National University of Singapore
School of Computing
CS3234 — Logic and Formal Systems

November 4, 2004

Assignment 3

(5% of final grade)

Deadline: November 14, 2004 (Your answers may be in handwritten form)

The purpose of this assignment is to provide some exposure to verifying imperative programs with arrays. First, let us augment the programming language defined in the lecture with array expressions. This language is defined by the following grammar.

$$\begin{aligned}
S &::= n \mid x \mid (-S) \mid (S+S) \mid (S-S) \mid (S*S)\\
A &::= a[S]\\
E &::= S \mid A \mid (-E) \mid (E+E) \mid (E-E) \mid (E*E)\\
B &::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B\&B) \mid (B||B) \mid (E<E)\\
C &::= x = E \mid a = a\langle S \mapsto S\rangle \mid C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}
\end{aligned}$$

In this definition, n stands for an integer constant, x stands for any *scalar* program variable, and a stands for any *array* program variable. The nonterminals S, A, E, B, and C represent the languages of scalar expreessions, array expressions, arithmetic expressions (i.e. both scalars and arrays), boolean expressions, and program commands, respectively. We shall consider that the arrays of our language are *infinite* (i.e. have an infinite number of elements), and *do not need to be declared*. We note that array expressions are of the form $a[S]$, where S is a scalar expression that does not contain other array references. We also note that the language of commands has been augmented with array assignments, of the form $a = a\langle S \mapsto S\rangle$. For example, $a = b\langle 3 \mapsto 5\rangle$ is a valid array assignment. This assignment has, on the right hand side, the expression $b\langle 3 \mapsto 5\rangle$, where b is an array. The expression evaluates to a *new array*, whose values are the same as those of array b, except the element of rank 3, whose value is 5. The assignment $a = b\langle 3 \mapsto 5\rangle$ assigns the entire arrray $b\langle 3 \mapsto 5\rangle$ to a. This notation is rather unusual, and may seem complicated at first sight, but will in fact result in simpler proof rules.

Here are two examples of using our array notation. The left column contains statements of the C programming language, while the right column contains the corresponding commands in our array assignment notation.

$a[i] = a[i+1]$	translates into	$a = a\langle i \mapsto a[i+1]\rangle$
$a[a[i]] = x$	translates into	$y = a[i]; a = a\langle y \mapsto x\rangle$

Exercise 1

Devise a partial correctness calculus for the augmented programming language given above. Note that the pre- and post-conditions of your Hoare triples need to be first order formulas. in order to allow statements about arrays. For example, stating that the elements of array a between ranks 1 and n are sorted could be done in the following way

$$\forall w\,(1 \leq w \leq n-1 \rightarrow a[w] < a[w+1])\ ,$$

where w is an *auxilliary*, quantified variable, that does not appear in the program at hand.

Exercise 2

Choose one of the following sorting algorithms: bubble-sort, selection-sort, insertion-sort, or shell-sort, and implement it in the language given above. You will assume that, at the start of your program, program variable `n` has already been assigned with a strictly positive value, and that the first `n` elements of the array `a` already contain the values to be sorted.

Exercise 3

Write a Hoare triple that specifies the partial correctness of your program.

Exercise 4

Write a proof tableau that proves the specification of Exercise 3.