Programming Language Technology

Exam

11 March 2013 at 8:30–12:30 in V

Course codes: Chalmers DAT151, GU DIT230 (as re-exam also: Chalmers TIN321, DAT150; GU DIT229)

Teacher: Aarne Ranta (tel. 1082)

Grading scale: Max = 60p, VG = 5 = 48p, 4 = 36p, G = 3 = 24p.

Exam review: Tuesday 9 April at 11:00-12:00 in office 6106, EDIT building.

Aids: an English dictionary.

Questions requiring answers in code can be answered in any of: C, C++, Haskell, Java, or precise pseudocode. Text in the answers can be in any of: Danish, Dutch, English, Estonian, Finnish, French, German, Icelandic, Italian, Norwegian, Spanish, and Swedish.

For any of the six questions, an answer of maximally one page should be enough.

Question 1. Write a BNF grammar that covers the following constructs of a C-like language (the same as discussed during the course):

- function definitions, such as `int g (int x, double y){return f(y);}`
- function calls, such as `f(x,1)`, and return statements, such as `return x;`
- variable expressions, such as `x`, and integer literals, such as `24`
- types: `int` and `double`

You may use the BNFC categories `Double`, `Integer`, and `Ident` as well as list categories and `terminator` and `separator` rules and `coercions` macros. (10p)

Question 2. Show the parse tree and the abstract syntax tree of the function definition

```
int main () {return foo(3,f(21),g()) ;}
```

in the grammar that you wrote in Question 1. (10p)

Question 3. Write the typing rule for function calls (with any number of arguments, including zero) in an imperative C-like language. Make sure to check everything that a compiler should check. (5p)

Write the interpretation rule for function calls in a C-like language. Make the environment explicit. Don't forget that expressions may have side effects! (5p)

Question 4. Consider the language $X*$, i.e. sequences of symbol X. Write two context-free grammars for it: one left-recursive and one right-recursive (4p). With both grammars, trace the LR parsing (i.e. the shift and reduce actions) of the string $XXXX$ (4p). What is the difference in stack size needed for parsing with the two grammars? (2p)

Question 5. Consider the following sequence of statements, assumed to appear in the body of a function with no arguments.

```
int i = 7 ;
int j = 2 + i ;
while (j > i) {
  j = j - 1 ;
  }
```

Write the corresponding JVM assembly code (Jasmin), showing clearly which instructions result from which statements/expressions (5p). Show the evolution of the stack after each instruction and each iteration of the loop (4p). How much stack space and variable storage is needed to execute the code? (1p) You don't need to remember the names of the instructions precisely, but only what they do and what arguments they take.

Question 6. Write the operational semantic rules for function abstractions and applications in call-by-value lambda calculus, using closures (6p).

Show the execution tree for the expression `(\x -> \y -> y) 2 3`, strictly following your operational semantic rules (4p).