AFRL-IF-RS-TR-2001-280
Final Technical Report
January 2002

SOFTWARE EVOLUTION USING HIGHER ORDER TYPED (HOT) LANGUAGE TECHNOLOGY

Yale University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. D888

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2001-280 has been reviewed and is approved for publication.

APPROVED:

NANCY A. ROBERTS

Project Engineer

FOR THE DIRECTOR:

MICHAEL TALBERT, Maj., USAF, Technical Advisor
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JANUARY 2002	Final Aug 96 - Jul 99

4. TITLE AND SUBTITLE
SOFTWARE EVOLUTION USING HIGHER ORDER TYPED (HOT) LANGUAGE TECHNOLOGY

5. FUNDING NUMBERS
C - F30602-96-2-0232
PE - 62702F
PR - D883
TA - 01
WU - 01

6. AUTHOR(S)
Paul Hudak, Zhong Shao, and John Peterson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Yale University
Department of Computer Science
51 Prospect Street
New Haven Connecticut 06520-8285

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFTD
525 Brooks Road
Rome NY 13441-4514

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2001-280

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: Nancy A. Roberts/IFTD/(315) 330-3566

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
The objective of this effort was to develop and demonstrate innovative approaches and techniques to support rapid incorporation of new requirements and technologies into and evolving system's capabilities and architecture. Their approach was to develop and demonstrate the integrated incremental programming environment by developing designs based on modular executable specs, capturing architectures by using a higher order typed (HOT) module language, and developing a programming environment which supports incremental programming. Yale developed and demonstrated HOT language technology in three areas: domain specific languages, compilers and architecture definition languages. They also created HOT ACME, an architectural description language (ADL). They collaborated with other groups on two widely used HOT languages: Haskell, as implemented by the Hugs interpreter, and ML, as implemented by SML/NJ using FLINT backend.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Higher Order and Typed Languages, Incremental Programming, Architecture Definition Languages	11
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
		UNCLASSIFIED	

Table of Contents

List of Figures

1.1 Objective

HOT languages are characterized as being "Higher-Order" and "Typed," two characteristics that greatly facilitate the development of evolvable, modular software. This project bases its work on the use of HOT language technology, with particular focus on the application and implementation aspects of the technology. In addition, the project focuses on the use of domain-specific embedded languages and rincipled interoperability. Most of the work is being done in the context of the HOT languages ML and Haskell.

1.2 Approach

This work is divided into three inter-related areas: software architecture, HOT infrastructure, and domain-specific embedded languages (DSEL).

In the area of software architecture, the project focuses on extending the ACME architectural description language with powerful features from the HOT language community. Two extensions are currently being investigated, one involves extending ACME with higher-order functions to support parameterized architectures in a more general way than templates; another involves extending ACME with user-defined constraints. Higher-order functions are useful for expressing complex architectures in a concise and simple manner; constraints allow many forms of program analysis to be captured through the use of domain-specific constraint solvers.

In the area of HOT infrastructure, the project focuses on two areas: First is the implementation of the FLINT system. FLINT is a common typed intermediate language that models the semantics and interactions of various HOT language features. It provides a common compiler backend that can be quickly adapted to generate compilers for new HOT languages. With its single unified type system, FLINT serves as a great platform for reasoning about cross-language interoperations. It can also serve as more powerful executable content than Java VM code, making all HOT programs Internet ready.

Another aspect of our HOT infrastructure is the implementation of HUGS, considered to be the premier programming environment for Haskell users. At the core of HUGS is a byte-code interpreter whose fast compilation times make it very popular. We are collaborating with the Universities of Glasgow and Nottingham to combine this flexible core with a GHC, considered the best compiler for Haskell. With the resulting merger, one will be able to mix interpreted and compiled code, just as in a traditional Lisp environment.

Finally, abstraction is arguably the most important factor in writing evolvable, modular software. But the ultimate abstraction for a particular application is a domain specific language: a programming language specifically tailored to the application, with which a person can quickly and effectively develop a complete software system. Unfortunately, designing and implementing

a new language is difficult, and furthermore, we can expect the language itself to evolve. Thus this project begins with the idea of inheriting the infrastructure of some other (HOT) languages---tailoring it in special ways to the domain of interest---thus yielding a domain specific embedded language (DSEL). The language FRAN (functional reactive animation) is one such example---designed for creating richly interactive, multimedia animations. FRAN is embedded in Haskell, but has the distinct look and feel of a completely new language.

1.3 Recent FY-1999 Accomplishments

Extended FLINT to support Java type system. Designed a conservative extension of the polymorphic lambda calculus (Fomega) as an intermediate language for compiling languages with name-based class and interface hierarchies. The extension enables the safe interoperation between class-based and higher-order languages and the reuse of common type-directed optimization techniques, compiler back ends, and runtime support. A paper entitled "Representing Java classes in a typed intermediate language" is to appear in 1999 ACM International Conference on Functional Programming (ICFP99).

Designed and implemented a cross-module inlining and specialization algorithm in the FLINT compiler. The new algorithm can guarantee inlining of functions across arbitrarily functorized code while still reserving separate compilation.

Developed a reference implementation for Functional Reactive Programming. This implementation has simple, easily understood semantics and may serve as a formal definition of one possible semantics for Functional Reactive Programming.

Completed the merger between Hugs and GHC. Developed a new runtime system to support interoperation between compiled GHC code and interpreted Hugs code. Released a new version of Hugs, compliant with the new Haskell 98 standard.

1.4 Technology Transition

The FLINT infrastructure has been recently incorporated into the SML/NJ program environment --- a production system jointly developed by Lucent, Princeton, Yale, and AT&T. Both FLINT and SML/NJ are now being used worldwide by large number of software developers (including the DARPA-supported projects such as the Fox project at CMU and the Zephyr project at U. Virginia and Princeton).

FRAN is being developed in collaboration with researchers at Microsoft, whose interest relates not just to easing graphics and animation programming, but also as a viable method for programming Talisman, their new sprite-based 3D graphics hardware.

Hugs is being used world-wide by hundreds of users. Most of these users are academic researchers, many are using it for teaching, but only some are using it in industrial settings. Hopefully, this latter category will increase, especially with the improvement in web functionality.

1.5 A Chronological List of Written Publications

Haskore Music Tutorial, by Paul Hudak, in Advanced Functional Programming, Springer Verlag LNCS 1129, pp. 38-68, August 1996.

Rolling Your Own Mutable ADT---A Connection between Linear Types and Monads, by Chih-Ping Chen and Paul Hudak, in Proceedings of 24th ACM Symposium on Principles of Programming Languages, January 1997.

Type Inference with Constrained Types, by Martin Sulzmann, Martin Odersky and Martin Wehr, in Proceedings of the FOOL'97 (Foundations of Object-oriented Languages) Workshop, January 1997.

Report on the Programming Language Haskell (version 1.4), by John Peterson (editor). Research Report YALEU/DCS/RR-1106, Yale University, April 1997.

Standard Libraries for the Programming Language Haskell (version 1.4), by John Peterson, Research Report YALEU/DCS/RR-1105, Yale University, April 1997.

The RBMH User Guide, by John Peterson and Gary Shu Ling. April 1997.

Hugs 1.4 User Manual, by Mark P. Jones and John C. Peterson, Yale University, Department of Computer Science, Research Report YALEU/DCS/RR-1123, 1997.

Principled Dynamic Code Improvement, by John Peterson, Paul Hudak, and Gary Shu Ling, Yale University, Department of Computer Science, Research Report YALEU/DCS/RR-1135, July 1997.

Functional Reactive Animation, by Paul Hudak and Conal Elliott, in Proceedings of 1997 International Conference on Functional Programming, June 1997.

Flexible Representation Analysis, by Zhong Shao, in Proceedings of 1997 International Conference on Functional Programming, June 1997.

An Overview of the FLINT/ML Compiler, by Zhong Shao, in Proceedings of 1997 ACM SIGPLAN Workshop on Types in Compilation, June 1997.

Green Card: A Foreign Language Interface for Haskell, by Alastair Reid, Simon Peyton Jones, and T. Nordin, in Proceedings of the 1997 Haskell Workshop, Amsterdam, the Netherland, June 1997.

Typed Common Intermediate Format, by Zhong Shao, in Proceedings of 1997 Usenix Conference on Domain Specific Languages, October 1997

Modular Domain Specific Languages and Tools, by Paul Hudak, submitted to 1998 International Conference on Software Reuse, October 1997.

An Intermediate Meta-Language for Program Transformation, by Mark Tullsen and Paul Hudak, submitted to 1998 ACM Symposium on Programming Language Design and Implementation, November 1997.

Implementing Typed Intermediate Languages by Zhong Shao, Christopher League, Stefan Monnier, Bratin Saha, and Valery Trifonov, submitted to 1998 ACM Symposium on Programming Language Design and Implementation, November 1997.

Type Inference with Constrainted Types, by Martin Odersky, Martin Sulzmann and Martin Wehr, invited submission to Journal of TAPOS (Theory and Practice of Object Systems), October 1997.

First-Class Schedules, by Rajiv Mirani and Paul Hudak, being submitted to ACM Transactions on Programming Languages and Systems, November 1997.

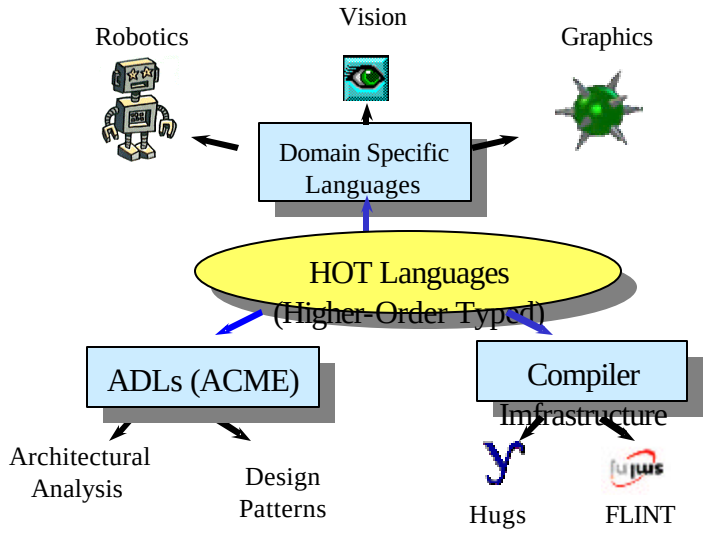
Efficient and Safe-for-space Closure Conversion, by Zhong Shao and Andrew Appel, being submitted to ACM Transactions on Programming Languages and Systems, December 1997.

Flexible Representation Analysis, by Zhong Shao, being submitted to ACM Transactions on Programming Languages and Systems, December 1997.

Typed Cross-Module Compilation, by Zhong Shao, being submitted to Journal of Functional Programming, December 1997.

1.6 Additional Information

Figure 1 is a quad chart describing the technology. Additional project information can be found at the following website: http://www.cs.yale.edu/edcs



New Ideas

- Domain-Specific Embedded Languages capture the semantics of an application domain.
- Use functional programming methodology to create the next generation of Architecture Description Languages.
- Use types to ensure reliability and correctness, express complex program properties, implement program analysis tools.
- Interoperability using advanced software components. Use types and higher-order functions to compose components reliably.

- Construction of domain-specific languages for robotics, vision, graphics allows rapid development and evolution of programs in these domains.
- Extensions to ACME (an Architectural Description Language) increase the power of ACME. Extensions include higher-order functions and analysis tools.
- Functional programming language implementations at Yale (Hugs and SML/NJ) are used by researchers worldwide.

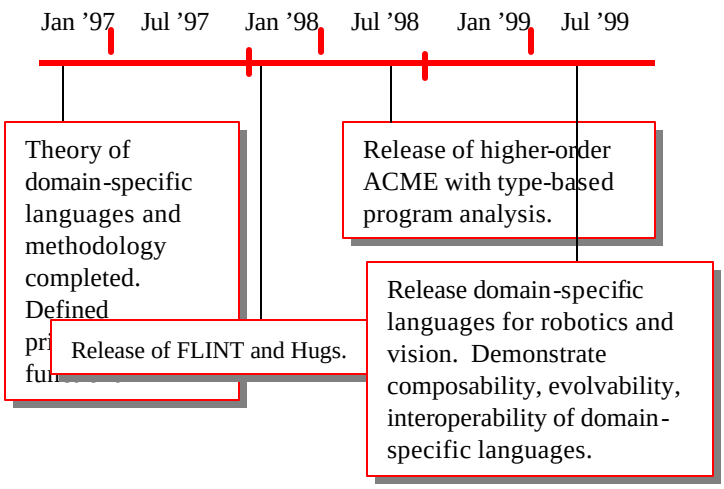


Figure 1