BRICS Research Series, Abstracts, 2008

December 15, 2008

This document is also available as *PostScript* and *DVI*.

References

[RS-08-6] *PostScript*, *PDF*, *DVI*.
Jacob Johannsen.
An Investigation of Abadi and Cardelli's Untyped Calculus of Objects.
June 2008.
xii+87 pp.

Abstract: We study the relationship between the natural (big-step) semantics and the reduction (small-step) semantics of Abadi and Cardelli's untyped calculus of objects. By applying Danvy et al.'s functional correspondence to the natural semantics, we derive an abstract machine for this calculus, and by applying Danvy et al.'s syntactic correspondence to the reduction semantics, we also derive an abstract machines for this calculus. These two abstract machines are identical. The fact that the machines are identical, and the fact that they have been derived using meaning-preserving program transformations, entail that the derivation constitutes a proof of equivalence between natural semantics and the reduction semantics. The derivational nature of our proof contrasts with Abadi and Cardelli's soundness proof, which was carried out by pen and paper. We also note that the abstract machine is new.

To move closer to actual language implementations, we reformulate the calculus to use explicit substitutions. The reformulated calculus is new. By applying the functional and syntactic correspondences to natural and reduction semantics of this new calculus, we again obtain two abstract machines. These two machines are also identical, and as such, they establish the equivalence of the natural semantics and the reduction semantics of the new calculus.

Finally, we prove that the two abstract machines are strongly bisimilar. Therefore, the two calculi are computationally equivalent.

[RS-08-5] *PostScript*, *PDF*, *DVI*.
Olivier Danvy and Jacob Johannsen.
Inter-Deriving Semantic Artifacts for Object-Oriented Programming.
June 2008.

ii+13 pp. Extended version of a paper to appear in WoLLIC 2008.

Abstract: We present a new abstract machine for Abadi and Cardelli's untyped calculus of objects. What is special about this semantic artifact (i.e., man-made construct) is that is mechanically corresponds to both the reduction semantics (i.e., small-step operational semantics) and the natural semantics (i.e., big-step operational semantics) specified in Abadi and Cardelli's monograph. This abstract machine therefore embodies the soundness of Abadi and Cardelli's reduction semantics and natural semantics relative to each other.

To move closer to actual implementations, which use environments rather than actual substitutions, we then represent object methods as closures and in the same inter-derivational spirit, we present three new semantic artifacts: a reduction semantics for a version of Abadi and Cardelli's untyped calculus of objects with explicit substitutions, an environment-based abstract machine, and a natural semantics (i.e., an interpreter) with environments. These three new semantic artifacts mechanically correspond to each other, and furthermore, they are coherent with the previous ones since as we show, the two abstract machines are bisimilar. Overall, though, the significance of these artifacts lies in them not having been designed from scratch and then proved correct: instead, they were mechanically inter-derived.

[RS-08-4] *PostScript*, *PDF*.
Olivier Danvy and Kevin Millikin.
Refunctionalization at Work.
June 2008.
ii+25 pp. To appear in *Science of Computer Programming*. A preliminary version is available as the research report BRICS RS-07-7.

Abstract: We present the left inverse of Reynolds's defunctionalization and we show its relevance to programming and to programming languages. We propose two methods to transform a program that is almost in defunctionalized form into one that is actually in defunctionalized form, and we illustrate them with a recognizer for Dyck words and with Dijkstra's shunting-yard algorithm.

[RS-08-3] *PostScript*, *PDF*, *DVI*.
Johan Munk.
A Study of Syntactic and Semantic Artifacts and its Application to Lambda Definability, Strong Normalization, and Weak Normalization in the Presence of State.
April 2008.
xi+144 pp.

Abstract: Church's lambda-calculus underlies the syntax (i.e., the form) and the semantics (i.e., the meaning) of functional programs. This thesis is dedicated to studying man-made constructs (i.e., artifacts) in the lambda calculus. For example, one puts the expressive power of the lambda calculus to the test in the area of lambda definability. In this area, we present a course-of-value representation bridging Church numerals and Scott numerals. We

then turn to weak and strong normalization using Danvy et al.'s syntactic and functional correspondences. We give a new account of Felleisen and Hieb's syntactic theory of state, and of abstract machines for strong normalization due to Curien, Crégut, Lescanne, and Kluge.

[RS-08-2] *PostScript*, *PDF*, *DVI*.
Gudmund Skovbjerg Frandsen and Piotr Sankowski.
Dynamic Normal Forms and Dynamic Characteristic Polynomial.
April 2008.
21 pp. To appear in ICALP '08.

Abstract: We present the first fully dynamic algorithm for computing the characteristic polynomial of a matrix. In the generic symmetric case our algorithm supports rank-one updates in $O(n^2 \log n)$ randomized time and queries in constant time, whereas in the general case the algorithm works in $O(n^2 k \log n)$ randomized time, where k is the number of invariant factors of the matrix. The algorithm is based on the first dynamic algorithm for computing normal forms of a matrix such as the Frobenius normal form or the tridiagonal symmetric form. The algorithm can be extended to solve the matrix eigenproblem with relative error 2^{-b} in additional $O(n \log^2 n \log b)$ time. Furthermore, it can be used to dynamically maintain the singular value decomposition (SVD) of a generic matrix. Together with the algorithm the hardness of the problem is studied. For the symmetric case we present an $\Omega(n^2)$ lower bound for rank-one updates and an $\Omega(n)$ lower bound for element updates.

[RS-08-1] Anders Møller.
Static Analysis for Event-Based XML Processing.
jan 2008.
23 pp. Appears in PLAN-X '08.

Abstract: Event-based processing of XML data – as exemplified by the popular SAX framework – is a powerful alternative to using W3C's DOM or similar tree-based APIs. The event-based approach is particularly superior when processing large XML documents in a streaming fashion with minimal memory consumption.

This paper discusses challenges for creating program analyses for SAX applications. In particular, we consider the problem of statically guaranteeing that a given SAX program always produces only well-formed and valid XML output. We propose an analysis technique based on existing analyses of Servlets, string operations, and XML graphs.