Programming Languages

Spring 2009

January 14, 2009

Note: This exam is open books and open notes.

Axiomatic Semantics [25 Points]

In this problem we will use the goto-language that we introduced in class to study total correctness (Chpt. 2 of Gumb's book).

Consider the following program

$$
\begin{array}{ll}
1: & start\ \ n \geq 0 \wedge \forall I(0 \leq I < n \rightarrow a[I] \geq 0) \\
2: & i = 1 \\
3: & b[0] = a[0] \\
4: & assert \ldots \\
5: & if\ \ i = n\ \ goto\ 9 \\
6: & b[i] = b[i-1] + a[i] \\
7: & i = i + 1 \\
8: & goto\ 4 \\
9: & stop \ldots
\end{array}
$$

Answer the following questions:

- provide the post-condition for the program; it should have the form:

$$\forall I(0 \leq I < n \rightarrow b[I] = \ldots)$$

- determine the loop invariant (line 4)
- determine all paths in the program
- determine for each path the path substitution, path condition, and verification condition; precisely discuss the validity of each verification condition
- determine a floydian expression for the loop in the program; establish the well-foundedness and the termination conditions and precisely discuss their validity.

Operational Semantics [15 Points]

Consider the following syntax for a case statement in an imperative language

$$\begin{array}{lcl} \langle statement \rangle & ::= & \textbf{switch } \langle identifier \rangle \ \langle cases \rangle \\ \langle cases \rangle & ::= & \textbf{case } \langle number \rangle : \langle statement \rangle \ \langle cases \rangle \\ & | & \textbf{default } \langle statement \rangle \end{array}$$

The behavior is as follows:

- the variable mentioned in the switch statement is evaluated
- the value of the variable is compared to the number indicated in each **case**; as soon as a match is found, the corresponding statement is executed and **the switch statement is terminated**;
- if no matches are found, then the statement associated with the **default** is executed

Note that, differently from C, as soon as a matching case is found, the switch statement is terminated (i.e., there is an implicit `break` associated with each case).

Provide sequents describing the operational semantics for both the switch statement as well as the various components of $\langle cases \rangle$.

Denotational Semantics [60 Points]

Consider the following syntax for an imperative language:

```
<program>      ::=  main { <declarations> ; <statement> }
<declarations> ::=  epsilon
                |   guard <boolexpression> => <statement> ; <declarations>
<statement>    ::=  <identifier> = <expression>
                |   <statement> ; <statement>
                |   if <boolexpression> then <statement> else <statement>
                |   repeat <statement> until <boolexpression>
<boolexpression> ::= <expression> < <expression>
                |    <expression> == <expression>
                |    not <boolexpression>
                |    <boolexpression> or <boolexpression>
<expression>   ::=  <number>
                |   <identifier>
                |   <expression> * <expression>
```

The guards define *daemons*. A guard is composed of a condition and a statement. The guards are defined at the beginning of the program. The guards are evaluated after the execution of each assignment statement. The condition of the guard is automatically evaluated after each assignment statement; if the

condition becomes true, then the statement of such guard should be immediately executed. This behavior should happen after each assignment statement. Once a guard has been successfully applied (i.e., it "fired"), it should be discarded.

Note:

- we assume that variables can be used without declarations, and the variables can only store natural numbers
- there is no need to perform any error checking

Provide the *complete* denotational semantics for this language. Make sure to clearly describe all the semantic algebras used and provide all valuation functions.