INJE08: Programming Paradigms
04IN1024: Programming Language Theory

Dry run WS 2014/15

Universität Koblenz-Landau, FB4
Prof. Dr. Ralf Lämmel
21.01.2015

Name, Vorname	
Matrikel-Nr.	
Studiengang	
ECTS (8 oder 6)	

Grading scheme

There are 10 questions with 0-2 points each: 2 points for a (mostly) correct and complete solution; 1 point for a somewhat reasonable solution with significant omissions or defects; 0 points otherwise. About 50 % of the points are needed to pass the exam. If you are a student enrolled according to the previous exam rules, then about 8/6*50 % of the points are needed to pass the exam.

Contents

The topics of the exam are published with the dry run and reused for the actual final and the resit as well. Of course, the topics are chosen to cover broad subjects areas. The actual questions are focused on specific competences that were emphasized in the lecture and exercised in the lab—in the edition of the course at hand.

1 Topic: *Abstract syntax*

Consider a simple expression language with constructs as follows:

- number literals (whose structure can be ignored here),
- variable identifiers (whose structure can be ignored here),
- binary, infix additions such as '$41 + 1$',
- unary, prefix negation such as '-42'.

Define the syntax of such expressions using BNF-like notation or Prolog.

2 Topic: *Operational semantics*

Identify the normal forms (values) for the expression forms of the previous question, if you were to assume a small-step semantics.

3 Topic: *Type systems*

Imagine you are providing a type system for a given programming language. Obviously, you need to provide the typing rules, eventually. How can you decompose the process of providing the type system? Name 2+ activities to be performed or entities to be defined.

4 Topic: *Lambda calculi*

Use a concise argument to show that self-application is not typeable in the simply-typed lambda calculus, i.e., no T can be found so that $\lambda x : T.\ x\,x$. Your argument should refer to the following typing rule of application:

$$\textsf{T-Application}\quad \frac{\Gamma \vdash t : U \to T \qquad \Gamma \vdash u : U}{\Gamma \vdash t\ u : T}$$

5 Topic: *Type safety*

Describe a very simple scenario in which a type system and semantics fail to collaborate in type safety.

6 Topic: *Polymorphism*

Consider the following typing rule of System F.

$$\frac{}{(\Lambda X.t)[T] \to [T/X]t}\ \text{E-TypeAppAbs}$$

What does it express? (Please, be very brief.)

7 Topic: *Object orientation*

Consider the following rules of FJ's type system:

$$\texttt{C} <: \texttt{C} \qquad \frac{\texttt{C} <: \texttt{D} \quad \texttt{D} <: \texttt{E}}{\texttt{C} <: \texttt{E}} \qquad \frac{\texttt{class C extends D \{...\}}}{\texttt{C} <: \texttt{D}}$$

What do they express? (Please, be very brief.)

8 Topic: *Denotational semantics*

Consider the following equation, as if it was part of a denotational semantics:

$$\mathit{Meaning}[\mathsf{while}\ b\ \mathsf{do}\ S] = \mathit{Meaning}[\mathsf{if}\ b\ \mathsf{then}\ S;\ \mathsf{while}\ b\ \mathsf{do}\ S\ \mathsf{else}\ \mathsf{skip}]$$

Explain how this equation fails to meet 'compositionality', which is a requirement for any equation of a denotational semantics description.

9 Topic: *Axiomatic semantics*

Consider the following formulae as they may appear in pre- and postconditions of triples:

- $a > b \;\&\&\; b > c$
- $!(a <= b) \;\&\&\; b > c$

These formulae are logically equivalent. Describe (informally) a rewrite rule so that both formulae would be syntactically equivalent by means of normalization.

10 Topic: *Concurrency*

Consider again the transition rules for CCS' composition operator:

- Com_1 $\quad \dfrac{E \xrightarrow{\alpha} E'}{E|F \xrightarrow{\alpha} E'|F}$
- Com_2 $\quad \dfrac{F \xrightarrow{\alpha} F'}{E|F \xrightarrow{\alpha} E|F'}$
- Com_3 $\quad \dfrac{E \xrightarrow{l} E' \quad F \xrightarrow{\bar{l}} F'}{E|F \xrightarrow{\tau} E'|F'}$

Even if two processes E and F could communicate (see the third rule), then transitions without communication (see first and second rules) are still feasible. Why is that?