Exam in Models of Computation TDA 183

Date: April 14, 2009, 14.00 - 17.00

Permitted aids: English-Swedish or English-other language dictionary.

Teacher: Bengt Nordström, phone 0730 79 42 89, Computer Science, University of Gothenburg and Chalmers

All solutions must be explained! It is not enough to just give a program without an explanation of why it works. The examination of the course consists of three parts: homework assignments count up to 40 points, weekly exercises count up to 20 points and this written exam count up to 140 points (20 points for each of the six first problems, 40 points for the last). You have to have 100 points in total in order to pass the course.

Solutions to the exam will be available from the homepage of the course.

1. A possible definition of the word enumerable is that a set **A** is enumerable if there is a total surjective function `f : N -> A`.

   (a) Is it important that the function is total?

   (b) Is it important that the function is surjective?

   If the answer is yes, show that the set of real numbers would be enumerable if the requirement is not satisified. If the answer is no, show how it is possible to construct a function which satisfies the requirement from a function which does not satisfy the requirement.

2. The set of all computable functions from **N** to **N** is enumerable. Prove or disprove!

3. Two questions about lambda-calculus:

   (a) What does it mean that a function `f : N -> N` is computable in lambda-calculus?

   (b) Give an example of a term in lambda-calculus which sometimes terminates, sometimes does not terminate, depending on the evaluation strategy.

4. Show that the addition function over natural numbers is primitive recursive!

5. Describe Turing's model of computation (Turing Machine)!

6. This problem is to describe a computation model for higher order primitive recursive functions. The language **X-PRF** is obtained from the language **X** – which was presented during the lectures – by the following modifications: Remove all syntax except lambda, application, variables, constants and abstraction. Then you add an operator for primitive recursion, i.e.

you add a syntactic expression `primrec e g f`, where `e`, `g` and `f` are expressions in X-PRF. This program is computed by first computing the value of `e`. If the value is `Z` then the value of the program is equal to the value of `g`. Otherwise, if the value of `e` is `(Succ a)` then the value of the program `primrec e g f` is equal to the value of `f a (primrec a g f)`. Give the formal operational semantics of the language **X-PRF**!

Good Luck!

Bengt