Computational Reflection and Meta-level Architectures

October 9, 1998

Computational reflection, proposed by Brian Smith, is a computational process that is able to reason about itself. This framework can be applied to extensible software systems (known as *Open Implementations*) that can be easily extended by modifying/replacing their implementations. We believe that reflection is beneficial to programming languages, especially for parallel and distributed systems, and design several reflective programming languages. Also, we study theoretical aspects of reflection as a formal background of the Open Implementations. This document introduces the research activities on computational reflection and meta-level architectures at the laboratory.

- Research Activities in the Lab.
- Publications from the Lab.
- Available Systems
- Members Working on Reflection
- Related Research Groups

Note: Postscript version of this document is also available.

1 Research Activities in the Lab.

Our interests on computational reflection are its theoretical aspects, reflective programming systems, and use of reflection for practical use.

1.1 Theoretical Aspects of Reflection

As a formal foundation of *Open Implementation,* theoretical aspects of reflection, such as semantics, are studied.

- Group-wide reflection and ACT/R [WY90]

  Group-wide reflection is a dimension of computational reflection in concurrent systems. A general model for group-wide reflection is described based on Actor model, and several properties are proved.

- Rscheme [IMY92]

  Rscheme is a reflective Scheme based on the *kernel-less* design principle of RbCl; all its run-time modules (including meta-interpreter components) can be replaced via the *system object table.*

- Black

  Inspired by Rscheme, general framework of reflective languages is studied. A reflective language *Black*, which admits us to replace meta-interpreter functions, is designed. Based on Black, we show an implementation framework of reflective languages, via *duplication of meta-interpreters and partial evaluation*[AMY96]. Further partial evaluation, which leads to efficient implementation techniques, are also studied[AMMY95]. An implementation in Scheme is available. An online partial evaluation designed for use in reflection is discussed in [AMY97].

1.2 Languages Systems for Object-Oriented/Concurrent/Logic Programming

In concurrent/distributed programming systems, like our ABCL family of languages, extensibility in both language constructs and run-time systems are beneficial for efficient execution. To provide such extensible language systems, reflection in concurrent/distributed systems are studied.

- ABCL/R[WY88]

  The notion of reflection proposed by Brian Smith and Pattie Maes, is introduced into object-oriented *concurrent* computation(link to ABCL project). The reflective architecture, later called as *Individual Based Reflective Architecture*, is designed and its description language ABCL/R is implemented.

- ABCL/R2

  To formulate reflective features provided in practical reflective systems (e.g., resource management), a *hybrid group reflective architecture* is proposed. This architecture has notion of group shared resources, which, for example, corresponds to processor elements of parallel computers[MWY91]. Its description language *ABCL/R2* is designed, and efficient implementation technique is studied[MMWY92].

A pseudo-parallel version of ABCL/R2 system is available via anonymous FTP, and used for several research projects as an extensible concurrent language platform.

- RbCl

  RbCl is a reflective object-oriented concurrent language[IMY92]. It is designed so that *any* run-time routine of the language can be replaced with RbCl objects—what we call *no kernel* in the language. The RbCl system has *system object tables*, which registers run-time routines, which is written in C++ at the beginning. The table can be modified, and user defined RbCl objects can be executed as run-time routines via a novel feature *linguistic symbiosis.*

- CodA[McA93, McA95a]

  CodA is a framework for creating, controlling and understanding concurrent computing environments. Objects in CodA conceptually has several meta-objects, which are individually replaceable. Its implementation on Smalltalk seamlessly ties CodA objects and Smalltalk objects, so that existing Smalltalk libraries can be easily extended for different environments, such as distributed computing. (For more information.)

- ABCL/R3

  Computational reflection is a potential way to optimize high-performance applications especially for parallel computers. To realize this, we re-design a reflective concurrent object-oriented language *ABCL/R3* so that various features of parallel computers can be managed via reflection[MMY93, MMY94, MMY96].

  Also, compilation frameworks for efficient execution are investigated[MMAY95, MY97, MY98b] based on partial evaluation[AMY97]. Given meta-level and base-level programs, our compiler specializes the meta-program with respect to the base-program, and yields a single-level program in which interpretive execution is substantially eliminated.

- RKL1

  RKL1 is a reflective version of KL1, which is a parallel logic programming language developed at ICOT. The notable feature of RKL1 is simplicity of its reflective architecture; programmers can modify only around the invocation of the body part of each clause. This simplicity helps us to easily develop a compiler that can generate programs that is embedded meta-level code, and runs efficiently. In fact, some example programs with meta-level descriptions exhibit good performance on multiple workstations connected via Ethernet. [TT94]

- MPC++

  MPC++ is a metalevel-architecture-based C++ compiler system. Researchers in RWCP(Real World Computing Partnership), in cooperation with our group, are developing this system with the aim of providing practical parallel C++ systems using its meta-level architecture.[TISY96, TISY97c, TISY97b, TISY97a, TISY98]

1.3 Applications

Inheritance Anomaly In concurrent object systems, inheritance and synchronization constraints often conflict each other—known as *inheritance anomaly.* One of our solution to this problem is the use of meta-level architecture; we proposed an object-oriented concurrent language X0/R with a reflective architecture and shown a solution on it. [MY90, MTY93, MY93, Mat93]

Tj Tj is a distributed computing framework implemented using the meta-architecture of CodA. Thanks to CodA, policies of language features related to distributed environment (e.g., object migration) is opened for programmers. [McA95b] (For more information.)

Synchronizer In object-oriented concurrent computing, the way to describe coordination of multiple concurrent objects is one major problem. Synchronization constraints (*Synchronizers*, in short) proposed by Frølund and Agha helps to describe such coordination independent from the description of each concurrent object. We designed an extended Synchronizers for ABCL, and implemented using reflective architecture of ABCL/R2. [Iga94]

References

[AMMY95] Kenichi Asai, Hidehiko Masuhara, Satoshi Matsuoka, and Akinori Yonezawa. Partial evaluation as a compiler for reflective languages. Technical report of the Department of Information Science, the University of Tokyo, 95–10, 1995. Postscript file is at http://www.is.s.u-tokyo.ac.jp/tech-reports/TR95-10-a4.ps.gz.

[AMY96] Kenichi Asai, Satoshi Matsuoka, and Akinori Yonezawa. Duplication and Partial Evaluation — For a Better Understanding of Reflective Languages —. *Lisp and Symbolic Computation,* Vol. 9, Nos. 2/3, pp. 203–241, Kluwer Academic Publishers, 1996. Postscript files are at http://www.is.s.u-tokyo.ac.jp/%7Easai/papers/lasc96-{a4,letter}.ps.gz.

[AMY97] Kenichi Asai, Hidehiko Masuhara, and Akinori Yonezawa. Partial Evaluation of Call-by-value lambda-calculus with Side-effects. *ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation (PEPM '97)*, pp. 12–21, 1997. Postscript file is at http://www.is.s.u-tokyo.ac.jp/%7Easai/papers/pepm97.ps.gz.

[IMY92] Yuuji Ichisugi, Satoshi Matsuoka, and Akinori Yonezawa. RbCl: A reflective object-oriented concurrent language without a run-time kernel. In *Proceedings of International Workshop on New Models for Software Architecture (IMSA): Reflection and Meta-Level Architecture*, pages 24–35, Tama City, Tokyo, November 1992. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.Z.

[Iga94] Atsushi Igarashi. Study on Mechanisms for Multi-Object Synchronization and their Implementation. *Senior's thesis,* Department of Information Science, University of Tokyo, 1994.

[McA93] Jeff McAffer. The CodA MOP. In *Proceedings of OOPSLA'93 Workshop on Object-Oriented Reflection and Metalevel Architectures*, Washington, D.C., September 1993. Postscript file is at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/meta/oopsla93.workshop/postscript/mcaffer-letter.ps.Z.

[McA95a] Jeff McAffer. Meta-level programming with CodA. In *Proceedings of European Conference on Object-Oriented Programming (ECOOP)*, 1995. to appear. Postscript file is at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/members/jeff/docs/ecoop95.a4.ps.gz.

[McA95b] Jeff McAffer. Meta-level Architecture Support for Distributed Objects.
Submitted to IWOOS'95. Postscript file is at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/members/jeff/docs/iwooos95.a4.ps.gz.

[MMWY92] Hidehiko Masuhara, Satoshi Matsuoka, Takuo Watanabe, and Akinori Yonezawa. Object-oriented concurrent reflective languages can be implemented efficiently. In *Proceedings of Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 127–145, Vancouver, B.C., October 1992. ACM. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.Z.

[MMY93] Hidehiko Masuhara, Satoshi Matsuoka, and Akinori Yonezawa. Designing an OO reflective language for massively-parallel pro-

cessors. In *Proceedings of OOPSLA'93 Workshop on Object-Oriented Reflection and Metalevel Architectures*, Washington, D.C., October 1993. Postscript file is at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/meta/oopsla93.workshop/postscript/masuhara.ps.Z.

[MMY94] Hidehiko Masuhara, Satoshi Matsuoka, and Akinori Yonezawa. An Object-Oriented Concurrent Reflective Language for Dynamic Resource Management in Highly Parallel Computing. In *IPSJ SIG Notes*, 94-PRG-18 (SWoPP'94), pp. 57-64, Jul., 1994. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.Z.

[MMAY95] Hidehiko Masuhara, Satoshi Matsuoka, Kenichi Asai, and Akinori Yonezawa. Compiling Away the Meta-Level in Object-Oriented Concurrent Reflective Languages Using Partial Evaluation. In *Proceedings of Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'95)*, ACM SIGPLAN Notices Vol.30 No.10, pages 300–315, Austin, 1995. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.gz.

[MMY96] Hidehiko Masuhara, Satoshi Matsuoka, and Akinori Yonezawa. Implementing Parallel Language Constructs Using a Reflective Object-Oriented Language. In *Reflection Symposium'96*, pp.79–91, April, 1996. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.gz.

[MY97] Hidehiko Masuhara and Akinori Yonezawa. Reasoning-conscious Meta-object Design of a Reflective Concurrent Language. In *Proceedings of International Workshop on New Models for Software Architecture (IMSA'97)*, pp.42–56, October, 1997. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.gz.

[MY98a] Hidehiko Masuhara and Akinori YONEZAWA. A Reflective Approach to Support Software Evolution. In *Proceedings of International Workshop on the Principles of Software Evolution*, pp.135–139, April 1998.

[MY98b] Hidehiko Masuhara and Akinori Yonezawa Design and Partial Evaluation of Meta-objects for a Concurrent Reflective Language. In *Proceedings of European Conference on Object-Oriented Programming (ECOOP'98)*, Lecture Notes in Computer Science Vol. 1445, pp.418–439, July, 1998. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.gz.

[MY90] Satoshi Matsuoka and Akinori Yonezawa. Metalevel solution to inheritance anomaly in concurrent object-oriented languages. In

ECOOP/OOPSLA'90 Workshop on Reflection and Metalevel Architectures in Object-Oriented Programming, 1990.

[MWY91] Satoshi Matsuoka, Takuo Watanabe, and Akinori Yonezawa. Hybrid group reflective architecture for object-oriented concurrent reflective programming. In *Proceedings of European Conference on Object-Oriented Programming (ECOOP)*, 1991. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.Z.

[MTY93] Satoshi Matsuoka, Kenjiro Taura, and Akinori Yonezawa. Highly Efficient and Encapsulated Re-use of Synchronization Code in Concurrent Object-Oriented Languages. In *Proceedings of Proceedings of Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'93)*, pp.109–126, 1993. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.Z.

[MY93] Satoshi Matsuoka and Akinori Yonezawa. Analysis of inheritance anomaly in object-oriented concurrent programming language. In Gul Agha, Peter Wegner, and Akinori Yonezawa, editors, *Research Directions in Concurrent Object-Oriented Programming,* pages 107-150. MIT Press, 1993. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.Z.

[Mat93] Language Features for Extensibility and Re-use in Concurrent Object-Oriented Languages. PhD thesis, Department of Information Science, the University of Tokyo, 1993. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.Z.

[TT94] Toshiyuki Takahashi and Masayuki Takeda. An Efficient Implementation of Reflection in KL1. In *FGCS'94 Workshop on Parallel Logic Programming*, pp.17–26, 1994.

[TISY96] Toshiyuki Takahashi, Yutaka Ishikawa, Mitsuhisa Sato, and Akinori Yonezawa. Parallel programming using meta-level architecture. *IPSJ SIG Notes*, Vol. 96, No. 82, pp. 79–84, August 1996. (In Japanese).

[TISY97a] Toshiyuki Takahashi, Yutaka Ishikawa, Mitsuhisa Sato, and Akinori Yonezawa. A compile-time meta-level architecture supporting class specific optimization. In Yutaka Ishikawa, Rodney R. Oldehoeft, John V.W.Reynders, and Marydell Tholburn, editors, *Scientific Computing in Object-Oriented Parallel Environments (1st International Conference, ISCOPE97)*, Vol. 1343 of *Lecture Notes in Computer Science*, pp. 89–96, December 1997.

[TISY97b] Toshiyuki Takahashi, Yutaka Ishikawa, Mitsuhisa Sato, and Akinori Yonezawa. High performance object-oriented computing supported by meta-level architecture. *IPSJ SIG Notes*, Vol. 97, No. 78, pp. 37–42, August 1997. (In Japanese).

[TISY97c] Toshiyuki Takahashi, Yutaka Ishikawa, Mitsuhisa Sato, and Akinori Yonezawa. Optimization technique for class libraries using meta-level architecture. In *Joint Symposium on Parallel Processing (JSPP)*, May 1997. (In Japanese).

[TISY98] Toshiyuki Takahashi, Yutaka Ishikawa, Mitsuhisa Sato, and Akinori Yonezawa. Optimization technique for class libraries using meta-level architecture. *Transactions of Information Processing Society of Japan*, Vol. 39, No. 6, pp. 1681–1690, June 1998. (In Japanese).

[WY88] Takuo Watanabe and Akinori Yonezawa. Reflection in an object-oriented concurrent language. In *Proceedings of Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 306–315, San Diego, CA, September 1988. ACM. (revised version in Chapter 3 of [Yon90]).

[WY90] Takuo Watanabe and Akinori Yonezawa. An actor-based metalevel architecture for group-wide reflection. In *Proceedings of REX School/Workshop on Foundation of Object-Oriented Languages*, 1990. Postscript files are at ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/papers-{a4, letter}.ps.Z.

[Yon90] Akinori Yonezawa, editor. *ABCL: An Object-Oriented Concurrent System*. MIT Press, Cambridge, MA, 1990.

If you can not obtain electrically, please make a request for hardcopy version to: paper-request@yl.is.s.u-tokyo.ac.jp.

2 Available Systems

Some systems are available via anonymous FTP at ftp.yl.is.s.u-tokyo.ac.jp. For the other systems, please make a request to each responsible person.

- ABCL/R: in directory `/pub/abclr/`.
- ABCL/R2: in directory `/pub/abclr2/`.
- Black: in directory `/pub/black/`.

3 Members Working on Reflection

- Professor Akinori Yonezawa
- Takuo Watanabe — currently with JAIST, leading several projects including LEAD/LEAD++.
- Satoshi Matsuoka — currently with Tokyo Institute of Technology, leading the OpenJIT project.
- Yuuji Ichisugi — currently with ETL, working on extensible compilers including EPP.
- Jeff McAffer — currently with Object Technology International.
- Kenichi Asai — currently with Imai lab.
- Hidehiko Masuhara — currently with Graduate School of Arts and Sciences.
- Toshiyuki Takahashi — currently with Real World Computing Partnership, working on MPC++.
- Atsushi Igarashi — currently with Kobayashi lab.

4 Related Pages

Research Groups

- Open Implementation Home Page — general information about Open Implementation, and their projects (CLOS MOP, Open C++, etc.).
- Apertos project at Sony CSL — a reflective, object-oriented operating system.
- OOPS at the Prof. Mario Tokoro group, Keio University (in Japanese) — working on reflective langugaes (AL-1/D, etc.) and operating systems.
- Massively Parallel Software Laboratory, Real World Computing Partnership — working on MPC++ (an open compiler) and OCore (a concurrent object-oriented language with meta-level architecture).
- OpenC++ Home Page — an open compiler based on compile-time reflection.
- Object-Oriented Systems and Languages Group, Ecole des Mines de Nantes — ClassTalk, NeoClassTalk, OpenCORBA

- DeVa Project, York University
- Dalang Java — Java with run-time reflection.
- Coyote Project and Iguana Programming Model — middleware support with meta-object protocols

Conferences and Workshops

- Reflection'96 online proceedings.
- ECOOP'97 Workshop on Reflective Real-Time Object-Oriented Programming and Systems
- ECOOP'98 Workshop on Reflective Object-Oriented Programming
- OOPSLA'98 Workshop on Reflective Programming in C++ and Java