Exam with answers: Models of Computation TDA183 – DIT310

Date: Dec 18, 2013, 8:30 – 12:30

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

Teacher: Bengt Nordstrom, 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, weekly exercises and this written exam (where each problem is worth 20 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 and it will be possible to discuss the grading on Thursday 23 Jan at 13:30 – 14:00 in Bengt Nordstrom's office.

Prove or disprove the following statements:

1. It is possible to write a function in Haskell (or some other language)

   ```
   halt : Bool -> Bool
   ```

   which returns `true` if the input terminates.

   Answer: True. You can define

   ```
   halt x = true
   ```

   which returns true if the input terminates. (Don't confuse this problem with the halting problem!)

2. All closed expressions in lambda-calculus have a unique (up to alpha-conversion) normal form.

   Answer: No. There are closed expressions which have no normal form. The standard example is `O O`, where `O` stands for the expression `\x.xx`. (Some students confuse this with Church-Rosser's theorem.)

3. If an open expression in lambda-calculus has a normal form, then this normal form is open.

   Answer: No. Take for instance the open expression `(\xy.x) (\u.u) z` which computes to the closed expression `\u.u`

4. The full evaluation of a program in **X** always terminates if the program has a weak head normal form.

   Answer: No, take for instance the program `c (rec x = x)` which is on weak head normal form (it is an application of a constructor to an argument), but its full evaluation is not terminating (the argument does not terminate). (Some students have tried with "\y. (rec x = x)", which is fully evaluated)

5. The set of all terminating Java-programs is enumerable.

   Answer: Yes. The set of all terminating programs is a subset of the set of all programs which is enumberable since we can construct an injective function from the set of all programs to the set of natural numbers by mapping each program to the number obtained by looking at the ascii-string of the program as a bitstring and looking at that bitstring as a number in the base 2.

6. The set of all total functions `Bool -> N` is enumerable.

   Answer: Yes, we can construct a function `e : (Bool -> N) -> N`

   ```
   e (f) = exp(2, f(true)+1) * exp(3, f(false) + 1)
   ```

   which is injective by the prime factorization theorem. (Many students have noticed the similarity between the function set`Bool -> N` and the cartesian product `N x N`).

7. It is possible to write a program in **PRF** which takes any number of arguments and always return 0. Here you must use the version of **PRF** which we have used in the course, so for instance the arity of `zero` is 0, the arity of `succ` is 1 and the arity of `proj(n,i)` is `n+1` if `i<=n`.

   Answer: Yes, we can define

   ```
   z = comp(zero,[])
   ```

   The syntactic requirement for `comp` is fulfilled, since its second argument is a list of 0 n-ary functions, where 0 is the arity of the first argument. The computation proceeds as follows if we apply it to an arbitrary list `as` of numbers: `comp(zero, [])[as] -> zero[] -> 0`

Good Luck!

Bengt