Programming Languages

Fall 2013 Qualifiers

May 8, 2013

Question 1

Introduction

In this question we will consider a simplified version of the Linda framework. Linda provides a blackboard, where concurrent threads can read and write data. Data in the blackboard are stored in the form of tuples; a tuple is a list of values $(v_1, \ldots, v_n)$ where each v_i, for simplicity, is a Nat. The blackboard may store tuples of arbitrary lengths.

The operation **out** allows us to write a tuple in the blackboard

$$\mathbf{out}(v_1, \ldots, v_n)$$

The read operation is *associative*; the program provides a pattern of a tuple, and a tuple matching the pattern is identified and returned as result. The pattern is composed of values that should be present in the tuple, and variables for values that are unspecified. For example, a pattern of the form $(5, ?x, 10, ?x)$ will match any tuple of length 4 in the blackboard that has 5 in the first position, 10 in the third position, and the value in the second and fourth positions are identical. The variable `x` will be assigned such value in the second/fourth positions.

We will consider two operations for reading the blackboard: the **in** operation poses a pattern, selects one matching tuple in the blackboard (assigning the variables in the pattern as needed) and removes the tuple from the blackboard. The operation **read** has the same behavior, except that the matching tuple is left in the blackboard.

If no tuple is present matching the pattern posed by **in**/**read**, then a fail result is returned by the operation.

Part A [40 Points]

Consider the following syntax for a language using Linda:

```
<program>    ::= <statement>
<statement>  ::= <statement> ; <statement>
```

```
              |   <identifier> = <expression>
              |   out(<explist>)
              |   iffalse in(<pattern>) then <statement>
              |   iffalse read(<pattern>) then <statement>
<explist>    ::= <expression>
              |   <expression> , <explist>
<pattern>    ::= <expression>
              |   ?<identifier>
              |   <expression> , <pattern>
              |   ?<identifier> , <pattern>
<expression> ::= <identifier>
              |   <number>
              |   <expression> + <expression>
```

Provide the complete denotational semantics. Note that the `iffalse` is a test that triggers the execution of the statement if the **in**/**read** operation fails to find a matching tuple in the blackboard.

Part B [30 Points]

The real Linda is meant to use Blackboard as a communication mechanism among threads. Each access is implemented as a transaction (i.e., it is atomic). An **in**/**read** operation without a matching tuple instead of failing suspends, waiting for another thread to post a matching tuple.

Discuss an efficient implementation of Linda focusing on the following aspects:

- How to handle threads, with focus on interacting with the blackboard (suspending, ensuring atomic behavior)
- How to implement tuples so that access is as efficient as possible (note, a sequential comparison between the pattern and each tuple is **not** a viable option)

Question 2 [30 Points]

Let us consider the following program:

```
$\{x > 1 \wedge \exists A(x = \ldots)\}$
$i = 1$
$a = 1$
$\{p\}$
while $(a \neq x)$ do
    $i = i + 1$
    $a = i * i * i$
end
$\{i = \ldots\}$
```

- Complete the precondition and postcondition according to the given patterns; remember that your programs are dealing only with natural numbers
- Provide the loop invariant
- Provide the steps necessary for proving the partial correctness of the loop part of this program.