexicographic ranking function*. 