

# Final Exam

CSC 7101, Spring 2005

10 May 2005

Read the whole exam first (there are 8 pages) and plan your time. You have 1 hour 50 minutes to complete all the questions. There are a total of 100 points. The exam is open notes but closed neighbors. Good luck!

**Name:**

1. (15 pts)

Answer the following short questions. For yes/no answers explain your choice.

(a) (5 pts) Is it possible to translate an arbitrary attribute grammar into an attribute grammar that uses only synthesized attributes? You can assume that the computation for evaluating an attribute can be arbitrarily complex.

(b) (5 pts) Is the following Hoare triple operationally valid?

```
{ x = 0 } do x < 0 -> x = x+1 [] x >= 0 -> x = x-1 od { x = 1 }
```

(c) (5 pts) What is the most general type of the following ML function?

```
fun foldr f b nil      = b
  | foldr f b (h::t) = f (h, foldr f b t)
```

2. (20 pts)

Consider a standard programming language (say, C or Pascal). Suppose we want to identify assignments in which a constant value is assigned to a variable. E.g., for the two assignments

```
x = 4 * 5;  
y = 2 * x + 2;
```

we would like to recognize that  $x$  and  $y$  are assigned the constant values 20 and 42, respectively. (This information would allow us to generate more efficient code for these assignments by calculating the constant value at compile time.) Note that there may be other code in between these two assignments and that the second assignment may be, e.g., inside a loop. We would like to recognize  $y$  as constant as long as the value of  $x$  used in the assignment to  $y$  comes from a unique, constant assignment to  $x$ .

Briefly explain how you would design an attribute grammar to identify constant expressions and constant assignments. Explain which attributes you need for expressions and statements to compute this information, and how their values will be synthesized or inherited.

If you believe that constant assignments cannot be recognized by an attribute grammar, explain why not.

3. (15 pts)

Consider the following loop statement:

$$\langle stmt \rangle ::= \text{while2 } \langle be \rangle \text{ do } \langle stmt \rangle \text{ end}$$

where  $\langle be \rangle$  is a boolean expression. The `while2` loop works like the `while` loop except that the loop body is executed at most twice. I.e., after the second iteration, the loop terminates regardless of whether the loop condition  $\langle be \rangle$  is true or false.

Define the operational semantics of the `while2` loop. Briefly explain your rules in English.

4. (15 pts)

Consider again the `while2` loop.

(a) (5 pts) Do we need a loop invariant for the `while2` loop? If no, why not? If yes, how is it different from the loop invariant for the corresponding `while` loop?

(b) (10 pts) Define the axiomatic semantics of the `while2` loop. Briefly explain your rule in English.

5. (15 pts)

Consider the following ML datatype declaration:

```
datatype Term = Fun of string * Term list
              | Var of string * int
```

Design a C++ or Java class hierarchy that implements the same data structure. Only show the inheritance hierarchy and the data fields of the classes. You don't need to show the constructors. Do not use unions and design the data structure such that code operating on terms does not need to use dynamic type checks (such as `typeid` in C++, `instanceof` in Java, or testing a field in the object that indicates whether a term is a function or a variable).

6. (20 pts)

The following program takes a positive integer  $n$  as input, computes the factorial function  $n!$ , and returns the result as the value of  $f$ . Recall that  $0! = 1$  and  $n! = 1 * \dots * n$  for  $n > 0$ .

Find the loop invariant and a termination function and prove the **total** correctness of this program.

```
{n ≥ 0}
f := 1;
i := n;
while i > 1 do
    f := f * i;
    i := i - 1
end
{f = n!}
```

Note that the algorithm used in this program is different from the factorial function that was on the homework!

Clearly separate the proof into the three parts before, in, and after the while loop, and indicate the verification conditions resulting from the rule of consequence.

(Space for answering question 6.)