Normalization by Evaluation

Course for PhD students in computer science
University of Warsaw
-
Assignment, due 31 March 2008

Peter Dybjer

Chalmers University of Technology

The level of difficulty of the following assignment ranges from easy to difficult. Please do an appropriate subset of the problems that interest you. Page numbers refer to the slides for the course. My email is `peterd@cs.chalmers.se`!

1. (a) The nbe-algorithm for monoids on p 19 returns right-leaning trees as normal forms. Change it so that it returns left-leaning trees instead!
   (b) Rewrite the algorithm on p 19 so that the model is $[a] \to [a]$ instead of $Exp\ a \to Exp\ a$! Why are elements of $[a]$ suitable as representations of the normal forms in $Exp\ a$?
   (c) Why is it possible to write a "generic" nbe-algorithm for normalizing elements in an arbitrary free monoid and also use this to decide equality? This assumes that the free monoid in question is presented "constructively". Discuss exactly what is required! Assume you have such a generic nbe-algorithm. What does it do for the free monoid $[a]$ of lists?
   (d) (p 20-22) Work out the details of the proof of correctness for the nbe-algorithm for monoids.
2. (a) Consider the monoid laws (p 15) as left-to-right rewrite rules. Prove that each term has a unique normal form with respect to this rewrite rule system! Hint: prove that the system is terminating and confluent!
   (b) Explain why the nbe-program on p 19 does not return normal forms in the sense of the rewrite system!
   (c) One can use the nbe-technique for getting an alternative proof of uniqueness of normal forms for the rewrite rule system. First, modify the nbe-algorithm so that it returns normal forms in the sense of the rewrite rule system! Then prove that e reduces to $nbe\ e$ using a similar technique as on p 22.
3. (p 24) What happens if you try to normalize group expressions with an analogous method to our method for monoid expressions? Can you decide equality in the free group in this way? Discuss!
4. Implement the bracket abstraction algorithm (p 40) in a functional programming language!
5. (a) (p 41) Reduce the combinatory version of $power\ m\ 3$ by hand
   (b) Add the combinators I and B (p 38) to the language on p 53, and extend the nbe-algorithm on p 55-56 accordingly!

(c) What happens if you extend the language on p 53 with a Y-combinator with the conversion rule $Y\ f \sim f\ (Y\ f)$?

(d) Extend the language of types on p 52 with products $a \times b$! Add combinators for pairing and projections, and the equations for projections. Do not add *surjective pairing*, however. Extend the nbe-algorithm on p 55-56 accordingly.

(e) Similarly, extend the language with sums $a + b$, injections and case analysis combinators, and extend the nbe-algorithm.

(f) Modify the algorithm on p 55-56, so that the clause for natural numbers instead is

$$[[Nat]] = (Exp\ Nat) \times N$$

where N is the type of metalanguage natural numbers!

(g) Modify the nbe-algorithm on p 55-56 so that it returns combinatory head normal forms instead of full normal forms.

(h) Define the dependent type (inductive family) `No a` of terms in normal forms of type `a`. Then write an application function

```
app : {a b : Type} -> No (a => b) -> No a -> No b
```

Note that `a => b` is the *object language* function space, whereas `->` denotes the *meta language* function space. (The above is Agda syntax, but you can do it on paper.)

6. (p 48) Work out the details of the normalization and confluence proofs for the reduction system for typed combinatory logic!
7. We explained on p 48 that nbe arises by extracting an algorithm from a constructive proof of *weak normalization*. What would happen if we instead start with a constructive proof of *strong normalization*? What would such an algorithm return?
8. Prove the statement on p 69! That is, define suitable notions of *formal inclusion* and *formal intersection* of neighbourhoods, and prove that a combinatory Böhm tree is a filter of neighbourhoods.
9. On p 65-69 we define the notions of combinatory head reduction of a combinatory Böhm tree using formal neighbourhoods. Write down the analogous definitions for head reduction and Böhm tree in the sense of the untyped lambda calculus
10. Implementation assignments for users of proof assistants for dependent types (Coq, Agda):
    (a) Implement the nbe-algorithm on p 55-56 in your favourite system!
    (b) Implement the monoid-normalization algorithm and its correctness proof!
    (c) Implement categories, functors, natural transformations, hom-functors, and prove the Yoneda lemma! Write an nbe-algorithm for the free category on a graph and connect its correctness proof to the Yoneda lemma!