Exam in Models of Computation TDA 183

Date: Dec 13, 2010, 14.00 - 18.00

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

Teacher: Bengt Nordström, phone 070 600 1546, 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. 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.

Answer first the following four questions (with a proof)!

1. (15p) Is the set **Fin(N)** of all finite subsets of **N** enumerable?
2. (15p) Is the set **Pow(N)** of all subsets of **N** enumerable?
3. (15p) Is the function `f : N -> N` which returns 1 for even numbers and is undefined for odd numbers computable?
4. (15p) Is it the case that if the evaluation of a term `M` in lambda-calculus terminates with a term `N`, then all evaluations of `M` terminates with a term which is alpha-convertible to `N`?
5. (15p) Define (in a functional language) the function `iter` which has the following properties:

   ```
   iter :: Int -> (a -> a) -> a -> a
   iter n f a = f (f ... (f a)...)
   ```

   where there are `n` occurrences of `f`, `n` is non-negative.
6. (15p) Define (still in a functional language) the function `iter` without using recursion. You can assume the existence of the function `rec` which has the following definition in Haskell:

   ```
   rec 0 d e = d
   rec (n+1) d e = e n (rec n d e)
   rec _ d e = error "negative argument to rec"
   ```

7. (15p) Show how to represent the natural numbers in lambda-calculus (it is enough if you show how 0, 1 and 2 are represented) and then give a definition of the function `iter` in lambda-calculus.

8. (15p) In the language **PRF(n)** of primitive recursive functions there is an operator which is similar to the function `rec` above. Describe it by giving its abstract syntax (using the set **PRF**(n), the set of primitive recursive functions of arity n) and give its operational semantics (big step). There is no need to give syntax and semantics of any other constructs in **PRF(n)**.

9. (20p) Implement `rec` in the language **X**! Show first the standard representation of a natural number.

Good Luck!
Bengt