

Programming Languages

Spring 2014 Qualifiers

November 20, 2013

This exam is open books and open notes. No electronic aids are allowed.

Question 1: Principles of Programming Languages

Introduction

The focus of this question is on the design of a simple programming language to manipulate directed labeled graphs. A Directed Labeled graph is described by $G = (N, E, \ell)$ where N is a finite set of nodes, E is the set of edges (i.e., $E \subseteq N \times N$) and $\ell : E \rightarrow \mathbb{N}$ assigns a label to each edge (for simplicity, just a natural number). We assume that each node has a unique name.

Part A: Language Design [15 Points]

Design a convenient syntax for an imperative language to manipulate graphs. The language should be declaration-free (no need to declare variables and variable types). Keep the language simple, in the style of the simple languages in Schmidt's book (e.g., no blocks, no procedures). The language should handle the following types of data:

- Natural numbers
- Nodes
- Edges
- Graphs

Because of some of the operations discussed below, you may also need to deal with sets or lists of objects (e.g., sets of edges). Thus, variables should be able to store any of these types of data.

Your language should have primitive operations to perform the following tasks:

- create empty graphs;
- add or remove nodes or edges to an existing graph;

- retrieve or modify the label of an edge;
- find all incoming edges or outgoing edges from a given node;
- determine all paths between two nodes in a graph.

The language should allow basic tests on a graph (e.g., presence/absence of a certain node or edge). The language should also provide iterators that are specific to a graph—in particular

- Iterate over the nodes/edges from a given graph
- Iterate over all edges entering/leaving a given node

Provide a formal description (as a grammar) of the syntax for this language.

Part B: Denotational Semantics [35 Points]

Provide the denotational semantics for the language you designed in Part A. In particular, provide clear description of the semantic algebras for nodes, edges, and graphs and the clear valuation functions for expressions and statements dealing with graphs (including the iterators).

Question 2: Axiomatic Semantics [35 Points]

Let us consider the following program—which assumes that the input is an array of n elements, called in (with elements from $in[0]$ to $in[n - 1]$):

```

{ $n > 1$ }
 $a = in$ ;
 $i = 1$ ;
{ $p$ }
while ( $i < n$ ) do
     $a[i] = a[i] + a[i - 1]$ ;
     $i = i + 1$ 
end
{ $\forall X (0 \leq X < n \Rightarrow a[X] = f(in, X))$ }

```

- Complete the postcondition according to the given pattern; you need to replace the function f with a formula that depends only on the content of the original array in and X ;
- Provide the loop invariant;
- Provide the first 2 steps necessary for proving the partial correctness of the loop part of this program;
- Provide the steps necessary to prove total correctness.

Question 3: Programming Languages Implementation [15 Points]

Consider a programming language that makes heavy use of recursion (as in functional and logic programming). One of the common problems with recursive procedures is the heavy cost of allocating/deallocating activation frames on the stack. Discuss possible optimizations that could be realized by restricting the type of recursion allowed (e.g., tail recursion).