Modularity Seminar
Spring 2002

Four Themes:

- Enhancing the Technology of Intersection Types
- Module Calculi
- XML-Technology-Based Standards for Manipulating Computer Programs
- Type Systems for Modules without Dependent Types

Enhancing the Technology of Intersection Types

READING:

Primary:

* Kfoury, Wells. In POPL 1999. Introduces expansion variables and System I.
* Wells, Haack. In ESOP 2002. Introduces branching types and lambda-B.
* Amtoft, Turbak. In ESOP 2000.
* Wells, Dimock, Muller, Turbak. In JFP 2002. Introduces lambda-CIL.

Secondary:

* Van Bakel. Ph.D. thesis.
* Barbanera, Dezani, de'Liguoro. In MSCS.

RESEARCH:

* Extend expansion variable technology to work with intersection types that have properties similar to associativity, commutativity, and idempotence of the intersection type constructor. The resulting system should have subject reduction, unlike System I.
* Develop a system integrating the features of expansion variables and branching types.
* Extend the above-mentioned technology to work with real-world programming language features like records, variants, and mutually recursive definitions.

Module Calculi

READING:

* Cardelli, POPL 1997
* Shao, Typed Cross-Module Compilation, ICFP 1998
* Machkasova, Turbak. ESOP 2000.
* Wells, Vestergaard. ESOP 2000.
* Fisher, Reppy, Riecke, A Calculus for Compiling and Linking Classes, ESOP 2000
* Hirschowitz, Leroy. ESOP 2002
* Ancona, Zucca. A Calculus of Module Systems, JFP.

RESEARCH:

* Dealing with assignments and other imperative operations. Partial progress by Hirschowitz/Leroy and Hirschowitz/Wells.
* Type systems for modules. In particular, mediating between intersection/union types and universal/existential types.
* Link-time optimization and cross-module transformations: extending calculi to formally describe more transformations, justifying correctness.
* Abstracting over the base calculus.
* Issues of call-by-value/name/need in the context of module calculi.
* Exploring relationships between modules and OO programming.

XML-Technology-Based Standards for Manipulating Computer Programs

READING:

* Forest/Hedge Automata Theory (very relevant for XML type checking)
* Functional programming + recursive data types + XML:
  ** Static types for dynamic documents (M.Shields' thesis)
  ** Scheme and XML
  ** XMLambda
  ** Haskell and XML
  ** \dl-calculus and DL (Santiago Pericas' thesis)
  Pointers to the preceding can be found at http://types.bu.edu/xml
* RelaxNG: next generation schema language for XML
* Stuff on XML Schema, XDuce, RELAX NG. http://www.xml.com/lpt/a/2001/12/12/schemacompare.html
* Burckhardt. M.A. Thesis 2000.
* The Cornell Synthesizer Generator and Its Successors (tree-based syntax editors). Web link?
* Stuff on incremental analysis/compilation during editing. Web link?

RESEARCH:

* Standardizing XML vocabularies/dialects for representing programs in particular languages.
* Developing cross-programming-language standards for representing common features.
* Developing technology for principal-typing-based incremental analysis/compilation during editing. (Technically, this has nothing to do with XML specifically, but only with trees. However, doing this to full advantage depends on standards for representing/communicating trees.)
* XML Transformations: transformation languages, static typing, regular automata vs. hedge/tree automata.