CS 524

The Lambda Calculus and its Concurrent Execution:

Lecture 17

Further Reading

J. Meseguer

1. Basic Textbooks and the Lambda Calculus with Types

The original monograph by Alonzo Church, the creator of the λ-Calculus is:

- Alonzo Church, "The Calculus of Lamba-Conversion", Annal of Mathematical Studies, 6, Princeton Univ. Press, 1941.

The most comprehensive text on the λ-Calculus has been written by Barendregt:

- H. P. Barendregt, "The Lambda Calculus — its Syntax and Semantics," Elsevier, 1984.

For many applications, typed versions of the λ-Calculus are very useful. Two excellent overviews of Typed λ-Calculi can be found in:

- H.P. Barendregt, "Lambda Calculi with Types", in S. Abramsky, D.M. Gabbay and T. Maibaum, Hanbook of Logic in Computer Science, Vol. 2, Oxford U.P., 1992, pages 117 - 309
- H.P. Barendregt, W. Dekkers and R. Statman, "Lambda Calculus with Types", Cambridge University Press, 2010.

2. Substitution Calculi and Compilation to Combinators

As we have seen, substitution in the standard formulation of the λ-Calculus is quite complex and hard to implement correctly. Therefore, an entire cottage industry of more efficient and easier to implement substitution calculi, or, as they are sometimes called explicit substitution calculi have been proposed.

The original paper by de Bruijn is:

- N. G. de Bruijn, "Lambda Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Applications to the Church-Rosser Theorem", Konink. Nederl. Akademie van Wetenschappen - Amsterdam, Proc. Series A, 75, No. 5, and Indag. Math., 34, No. 5, 1972

A good survey of de Bruijn notation and other topics can be found in:

- F. Kamareddine, "Reviewing the Classical and the de Bruijn Notation for λ-calculus and Pure Type Systems," J. Logic Computat. Vol. 11, No. 3, 363-394, 2001

Several explicit substitution calculi have been proposed. Without trying to be exhaustive, the following can be mentioned:

- M. Abadi, L. Cardelli, P.-L. Curien and J.-J. Levy, "Explicit Substitutions," J. Functional Programming, 1 (4): 375-416, 1991.
- Mark-Oliver Stehr, "CINNI - A Generic Calculus of Explicit Substitutions and it Application to λ-, ς- and π-Calculi," Electronic Notes in Theor. Comp. Sci., 36, 2000

Stehr's is, to my knowledge the <u>most general</u> such calculus, since it can be applied to <u>any</u> binding operators, not just to the λ-binder.

Explicit substitution calculi are, by themselves, a good option to <u>implement</u> λ-calculi. But the other alternative is, as we have already seen, compilation into <u>combinators</u>. classically into <u>classical</u> CL combinators S, K and I. But many other possible choices of combinators and <u>supercombinators</u> are possible. An excelent book on this approach for compilation of functional languages is:

Simon L. Peyton Jones, "The Implementation of Functional Programming Languages," Prentice Hall, 1987.

But there is still another possibility, namely, <u>categorical combinators</u>, where "categorical" is used in the sense of <u>Category Theory</u>. Really? Yes, really, and is <u>very</u>

practical. The key idea is the following: the most natural and beautiful models of the simply typed lambda calculus are cartesian-closed categories. But such categories have a natural algebraic description by a (typed) signature Σ_Λ of operators. Those are the categorical combinators.

This approach was used as the basis of an abstract machine, called the Categorical Abstract Machine to compile the ML functional language into its CML implementation, which later expanded into the OCaml language.

The key paper is:

- G. Cousineau, P.-L. Curien and M. Mauny, "The Categorical Abstract Machine," Sci. of Computer Programming, 8, 173-202, 1987.

For a very detailed account of this entire approach, an excelent monograph is:

- Pierre-Louis Curien, "Categorical Combinators, Sequential Algorithms, and Functional Programming," Springer, 1993.

3. Parallel Functional Computation in the Lambda Calculus: the Big Picture

There are of course many possible formalizations of the

λ-, λ-η, and λ-η-α Calculus. We have seen the following:

1. Formalization of λ-$\eta\alpha$ in its <u>classical</u> foundation as a <u>rewrite theory</u> (parametric)

   $\mathcal{R}_{\lambda\text{-}\eta\text{-}\alpha}[N]$

   where α is <u>not executable</u>

2. Formalization of the λ-calculus in de Bruijn notation, where α becomes <u>syntactic identity</u> for closed lambda terms using de Bruijn numbers as a rewrite theory:

   $\mathcal{R}^{DB}_{\lambda}$

3. Formalization of the λ-calculus [could be trivially extended to the λ-η-calculus in CINNI notation, where α-equivalence becomes <u>syntactic identity</u> for closed λ-terms adding and extra <u>conditional equation</u> that replaces all names by a fixed one, formalized as a rewrite theory

   $\mathcal{R}^{CINNI}_{\lambda}[N]$

Leaving aside the issue of α-equivalence, all these rewrite theories have <u>isomorphic computations</u>. That is, for a <u>concrete</u> choice of a set Vars of <u>variable names</u> we have <u>isomorphisms</u> of <u>categories of conc. computations</u>:

$$\mathcal{T}^{CL}_{\mathcal{R}_{\lambda,\alpha}[Vars]} \simeq \mathcal{T}^{CL}_{\mathcal{R}^{DB}_{\lambda}} \simeq \mathcal{T}^{CL}_{\mathcal{R}^{CINNI}_{\lambda,\alpha}[Vars]}$$

where, to avoid the complexities of terms with free variables I have restricted the categories to the closed λ-terms. Note that the equation for α is not executable in $\mathcal{T}_{\mathcal{R}_{\lambda,\alpha}[Vars]}$, but it is executable in $\mathcal{T}_{\mathcal{R}^{CINNI}_{\lambda,\alpha}[Vars]}$.

So, all these models give us a precise mathematical description of the parallel computations of functional programs expressed in the λ-calculus.

A natural question to ask [as we did for Petri nets] is: what are computations in those [isomorphic] categories like? That is, do they correspond to some already known descriptions of parallel λ-calculus computations?

The answer is Yes! It has been given in:

- Cosimo Laneve and Ugo Montanari, "Axiomatizing Permutation Equivalence", Math. Struct. in Comp. Sci., (1996), vol. 6, 219-249.

The essential point of their answer is that, by adding a few additional equations E_{AD} to those defining $\mathcal{T}_{\mathcal{R}_\lambda[Vars]}$ as a category, we obtain a quotient category $\mathcal{T}_{\mathcal{R}_\lambda[Vars]}/E_{AD}$ that is isomorphic to the notion of permutation equivalence between parallel computations of the λ-calculus.