Programming Language Technology

Exam, 15 April 2015 at 08.30–12.30 in M

Course codes: Chalmers DAT150/151, GU DIT231. As re-exam, also TIN321 and DIT229/230.
Teacher: Andreas Abel (tel. +49 176 400 333 23)

Grading scale: Max = 60p, VG = 5 = 48p, 4 = 36p, G = 3 = 24p.
Allowed aid: an English dictionary.
Exam review: *will be announced on* `plt-2015-lp3` *mailing list*

Please answer the questions in English. Questions requiring answers in code can be answered in any of: C, C++, Haskell, Java, or precise pseudocode.

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

Question 1 (Grammars): Write a BNF grammar that covers the following kinds of constructs in Java/C/C++:

- Statements:
  - `while` loops
  - `if` statements with `else`
  - statements formed from expressions by adding a semicolon `;`
- Expressions:
  - identifiers
  - integer literals
  - preincrements (`++x`) or postincrements (`x++`) of identifiers (`x`)

An example statement is shown in question 2. You can use the standard BNFC categories `Integer` and `Ident`. (10p)

Question 2 (Trees): Show the parse tree and the abstract syntax tree of the statement

```
while (x++) if (cond) ++x ; else 5 ;
```

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

Question 3 (Typing and evaluation):

1. Write syntax-directed typing rules for the *statements* of Question 1: `while`, `if-else`, statements from expressions. You can assume a typing relation $\Gamma \vdash e : t$ for *expressions* e and refer to it. (5p)

2. Write syntax-directed interpretation rules for the statements of Question 1. The environment must be made explicit. You can assume an interpreter for expressions $\gamma \vdash e \Downarrow \langle v; \gamma' \rangle$ and refer to it. (5p)

Question 4 (Parsing): Consider the language S^*, i.e., (possibly empty) sequences of symbol S. 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 SSS. (4p) What is the difference in stack size needed for parsing with the two grammars? (2p)

Question 5 (Compilation):

1. Write compilation schemes for each of the grammar constructions in Question 1 generating JVM (i.e. Jasmin assembler). It is not necessary to remember exactly the names of the instructions – only what arguments they take and how they work. (6p)

2. Give the small-step semantics of the JVM instructions you used in your compilation schemes. (4p)

Question 6 (Functional languages):

1. Give the typing rules for simply-typed lambda-calculus! Simple types are given by the grammar $t ::= \texttt{int} \mid t \to t$. (5p)

2. Give a type and a typing derivation for the term $\lambda f.\lambda x.\,((f\,x)\,x)$. (5p)