Dr. Matthias Heizmann
Tanja Schindler
Dominik Klumpp

Hand in until 10:00 on May 20, 2019
Discussion: May 20, 2019

Tutorial for Program Verification

Exercise Sheet 6

Exercise 1: Boogie

3 Points

Implement the following programs in Boogie[1].

(a) Implement a procedure with signature `gcd(x : int, y : int) returns (div : int)` that takes two (mathematical) integers x, y and, if they are both not equal to 0, computes their greatest common divisor z. The algorithm may only make use of addition and subtraction, but not use multiplication, division or modulo.[2]

(b) Implement a procedure with signature `prime(x : int) returns (isprime : bool)` that takes an integer x and, if $x > 0$, returns **true** if and only if x is a prime number.

(c) Implement a procedure with signature `pow(x : int, y : int) returns (exp : int)` that takes two integers x, y, and, if y is greater than 0, returns x^y.

You can use the Boogie interpreter Boogaloo[3] to test your program. A user manual is available[4]. The Boogie standard does not define division and modulo. In this lecuture we will consider an extension of Boogie where these two operations are defined via the SMT-LIB semantics for divison and modulo (Euclidean division). In the Boogaloo interpreter the syntax is `div` and `mod`. In Ultimate the syntax is `/` and `%`. In this exercise you may use the syntax that you like most.

Please submit your Boogie programs electronically (via Email)!

Exercise 2: Satisfiability of FOL Formulas

2 Points

Are the following formulas φ_i satisfiable with respect to the theory of integers $T_{\mathbb{Z}}$? If the formula is satisfiable, give a satisfying assignment.
You may use an SMT solver (e.g. Z3[5]) to solve this task.

- $\varphi_1 := \forall x, y.\ a \neq 21 \cdot x + 112 \cdot y$
- $\varphi_2 := \exists x.\ (x = 10 \cdot a + b \land a + b = 9 \land \neg \exists y.\ x = 3 \cdot y)$

---

[1] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/12/krml178.pdf

[2] Hint: https://en.wikipedia.org/wiki/Euclidean_algorithm

[3] https://comcom.csail.mit.edu/comcom/#Boogaloo

[4] https://bitbucket.org/nadiapolikarpova/boogaloo/wiki/User%20Manual

[5] https://rise4fun.com/Z3

.

Exercise 3: Boo Grammar 2 Points

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

In the lecture slides we propose the grammar $\mathcal{G}_{\mathsf{I}} = (\Sigma_{\mathsf{I}}, N_{\mathsf{I}}, P_{\mathsf{I}}, S_{\mathsf{I}})$ for integer expressions, where $\Sigma_{\mathsf{I}} = \{-, +, *, /, \%, (,), 0, \ldots, 9, a, \ldots, z, A, \ldots Z\}$,
$N_{\mathsf{I}} = \{X_{iexpr}, X_{num}, X_{num'}, X_{var}, X_{var'}\}$, $S_{\mathsf{I}} = X_{iexpr}$ and the following derivation rules.

$$\begin{array}{rcl}
P_{\mathsf{I}} = \{X_{iexpr} & \rightarrow & \texttt{(}X_{iexpr}\texttt{)} \\
X_{iexpr} & \rightarrow & \texttt{-}X_{iexpr} \\
X_{iexpr} & \rightarrow & X_{iexpr}\texttt{+}X_{iexpr} | X_{iexpr}\texttt{-}X_{iexpr} | X_{iexpr}\texttt{*}X_{iexpr} | X_{iexpr}\texttt{/}X_{iexpr} | X_{iexpr}\texttt{\%}X_{iexpr} \\
X_{iexpr} & \rightarrow & X_{var} \\
X_{iexpr} & \rightarrow & X_{num} \\
X_{num} & \rightarrow & \texttt{0}X_{num'} | \ldots | \texttt{9}X_{num'} \\
X_{num'} & \rightarrow & \texttt{0}X_{num'} | \ldots | \texttt{9}X_{num'} | \varepsilon \\
X_{var} & \rightarrow & \texttt{a}X_{var'} | \ldots | \texttt{z}X_{var'} | \texttt{A}X_{var'} | \ldots | \texttt{Z}X_{var'} \\
X_{var'} & \rightarrow & \texttt{a}X_{var'} | \ldots | \texttt{z}X_{var'} | \texttt{A}X_{var'} | \ldots | \texttt{Z}X_{var'} | \texttt{0}X_{var'} | \ldots | \texttt{9}X_{var'} | \varepsilon\}
\end{array}$$

Next, we proposed the grammar $\mathcal{G}_{\mathsf{B}} = (\Sigma_{\mathsf{B}}, N_{\mathsf{B}}, P_{\mathsf{B}}, S_{\mathsf{B}})$ for Boolean expressions, where $\Sigma_{\mathsf{B}} = \Sigma_{\mathsf{I}} \cup \{!, \&\&, \|, ==>, ==, <, >, <=, >=\}$, $N_{\mathsf{B}} = N_{\mathsf{I}} \cup \{X_{bexpr}\}$, $S_{\mathsf{B}} = X_{bexpr}$ and the following derivation rules.

$$\begin{array}{rcl}
P_{\mathsf{B}} = \{X_{bexpr} & \rightarrow & \texttt{(}X_{bexpr}\texttt{)} \\
X_{bexpr} & \rightarrow & \texttt{!}X_{bexpr} \\
X_{bexpr} & \rightarrow & X_{bexpr}\texttt{\&\&}X_{bexpr} | X_{bexpr}\|X_{bexpr} | X_{bexpr}\texttt{==}X_{bexpr} \\
X_{bexpr} & \rightarrow & X_{iexpr}\texttt{==}X_{iexpr} | X_{iexpr}\texttt{<}X_{iexpr} | X_{iexpr}\texttt{>}X_{iexpr} | X_{iexpr}\texttt{<=}X_{iexpr} | X_{iexpr}\texttt{=>}X_{iexpr} \\
X_{bexpr} & \rightarrow & X_{var} \\
X_{bexpr} & \rightarrow & \textbf{true} | \textbf{false}\} \cup P_{\mathsf{I}}
\end{array}$$

We propose that you use $\Sigma_{\mathsf{Boo}} = \mathcal{G}_{\mathsf{B}} \cup \{\texttt{while}, \texttt{if}, \texttt{else}, \{, \}, \texttt{;}, \texttt{:=}\}$ 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 4: Derivation Tree 1 Point

Give a derivation tree for the grammar $\mathcal{G}_{\mathsf{I}}$ and the word $15 + a + 4$.