Programming Language Technology

Exam, 6 April 2016 at 08:30 – 12:30 in M

Course codes: Chalmers DAT151, GU DIT231. As re-exam, also DAT150, TIN321 and DIT229/230.
Teacher: Fredrik Lindblad, will visit around 09:30 and 11:00.

Grading scale: Max = 60p, VG = 5 = 48p, 4 = 36p, G = 3 = 24p.
Allowed aid: an English dictionary.

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 labelled BNF grammar that covers the following constructs in a C-like imperative language: A program is a list of statements. Statement constructs are:

- `if` statements with non-optional `else` branch.
- block statements (lists of statements surrounded by curly braces)
- expression statements (`E;`)

Expression constructs are:

- identifiers/variables
- integer literals
- assignments of identifiers (`x = E`)
- addition (`E + F`)
- multiplication (`E * F`)

Operator precedences and associativity should follow the C standard. You can use the standard BNFC categories `Integer` and `Ident` as well as list shorthands, and `terminator`, `separator` and `coercions` rules. (10p)

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

```
if (b) { x = y + 5 * z; } else x = 0;
```

in the grammar that you wrote in question 1. In the parse tree show the coercions explicitly. (10p)

Question 3 (Typing and evaluation):

A. Write standard typing rules or syntax-directed type-checking code (or pseudocode) for the *expression* constructs (5 constructs) of the grammar in question 1. The variable context must be made explicit. (5p)

B. Write big-step operational semantic rules or syntax-directed interpretation code (or pseudocode) for the expression constructs of the grammar in question 1. The environment must be made explicit. (5p)

Question 4 (Regular expressions):

A. Write a regular expression the recognizes the following language (and only this): A string in the language is a sequence of tokens separated by one or more space-characters. A token is either of these two forms:

- Identifier: Any letter (a-z or A-Z) followed by any number of characters which are either a letter or a digit.
- String literal: A double quote ("), followed by any sequence of characters except double quote, followed by a double quote.

Do not use any short-hand regular expression constructs for letters and digits. You may refer to `char` as a short-hand for any character and `-` for which `A - B` represents the characters in `A` but not i `B`. (5p)

B. Write a deterministic finite-state automaton (DFA) for the same language as in part A. (5p)

Question 5 (Compilation):

A. Write compilation schemes for each of the constructs (statement and expression, 8 in total) of the grammar in question 1. It is not necessary to remember exactly the names of the JVM instructions – only what arguments they take and how they work. (6p)

B. Give the small-step semantics of the JVM instructions you used in the compilation schemes in part A. (4p)

Question 6 (Functional languages): Show the big-step operational semantics rules (not as code) for a functional language with the expression constructs function application, λ-abstraction, variables, addition and multiplication. The evaluation strategy should be call-by-value. Use closures and explicit environment. (6p)

Show the derivation tree (using your operational semantics) of the evaluation of the expression

```
(\f -> x + f x) (\y -> x * y)
```

in the environment `{x := 3}`. (4p)