# 1.2. ロバストソフトウェア構成法

武市正人　胡 振江　筧 一彦
情報理工学系研究科　数理情報学専攻

**概要**

本プロジェクトは、モデル検査等の形式的手法に基づいてロバストなソフトウェアを構成する方法を確立し、融合プロジェクトを通じてその手法の有効性を実証しようとするものである。

## 1 はじめに

問題解決のためのプログラムの信頼性を高め、ロバストなソフトウェアを構築するための形式的な方法論を確立するためには、それが基礎としているモデルの検証が必要である。プログラムが仕様に適合するものであるかどうかを検証する技術は 1970 年代から研究されているものの、実用規模のプログラムの正当性を検証することはその複雑さのために不可能であるという現実がある。本プロジェクトでは、ソフトウェアの注目すべき特定の性質を抽出したモデルに対して形式的手法を適用し、モデル検査を通じてロバストなソフトウェアの構築法を確立しようとするものである。

(1) モデル検査システムに関する研究：形式的手法の一つであるモデル検査の手法は、基本的には、有限状態オートマトンにおける状態遷移を検査することによってシステムの検証を行なうもので、各種の効率的な検査手法が開発されている。これらのなかには、「力まかせ」による部分もあり、近年の計算能力の向上によって実用化されたものもある。既存のモデル検査システムを調査するとともに、従来、多く使われてきたハードウェア検査だけではなく、ソフトウェア(プログラム)のロバスト性検査に適合させるための手法を追究する。

(2) モデル検査のためのモデル構築に関する研究：プログラムの検証の困難さは、動的な振舞いの複雑さにある。モデル検査手法によってそのロバスト性を高める本研究の手法は、その振舞いを抽象化し、ある側面から見た性質だけを検査するという抽象解釈の手法により状態数を抑えようというものである。具体的なプログラムに対して、注目する性質が与えられたときに、その抽象化を行なうことはそれほど簡単なものではないが、いくつかの代表的な性質に対する抽象的な領域の設定を通してモデル構築の手法を提示する。

(3) プロジェクト融合によるプログラム検査の実施：他のプロジェクトで対象としているアルゴリズムやプログラムに対して、モデル検査手法を適用し、プログラムの信頼性を高めることを試みる。

以上の研究実施にあたっては、この分野の国内外の研究者との交流によって、最初の 2 年間で形式的手法を確立するとともに、実用的な検査システムの応用技術を蓄積し、そのあとで現実の問題を扱うこととする。

## 2. 平成 14 年度の研究成果

本プロジェクトに関連する成果として、モデル検査技術を用いたプログラム解析法に関する論文を発表するとともに、ロバストソフトウェアの

構築法に関する国際ワークショップで研究課題の方向付けを確認した。

2.1 モデル検査によるプログラム解析の自動化

情報処理学会第 42 回プログラミング研究会(2003 年 1 月 22 日～23 日)で以下の論文を発表した。

山岡祐司、胡振江、筧一彦、武市正人. モデル検査技術を利用したプログラム解析の自動生成

概要：プログラムの伝統的な最適化において利用されるプログラム解析の多くがプログラムのデータフロー上での性質を表現した時相論理によって記述することができる。本研究は、時相論理式で記述されたプログラムの性質の解析処理を自動化するツールの開発とその実用性を論じたものである。このツールは、現在、一般に使われている実用言語 Java と相互に変換可能な中間言語 Jimple のプログラムが時相論理 CTL-FV による性質記述を満たしているかどうかをモデル検査技術により検証する。そこでは、解析対象の Jimple プログラムと解析したい性質記述の CTL-FV 論理式を入力とし、モデル検査ツール SMV 向けのモデルを生成する。それにより、SMV の実行を通じてそのプログラムが特定の性質を満たすかどうかを解析することができる。本論文では CTL-FV からモデルを構築する方法を中心に述べている。

2.2 Workshop on Robust Software Construction

ワークショップ WRSC2003 を 2003 年 2 月 28 日～3 月 2 日に神奈川県葉山の IPC 生産性国際交流センターで開催した。企画から開催までの時間が十分に得られなかったが、4 名の研究者を海外から招聘するとともに、わが国の関連研究者の参加を募った。

Information Processing Laboratory in the University of Tokyo starts a project named *Robust Software Construction*. As its name indicates, this project aims to formalize a framework which assists construction of *robust* software that is efficient, correct, safe, and secure.

As the beginning of this five-year project, we would like to hold a workshop and welcome researchers of various fields in order to deepen knowledge and insight each other regarding robustness of software construction. The research topics of this workshop are (but not limited to):

- program transformation and partial evaluation,
- model checking and program analysis,
- program optimization,
- algorithm derivation and synthesis,
- formal method on software engineering.

In this workshop there are four invited talks from abroad.

Hong Mei (Peking University)
Alberto Pettorossi
(University of Roma Tor Vergata)
Maurizio Proietti (IASI-CNR)
Eric Van Wyk (University of Minnesota)

The workshop organizers are as follows.

Masato Takeichi (University of Tokyo)
Yoshihiko Futamura (Waseda University)
Zhenjiang Hu (University of Tokyo)
Kazuhiko Kakehi (University of Tokyo)

プログラム：

February 28(Friday)

14:00-15:00 Invited Talk (Chair:M.Takeichi)
*Derivation of Efficient Logic Programs by Specialization and Reduction of Nondeterminism,*
Alberto Pettorossi (Univ. of Roma Tor Vergata)

15:20-16:30 Session 1 (Chair: Mei Hong)}
*Parallelization with Tree Skeletons*
Kiminori Matsuzaki (Univ. of Tokyo)
*Cumulative Method: Recursion Removal from Mutual Recursive Programs with One Descent Function*
Yuusuke Ichikawa, Zenjiro Konishi, Yoshihiko Futamura (Waseda Univ.)

16:50-18:00 Session 2 (Chair: Zhenjiang Hu)
*Towards Self-Applicable GPC: Progress Report*
Masahiko Kawabe, Zenjiro Konishi, Yoshihiko Futamura (Waseda Univ.)
*The Translation Power of the Futamura Projections*
Robert Gluck
(PRESTO, JST & Waseda Univ.)

March 1(Saturday)

9:00-10:00 Invited Talk (Chair: M. Ogawa)
*Software Verification and Synthesis via Program Transformation*
Maurizio Proietti (IASI-CNR)

10:20-11:30 Session 3 (Chair: A. Pettorossi)

*Maximum List Marking in Parallel*
Zhenjiang Hu
(Univ. of Tokyo & PRESTO, JST)
*Systematic Generation of Efficient Pattern Matchers for Compressed Data*
Yoshihiko Futamura, Zenjiro Konishi, Kazuaki Maeho, Masahiko Kawabe (Waseda Univ.)

11:30- Introduction of other researches

13:20-14:20 Invited Talk (Chair: S. Nakajima)
*Language Extensions for Robust Computing*
Eric Van Wyk (Univ. of Minnesota)

14:40-16:25 Session 4 (Chair: Eric Van Wyk)
*Automatic Generation of Program Analyses on Model Checking*
Yuji Yamaoka (Univ. of Tokyo)
*Model-Checking of Component Integration Frameworks - A Case Study -*
Shin Nakajima
(Hosei Univ. & PRESTO, JST)
*Generic Solution for Maximum Marking Problems in Generic Haskell*
Isao Sasano (Univ. of Tokyo)

16:50-18:00 Session 5 (Chair: M. Proietti)
*Functional Meta-programming for Program Calculation*
Tetsuo Yokoyama (Univ. of Tokyo)
*Catamorphic Approach to Control Flow Analysis*
Mizuhito Ogawa
(PRESTO, JST & Univ. of Tokyo)

March 2(Sunday)

9:00-10:00 Invited Talk (Chair: Y. Futamura)
*ABC: Architecture Based Component Composition*
Hong Mei (Peking Univ. )

10:20-11:30 Session 6 (Chair: R. Gluck)
*Fusing Functions of Regular Expression Types and Patterns*
Kazuhiko Kakehi (Univ. of Tokyo)
*A Compositional Framework for Mining Longest Ranges*
Haiyan Zhao (Univ. of Tokyo)

11:30- Closing

Abstract of the talk *Derivation of Efficient Logic Programs by Specialization and Reduction of Nondeterminism* by Alberto Pettorossi (join work with F. Fioravanti, M. Proietti, and S. Renault):

Program specialization is a program transformation methodology that improves program efficiency by exploiting the information about the input data that are available at compile time. We show that current techniques for program specialization based on partial evaluation do not perform well on nondeterministic logic programs. We then consider a set of transformation rules that extend the ones used for partial evaluation, and we propose a strategy for guiding the application of these extended rules so to derive very efficient specialized programs. The efficiency improvements that sometimes are exponential, are due to the reduction of nondeterminism and to the fact that the computations which are performed by the initial programs in different branches of the computation trees, are performed by the specialized programs within single branches. In order to reduce nondeterminism we also make use of mode information for guiding the unfolding process. We also extend our technique to logic programs with constraints and we show that, by making use of a new transformation rule called clause splitting, we can generate efficient, specialized programs that are deterministic. To exemplify our technique, we show that one can automatically derive very efficient matching programs and parsing programs. The derivations we have performed could not have been done by previously known partial evaluation techniques.

Abstract of the talk *Software Verification and Synthesis via Program Transformation* by Maurizio Proietti (joint work with F. Fioravanti and A. Pettorossi):

We show how program transformation can be used for proving properties of programs and for synthesizing programs from logical specifications. We consider constraint logic programs with locally stratified negation and we propose a technique for showing that a closed first order formula phi holds in the perfect model M(P) of a program P, written as M(P) |= phi. For this purpose we consider a new version of the unfold/fold transformation rules and we show that this version preserves the perfect model semantics. Our proof method, called unfold/fold proof method, shows M(P)|= phi in two steps: (Step 1) the formula phi is encoded as a set of clauses F(f,phi), where f is a

new 0-ary predicate symbol such that M(P) |= phi iff M(P U F(f,phi)) |= f, and then (Step 2) the program P U F(f,phi) is transformed, by using the unfold/fold rules, into a new program Q such that M(P U F(f,phi)) |= f iff M(Q) |= f. If Q contains the clause f <- (i.e. a statement asserting that f is true), then M(P)|= phi holds. If Q does not contain any clause for f,then M(P)|= phi does not hold. We also present a strategy for applying our unfold/fold rules in a semi-automatic way. Due to well-known undecidability results, our strategy is incomplete, that is, for some initial program P and formula phi such that M(P)|= phi holds, our strategy is not able to derive the clause f <-. However, we identify some classes of programs and formulas for which our strategy is fully automatic and always terminates by deriving either f <- or the empty set of clauses for f. Thus, our strategy is a decision procedure for checking whether or not M(P)|= phi for any given program P and formula phi in those classes. As an example, we show that the weak monadic second order theory (WS1S) can be decided by using the unfold/fold proof method. Finally, we show how the unfold/fold proof method can be enhanced to perform program synthesis from first order formulas with free variables.

Abstract of the talk *Language Extensions for Robust Computing* by Eric Van Wyk:

Extensible languages and compilers allow programmers to import new language features into their programming environment. These features may be general purpose or domain specific; they may be defined by the programmer or by a domain expert. They define their own syntax, semantics and optimizations and are often implemented as transformations into equivalent constructs in the base source language.

Besides providing a means for raising the *level of abstraction*, language extensions can also specify and verify certain correctness properties. In this talk, we describe an extensible language framework for Java and example extensions that help programmers write more robust code. One extension uses regular expressions over method names to specify the valid order in which method calls to an object can be made; it also dynamically ensures that this ordering is followed.

Abstract of the talk *ABC: Architecture Based Component Composition* by Mei Hong:

How to compose prefabricated components is a key issue in component-based reuse. Research on Software Architecture (SA) and Component-based Software Development (CBSD) provides two hopeful solutions from different perspectives. SA provides a top-down approach to realize component-based reuse. However, it pays less attention to the refinement and implementation of the architectural descriptions, not providing the necessary capability to automate the transformation or composition to form finally an executable application. CBSD technology such as J2EE and CORBA provides a feasible bottom-up way to construct systems from standard components, forming an implementation basis for an integrated component-oriented development process. However, these technologies do not pay attention to the systematic methodology to guide the CBSD process, especially the component composition at higher abstract levels. We argue that it is a natural solution to combine these two approaches.

In this talk, an architecture-based component composition (ABC) approach is presented. In this solution, SA description is used as the blueprint and middleware technology as the runtime scaffold for component composition, using mapping rules and mini-tools to shorten the gap between design and implementation. Our approach presents an ADL, called ABC/ADL, supporting component composition. Besides the capability of architecting software systems, it provides support to the automated application generation based on SA model via mapping rules and customizable connectors. A tool to support ABC approach, called ABC-Tool, is presented, which provides a graphic user interface to design and reason software architecture, then transforms architectural model into implementation running on middleware platform by composing pre-fabricated components into the target system in an automated process.