Programming Languages

Qualifying Examination
Spring 2015

November 19, 2014

Part 1: Denotational Semantics [40 Pts]

Consider the following syntax of an imperative language:

```
<vardecl>     ::= var <id>
                |   var <id> ; <vardecl>
<block>       ::= begin <vardecl> <statement> end
<statement>   ::= <statement> ; <statement>
                |  <id> = <expression>
                |  define procedure <id> (<id> <id>) <block>
                |  if <expression> then <statement>
                |  call <id> (<expression> <expression>)
```

Note the following features of the language:

- Variables have to be declared (let us assume that Nat is the only data type)
- One can dynamically define procedures using the `define procedure` statement
- Procedures have always two arguments.

Provide the denotational semantics for the language under the following assumptions:

1. Don't worry about the expressions (assume a standard evaluation function as in the book)
2. Non-local references are handled via dynamic scoping
3. The first parameter is always communicated using call-by-reference

4. The second parameter is always communicated using call-by-name
5. Do not worry about type checking expressions; but you should allow procedures to be passed as arguments.

Part 2: Axiomatic Semantics [40 Pts]

Write a goto-program that takes as input three non-negative integer numbers and determines the minimum of the three; the only operations that you are allowed to use are

- increment (by one) and decrement (by one)
- conditional jumps if a variable assumes valuze zero
- unconditional jumps
- assignments that copy the value of one variable into another variable
- assignments that assign a constant number to a variable

Prove total correctness (using Floyd's Method) of the program.

Part 3: Language Design [20 Pts]

Consider the construct with the syntax

```
<statement>  ::= forall <id> in <number> ... <number> do <statement>
```

which captures a for loop with iterations that can be executed independently in parallel. For example

```
forall x in 1..5 do
    a[x] = a[x] + x
```

executes the statement `a[x] = a[x] + x` for all values of `x` in the range from 1 to 5 in parallel.

Answer the following questions:

1. Provide the denotational semantics for this particular statement; your semantics specification should meet the following requirements
    - the behavior should be deterministic

- the iterations of the loop should happen concurrently—thus, the effects of each iteration are not visible to the other iterations until they are all finished
- the loop is valid only if the first value of the range is less or equal to the last value

2. Discuss how this construct could be implemented, addressing clearly the issue of deterministic outcome.