新しいソフトウェアの実現 —科研「情報学」プロジェクト A01 柱を振り返って—

玉井哲雄[†] 稲葉雅幸[†] 外山芳人[††]

文部科学省科学研究費補助金特定領域研究「IT の深化の基盤を拓く情報学研究」(略称：「情報学」，研究代表：安西祐一郎）の 6 本の柱の一つ「A01 新しいソフトウェアの実現」(柱長：玉井哲雄）の概要と，その中の代表的な研究例 3 件について紹介する．

New Approach to Software Construction —In retrospect of Kaken project on informatics—

Tetsuo Tamai,[†] Masayuki Inaba[†] and Yoshihito Toyama[††]

1. 「新しいソフトウェアの実現」プロジェクトとは

1.1 プロジェクトの構想

特定領域研究「情報学」は 2001 年 10 月より 2006 年 3 月までの 4 年半の期間で実施された．その中の柱の一つである「A01 新しいソフトウェアの実現」の構想は，次のような問題認識からスタートした．

コンピュータのためのソフトウェアが作られるようになってから，50 年が経つ．ソフトウェアを系統的な手法で開発するソフトウェア工学という分野が誕生してから数えても，すでに 30 年が経過している．しかし，現在のソフトウェア開発は新しい局面を迎え，従来とは異なる課題に直面している．

明らかな状況として言えるのは，ソフトウェアが社会において果たす役割がますます大きくなる一方で，それがは人の目には直接見えないものとなっていることである．しかし，いったんソフトウェアの不具合が顕在化すると，それが社会に及ぼす影響は甚大である．

ソフトウェア開発の生産性と信頼性を高めることを目的としたソフトウェア技術は，主に大規模・複雑な情報システム，たとえば国防システムや金融システムなどを的としてきた．しかし，21 世紀のソフトウェア開発の課題は，メインフレーム・コンピュータ上で稼動する大規模システムをどう構築するかという問題意識では，とても捉えられないものとなった．そのために新しいソフトウェアの実現方法が探究されなければならない．対象となる新しいソフトウェアは，どのような性質をもつだろうか．

遍在性： あらゆるところに存在する．
インターネットに象徴されるように，ソフトウェアはネットワークを介してあらゆるところからアクセス可能となり，またその動作自体がネットワーク上に分散したシステムとして機能するようになった．携帯電話，携帯端末，移動体通信，移動エージェントといった「モーバイル」製品の普及が，この実感を強めている．さらに組込みソフトウェアは，自動車，家電製品，通信機器，オフィスや住宅などあらゆるところに入り込み，ソフトウェアの存在を意識させないままに，遍く存在するという状況を呈している．

発展性： めまぐるしく変化・発展する．
ソフトウェアはもともと変化させやすいという性質を持つが，近年その変化のスピードは加速的である．システムは継続的に進化させられるので，従来の開発と保守の境界は，あいまいなものとなってきた．さらに，とくに応答型のシステムは，運用を続けながらそれを停めることなく進化させることが必要なため，開発環境と運用環境の間も区別しがたいものとなりつつある．そのようなソフトウェアが地球規模のネットワーク上に生息し，自己増殖的に発展しているのである．

多様性： 多様な表現形態があり，多数のバリエーションをもつ．

† 東京大学
The University of Tokyo
†† 東北大学
Tohoku University

マルチメディアの登場によりソフトウェアの表現形態は多様化したが，さらにさまざまなレベルの多様性が現代のソフトウェアには求められている．まず利用者が多様であり，使い方もさまざまなため，多品種を提供し管理していくことが要求される．さらに，上に挙げた発展性とも関係して，多くの版を次々と提供することが求められ，同時に開発期間の大幅な短縮が要請される．

信頼性の要求： 正しく動き，信用できることが要求される．

ソフトウェアが広く社会に浸透したため，ソフトウェアの故障，誤動作などの不具合の影響は計り知れない．利用者の信用が得られるように正しく動き，突然の停止や暴走などを起こさず持続的に機能が提供されることを保証するのは，ソフトウェア開発者に課された社会的使命といえる．さらにソフトウェアを使ってプライバシーを侵害する，といった悪用の可能性を防ぎ，また悪意を持った者からの攻撃にも強いソフトウェアであることが，強く要請される．

本プロジェクトの構想時から 5 年経った現在でも，この基本認識自体は大きく変える必要はないだろう．むしろこれらのソフトウェアの特徴が，より一層顕著になってきたというのが現状であろう．

1.2 プロジェクトの展開と成果

A01 柱は 2 つの計画班と 20 前後の公募班で構成された．公募は毎年行われ，採択数は年によって変動した．ただし，最後の 2 年は継続で，2004-5 年度に公募で参加した班の数は 22 であった．どのようなテーマの研究が応募され採択されるかは，直接コントロールできない．しかし，公募説明として述べた企画の背景と全体の研究のねらいが理解され，多様ではあるが企画側の趣旨に沿った優れた研究計画が集まった．そこで集まった公募研究に計画班を合わせて，それらのテーマをグループに分類すると，次のようにまとめることができる．

- 「コンポーネント技術」グループ
- 「並行・分散計算資源モデル」グループ
- 「プログラム理論・言語・検証手法」グループ
- 「開発方法論と開発支援環境」グループ

これらに共通する特徴を要約すると，時間（計算速度）や空間（記憶域）といった計算資源をもっとも有効に活用し，高度な信頼性を持つソフトウェアを，高い生産性で開発する技術を確立する，ということになるだろう．

4 年半の期間の間，研究会を 12 回開催した．また，国際ワークショップとして WNASC(Workshop on New Approaches to Software Construction) を 2004 年，2005 年のいずれも 9 月に東京で開催した．

研究成果は学術論文として数多く発表され，高い評価を受けている．ソフトウェア研究では，具体的なソフトウェアやその適用事例が注目されがちであるが，ソフトウェアの基礎理論も世界に多くの優れた研究者がおり理論成果を競っている．その中で，このプロジェクトには多くの理論研究者が参加し，質の高い論文を世界に問うていることは，特筆できる．

もちろん直接的な成果としてのソフトウェアそのものにも，大いなる収穫があった．研究のツールとして役立つソフトウェアを公開することにより，他の研究者がそれを利用して研究を深める機会を広げたこともあれば，実用的な価値が認められてより広い利用者に迎えられたものもある．

このような課題にさまざまなアプローチで取り組んできたわれわれの A01 柱の研究は幅広く，多くの成果を挙げてきたと自負できるが，その全貌をここで伝えることはとてもできない．そこで以下では 3 つの研究例に絞ってその概要を紹介する．もちろん，この 3 例に優るとも劣らない研究は枚挙できるが，それぞれ特徴が異なる 3 つの例として取り上げるものである．

1 つ目はソフトウェア工学分野から，コンポーネントに基づくソフトウェア開発技術を取り上げる．2 つ目に紹介するのは，ユニークな適用対象としてのロボットを動かすソフトウェアの研究である．3 つ目はソフトウェアの基礎理論分野から，プログラム変換に関する研究事例を示す．

2. コンポーネントに基づいた信頼性の高いソフトウェア開発

2.1 ソフトウェアの複雑さ

炊飯器からクルマまで，人々が意識しないようなあらゆるところでソフトウェアが使われている現在，社会が必要とするソフトウェアを効率よく，しかも正しく作成することがきわめて重要である．ところがそのソフトウェアは，機能の上でも構造においても，ますます複雑さを増し，手に負えない怪物になろうとしている．これを神話にたとえれば，古事記に出てくるヤマタノオロチのようなものといえるかもしれない．

ヤマタノオロチは 2 つの面でやっかいな代物である．

(1) 構造の複雑さ： 8 つの頭と 8 つの尾を持つという構造上の複雑性をもつ．
(2) 振舞いの複雑さ： 若い娘を喰らい，8 つの山と谷を破壊するという振舞い上の複雑性をもつ．

同じように，現代のソフトウェアも構造と振舞いの上

で，大いなる複雑性をもつ．

(1) 構造の複雑さ: プログラムの規模が大きく多くの部分で構成されるだけでなく，その部分間の関係が入り組んでいる．

(2) 振舞いの複雑さ： プログラムの構成要素がそれぞれ複雑な振舞いをする上に，それが組み合わされた全体の動作を把握することが困難である．

スサノオはヤマタノオロチを退治した．その戦略は明快である．

(1) 振舞いの複雑さに対しては，8 つの樽に酒を用意してヤマタノオロチを酔わせ，その振舞いを手なずけた．

(2) 構造の複雑さに対しては，剣で 8 つの頭と尾を断ち切った．

同じように，ソフトウェアの複雑さに対するわれわれの戦略も，明快である．

(1) 構造の複雑さに対しては，簡明で柔軟なコンポーネント化手法によって対応する．

(2) 振舞いの複雑さに対しては，コンポーネントとその合成システムに対する形式的な検証手法によって対応する．

以下，この 2 つの戦略を簡単に紹介する．

2.2 簡明で柔軟なコンポーネント化手法

コンポーネントとはソフトウェアの部品のことである．コンポーネントはどのような機能をもちどのように振舞うかが，明快でなければならない．また，コンポーネントを組み合わせてシステムを作る際に，柔軟な結合を許すものでなければ不便である．さらに，ソフトウェアは常に利用者の要求や環境の変化に対応して変更を求められるので，それに応じられるという意味でも柔軟性を必要とする．

そのような簡明で柔軟なコンポーネントの切り分けと，それを合成する基本的な枠組み（建築の用語を借用して「アーキテクチャ」と呼ばれることが多い）を与えるものとして，われわれは 2 つの技術を土台とした．一つは最近広く注目を集めているアスペクト指向技術である．われわれは早くからこの技術の発展に取り組み，多くの新しいアイディアとそれに基づく計算モデルや言語を提案してきた．もう一つは役割モデルである．オブジェクト指向技術の中心概念であるオブジェクトは，一度定義されるとその形態や機能を時とともに変化させることが難しいが，われわれが提案した役割モデルでは，オブジェクトの協調動作する場を独立な環境として記述できるようにし，オブジェクトはそこに参入して役割を獲得したり，またそれを手放すことを可能として，柔軟に動作を変化させることができるようにしている．このアスペクトや役割の場が，新たなコンポーネントの単位となることで，従来のプログラムの部品単位と比べ，はるかに豊かな表現力と柔軟性を実現した．

2.3 形式的な検証手法

プログラムの振舞いに誤り，不具合がないかどうかを確かめるには，通常テストがおこなわれる．しかし，テストで場合を尽くすことはできず，どれだけ多くのテストをしても完全に信頼できるというレベルには達しない．それに対し，形式化された枠組みを用い，プログラムやそのもととなる設計モデルの性質を数学的に検証することができれば，テストの持つ限界を打破することができる．

われわれは，コンポーネントとコンポーネントの合成で作られるシステムに対し，その振舞いがもつ性質を形式的に検証する手法を研究した．その基盤となる技術には，大きく分けると 2 つがある．第一は型システムである．データの型に関する理論は，関数型プログラミング言語を中心に発達したが，オブジェクト指向言語もクラスという型によってプログラムが構成されるという意味で，型を中心概念としている．われわれは新たに提案したコンポーネント記述の表現形式が，型システムとして健全であることを形式的に証明することにより，安心して使えるものとした．第二はプログラムの振舞いを状態遷移モデルで捉え，その性質を時相論理式で記述して，その整合性をモデル検査という手法で検証とするという方法である．コンポーネントのアーキテクチャに対してこの方法を適用し，信頼性を確かめた．

2.4 組木プロジェクト

われわれはこの研究を組木プロジェクトと名づけた．組木は図 1 のようなコンポーネントから，図 2 のような作品を組み立てる．日本の伝統的な工芸である組木細工のひそみに倣い，これからもソフトウェアのコンポーネント技術をさらに発展させていきたい．

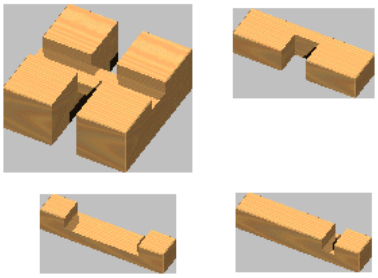


図 1 組木のコンポーネント

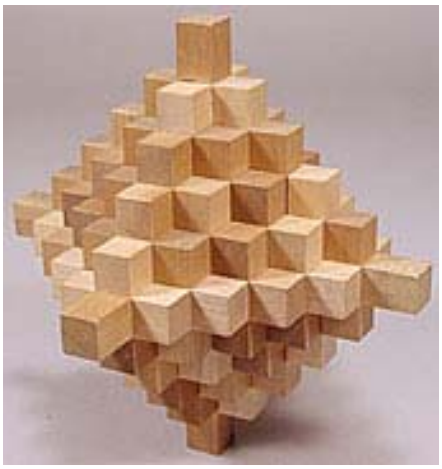


図 2 51 本組木 (製作: 山中組木工房，画像提供: 百町森)

なお，この節で紹介したプロジェクトの成果は，数多くの論文や解説としてさまざまな形で出版されており，参考文献として挙げるにはスペースが足りない．興味をもたれた読者は，A01 柱全体の報告書[6] や共通のウエブサイト[8] にリンクされたこのプロジェクトのページから，論文リストをたどられるようにお願いしたい．

3. ロボットのソフトウェア

ロボットは，1960 年代後半より計算機を用いて感覚情報に基づいた動作制御を行うことが可能となった．1980 年代には，産業用として工場で溶接，組立作業などを行うアーム型ロボットや物を運搬する移動ロボットが実用化された．1990 年代後半には，人間のように自立して腕と脚を扱える人間型ロボット（ヒューマノイドロボット）の実現が可能となった．

2003 年には，国のプロジェクトで開発された等身大のヒューマノイドロボットを，研究プラットフォームとして大学・国立研究機関で利用できるようになった．また，卓上で利用可能な小型の人間型ロボットも趣味の世界で利用できるようになり，図 3 左に示したような視聴覚機能をもつ研究用小型ヒューマノイドも市販されるようになった．これらはロボットの身体を持つ計算機周辺装置といえ，その行動を制御するのは周辺装置の制御プログラムの一種になる．図 3 右の図は，ヒューマノイドロボットの基本的な行動制御の構造を示している．

ヒューマノイドロボットは，四肢のそれぞれに 6 個程度の関節軸をもち，その関節の角度や発生力を実時間制御しながら，全身の安定を保ちつつ歩行や物体操作といった行動を行う．目的行動を決定したり行動結果を評価し判断するには，視覚・聴覚などの感覚系をもちいる．このような多種類の感覚系と全身の関節の動作系とを結んで，安定で環境に適応可能な行動として実現するのが，計算機上のソフトウェアの役目となる．そのようなヒューマノイド用ソフトウェアは，高度な認識判断行動の情報処理を行いながら実時間制御を行わなければならない．認識判断行動のプログラムの実装には，メモリを動的に割り付けて再利用する仕組みが不可欠であり，そのメモリ管理機構が実時間制御を阻害しないようにする必要がある．

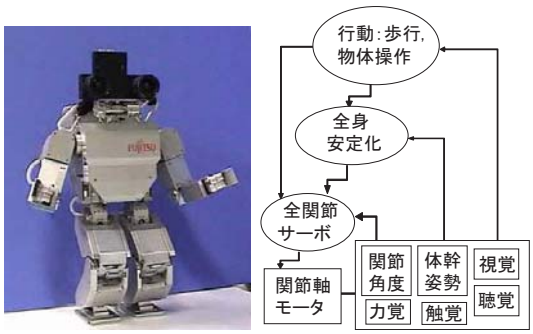


図 3 人型ロボットとプログラムの制御構造

A01 柱のプロジェクトの一つとして，ヒューマノイドロボットのソフトウェアの基盤環境を実現するために，「マルチスレッド Lisp の実時間 GC 機能の導入とヒューマノイド行動の実現」というテーマの研究を行なった．認識判断行動の情報処理部を記述するために並列処理記述が可能な Lisp を用い，その Lisp に実時間自動ごみ集め機能を搭載し，図 3 のヒューマノイドロボットの実時間制御を行うというものである．

実時間歩行制御を行っている際に，実時間対応していない自動ごみ集め処理が起動されると，運動タイミングがずれ，転倒してしまうことが起こりうる．そこで実時間歩行中に随時自動ごみ集めが可能となるような処理を実現する必要がある．そのような実時間ごみを集めを組み込んだソフトウェアを実現し，ヒューマノイドロボットの実機で評価実験を行った．

図 4 は，そのヒューマノイド実験システムの構成図である．実時間 OS として RT-Linux を用い，マルチスレッド Lisp としては，旧電総研で松井俊浩氏らにより開発された Euslisp[1] を用いた．実時間ごみ集め処理は京都大学湯浅研究室のリターンバリア機構を導入した実装がなされ[2]，ロボットの行動ソフトウェアとの統合は東京大学稲葉研究室でなされた[3]．

図 5 と図 6 は，ロボットが屈伸を行った際の，実時間ごみ集め機能のない場合とある場合での行動実験

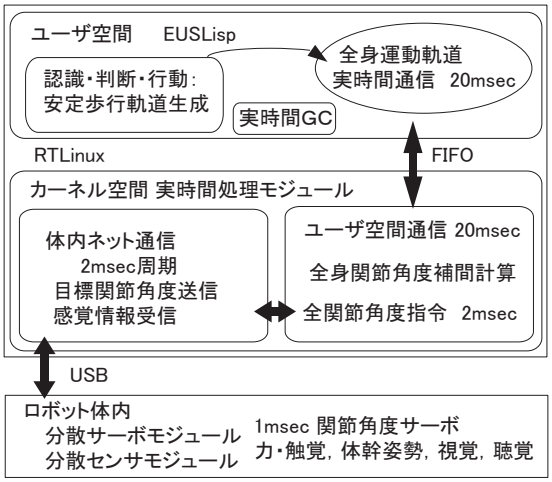


図 4 ロボットソフトウェアのプログラム構成

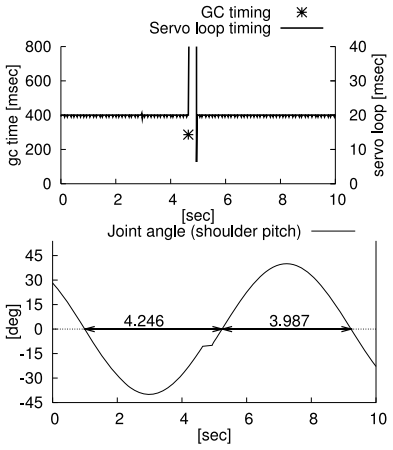


図 5 一括 GC を持つ EusLisp の挙動．上）GC のタイミングと制御ループ周期，下）関節角度

結果をそれぞれ示している．それぞれで上のグラフが GC の処理時間，下がロボットの膝関節の角度を示している．図 5 では，5 秒のあたりで一括 GC が起動し，膝の周期運動が 0.3 秒ほど止まっているが，図 6 の実時間 GC の方は周期性がほぼ保たれている．

図 7 は，歩行動作時の実験例であり，一括 GC の場合の転倒例と実時間 GC の際の転倒していない動作例を示している．

ロボットのソフトウェアには，物理的な環境からの制約で実時間性を必要とする制御と，時間制約よりは上位の認識性能や適応的な判断能力を実現する機能との融合を，ソフトウェア基盤の上で実現することが求められている．少子高齢化社会の時代に，ロボットが人と社会を支援することが広く求められている状況で，ロボットのソフトウェアの果たすべき役割が高まって

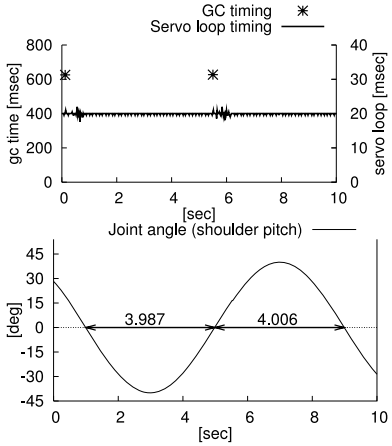


図 6 実時間 GC を持つ EusLisp 挙動．上）GC のタイミングと制御ループ周期，下）関節角度

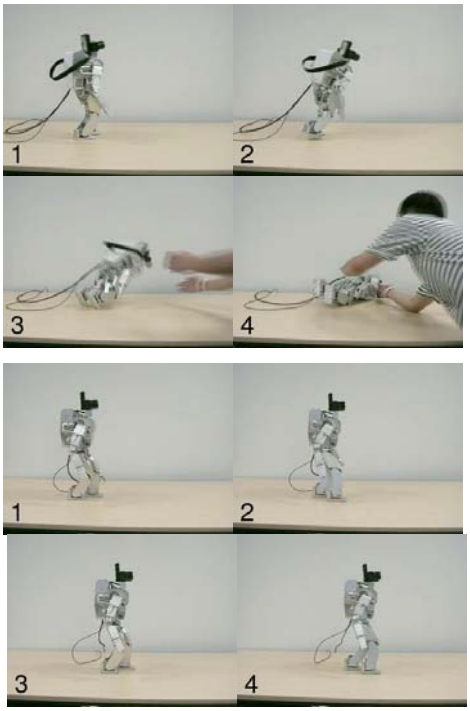


図 7 歩行動作実験．上）一括ＧＣが起こったと場合，下）実時間ＧＣ環境での歩行動作

おり，情報学の分野におけるロボットのソフトウェア研究の発展が期待されている．

4. 定理自動証明とプログラム変換の融合

基本的なプログラムや仕様から出発して，プログラムの生成・変換・検証を繰り返しながら，徐々にプログラムを発展・進化させる発展的プログラミングは，

大規模・高信頼ソフトウェアの開発手法として有効である．このような発展過程において重要な役割をはたす技術がプログラムの自動変換である．われわれは新しいプログラム変換の理論を確立する目的で，書き換えシステムを基礎とした関数型プログラムの変換に関する研究を進めて来た．関数型プログラムの変換は，書き換えシステムの等価変換として数学的にモデル化できる．一方，自動証明の分野では書き換えシステムの**完備化**が広く利用されている．これは，書き換えシステムの等価変換によって，等式証明をリダクションによる効率的な計算に置き換えるものである．

A01 柱の研究の一つとして，プログラムの**融合変換**や**パターンに基づくプログラム変換**の手法と，**完備化**や**書き換え帰納法**に代表される定理自動証明の手法の関係を理論的に考察し，両者の特長を組み合わせた新しいプログラム変換手法の提案をした．すなわち，プログラム変換と定理自動証明を書き換えシステムの枠組みをもちいて形式的にモデル化し，定理自動証明とプログラム変換の両者に共通な理論的基礎を確立するとともに，それに基づいた実験システムを構築して有効性を確認する．このために，プログラム変換に不可欠な高階書き換えシステムの停止条件，高階パターマッチング手法，高階帰納法の証明手法などに関する理論の確立を試みた．さらに，これらの理論的成果の上に，パターンに基づくプログラム変換システムのプロトタイプを実装した[4),5)]．

4.1 パターンに基づくプログラム変換

プログラムの計算量の効率化にはいくつかの定石がある．これらの定石を高階の変換パターンで表現し，パターン照合によるプログラム変換でプログラムの効率を改良する方法が，パターンに基づくプログラム変換である (図 8)．変換の正当性を検証するためには，プログラムのさまざまな帰納的性質を証明する必要がある．そこで，プログラムを書き換えシステムでモデル化し，定理自動証明の手法である潜在帰納法や書き換え帰納法を適用することによって，プログラムの変換と正当性の検証を完全に自動化したシステムを実現した．

プログラム変換システム RAPT (Rewriting-based Automated Program Transformation system) は，項書き換え理論に基づいて設計されており，プログラムは項書き換えシステム，変換パターンはパターン変数を含む項書き換えシステムにより表現されている．このため，項書き換えに基づく定理自動証明手法を適用することで，変換の正当性を保証するためのプログラムのさまざまな帰納的性質 (例えば加算の結合律や交換律など) を自動的に検証することが可能となる．本システムの大きな特徴は，このようにプログラム変換の理論と定理自動証明の理論を項書き換えシステムという共通の枠組みの中でうまく融合させることにより，プログラム変換の正当性を完全に自動検証できる点である．

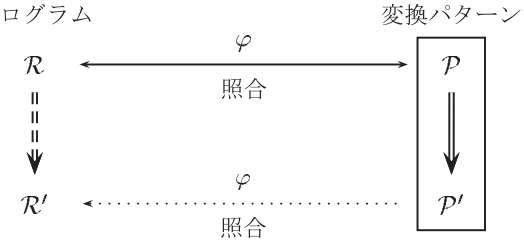


図 8 パターンに基づくプログラム変換

以下の $\mathcal{R}$ はリストの総和を求める関数 sum を再帰的に定義する項書き換えシステムである．ここで，$0, \mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)), \ldots$ によって，自然数 $0, 1, 2, \ldots$ を表現している．

$$\mathcal{R} \left\{ \begin{array}{lcl} \mathsf{sum}([\,]) & \to & 0 \\ \mathsf{sum}(x:y) & \to & +(x, \mathsf{sum}(y)) \\ +(0, x) & \to & x \\ +(\mathsf{s}(x), y) & \to & \mathsf{s}(+(x, y)) \end{array} \right.$$

変換パターンはパターン変数を用いて抽象化した項書き換えシステム (項書き換えパターン) によって構成される．以下の変換パターン $\mathcal{P} \Rightarrow \mathcal{P}'$ を用いて $\mathcal{R}$ を変換することを考える．

$$\mathcal{P} \left\{ \begin{array}{lcl} \mathsf{f}(\mathsf{a}) & \to & \mathsf{b} \\ \mathsf{f}(\mathsf{c}(u, v)) & \to & \mathsf{g}(u, \mathsf{f}(v)) \\ \mathsf{g}(\mathsf{b}, u) & \to & u \\ \mathsf{g}(\mathsf{d}(u, v), w) & \to & \mathsf{d}(u, \mathsf{g}(v, w)) \end{array} \right.$$

$$\mathcal{P}' \left\{ \begin{array}{lcl} \mathsf{f}(u) & \to & \mathsf{f}_1(u, \mathsf{b}) \\ \mathsf{f}_1(\mathsf{a}, u) & \to & u \\ \mathsf{f}_1(\mathsf{c}(u, v), w) & \to & \mathsf{f}_1(v, \mathsf{g}(w, u)) \\ \mathsf{g}(\mathsf{b}, u) & \to & u \\ \mathsf{g}(\mathsf{d}(u, v), w) & \to & \mathsf{d}(u, \mathsf{g}(v, w)) \end{array} \right.$$

項書き換えシステム $\mathcal{R}$ を変換パターン $\mathcal{P} \Rightarrow \mathcal{P}'$ を用いて変換するためには，最初に項書き換えパターン $\mathcal{P}$ と項書き換えシステム $\mathcal{R}$ とのパターン照合問題を解く必要がある．我々の提案した項パターン照合アルゴリズムをもちいて項書き換えパターン $\mathcal{P}$ と項書き換えシステム $\mathcal{R}$ のパターン照合を解くと以下の項準同型写像 φ が得られる．

$$\varphi = \left\{ \begin{array}{ll} \mathsf{f} \mapsto \mathsf{sum}(\square_1), & \mathsf{g} \mapsto +(\square_1, \square_2), \\ \mathsf{a} \mapsto [\,], & \mathsf{b} \mapsto 0, \\ \mathsf{c} \mapsto \square_1 : \square_2, & \mathsf{d} \mapsto \mathsf{s}(\square_2) \end{array} \right\}$$

パターン照合の解は一般に複数存在するが，ここではプログラム変換に適した解として，この研究で新たに導入した CS 準同型写像と呼ばれるものに限定している．この φ を項書き換えパターン $\mathcal{P}'$ に適用することで以下の項書き換えシステム $\mathcal{R}'$ が得られる．

$$\mathcal{R}' \quad \left\{ \begin{array}{lcl} \mathsf{sum}(x) & \to & \mathsf{sum1}(x, 0) \\ \mathsf{sum1}([\,], x) & \to & x \\ \mathsf{sum1}(x : y, z) & \to & \mathsf{sum1}(y, +(z, x)) \\ +(0, x) & \to & x \\ +(\mathsf{s}(x), y) & \to & \mathsf{s}(+(x, y)) \end{array} \right.$$

入力項書き換えシステム $\mathcal{R}$ と出力項書き換えシステム $\mathcal{R}'$ を比較すると，$\mathcal{R}$ は再帰的に関数 sum を定義しているのに対し，$\mathcal{R}'$ では反復的な定義となっている．再帰的なプログラムより反復的なプログラムの方が効率的であることはよく知られており，実際，$\mathcal{R}'$ の方が $\mathcal{R}$ より効率的な定義となっている．

プログラム変換の正当性を保証するためには，一般的にはプログラムの帰納的な性質を検証する必要がある．例えば，上記の項書き換えシステム $\mathcal{R}$ から $\mathcal{R}'$ へのプログラム変換において，変換の正当性を保証するためには関数 $+$ の結合律と $+(0, n) = +(n, 0)$ という 2 つの帰納的性質が成立している必要がある．このような帰納的性質を変換パターンのレベルで抽象化したものが，以下の仮定 $\mathcal{H}$ である．

$$\mathcal{H} \quad \left\{ \begin{array}{lcl} \mathsf{g}(\mathsf{b}, u) & \approx & \mathsf{g}(u, \mathsf{b}) \\ \mathsf{g}(\mathsf{g}(u, v), w) & \approx & \mathsf{g}(u, \mathsf{g}(v, w)) \end{array} \right.$$

変換パターンを正しく適用するためには，照合された項書き換えシステムが仮定 $\mathcal{H}$ をみたす必要がある．これまで提案されていた変換パターン基づくプログラム変換システムでは，仮定 $\mathcal{H}$ の検証はユーザーに委ねられていた．一方，RAPT では，変換の正当性を保証するために必要な項書き換えシステムの停止性，合流性，十分完全性の自動検証を行うとともに，書き換え帰納法を利用した仮定 $\mathcal{H}$ の自動検証も行う．このような一連の自動検証は，RAPT のように定理自動証明とプログラム変換が共通の枠組みの中で有機的に連携して働くことによって初めて可能となる．

4.2 プログラム変換システム RAPT

RAPT は与えられた多ソート項書き換えシステムを，与えられた変換パターンに基づいて変換し，それと等価な多ソート項書き換えシステムを出力する．RAPT の全体構成を図 9 に示す．実線の矢印はデータの流れを表わし，点線の矢印は各部分で得られた情報がどこで用いられるかを示している．RAPT の実装は Standard ML of New Jersey によって行われ，ソースコードは約 5,000 行である．

入力検証部では，入力項書き換えシステムのプログラムとしての構文的な正しさを検証する．簡約順序検出部では，制約解消アルゴリズムと辞書式経路順序をもちいて入力項書き換えシステムの停止性を判定する．合流性・十分完全性検証部では，入力項書き換えシステムの合流性および十分完全性の自動証明を行う．入力項書き換えシステムの停止性はすでに保証されているため，合流性の判定には危険対の合流性を判定し，十分完全性については擬簡約性を判定する．パターン照合部では，照合アルゴリズムを用いて項書き換えパターンから項書き換えシステムへの CS 準同形写像を求める．照合アルゴリズムでは，主関数から副関数へと先に得られた照合解を利用しながら照合問題を効率的に解くように工夫されている．仮定検証部では，パターン照合部で得られた CS 準同形写像を用いて仮定を具体化し，それが入力項書き換えシステムの帰納的定理であることを書き換え帰納法によって自動証明する．出力検証部では，得られた出力項書き換えシステムのプログラムとしての構文的な正しさとともに、停止性と十分完全性を検証している．

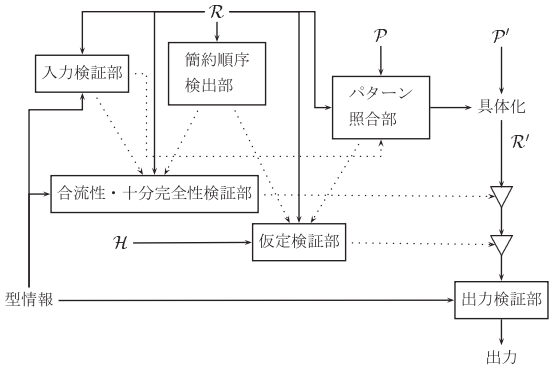


図 9 RAPT の全体構成

5. 終わりに

以上紹介した 3 つの研究例それぞれも，紙幅の関係でごく簡略な説明に留まっている．そして繰り返しに

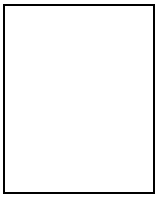
なるが，これ以外にも実に豊富な研究事例がある．それらの詳細については，まず平成 13 年度から 17 年度にわたり毎年公表してきた報告書[6] を参照されたい．また，領域全体を一般向けに分かりやすく紹介した冊子として,「情報学を創る」[7] がある．さらにウエブサイト[8] から，個々のプロジェクトのページにアクセスが可能である．

この 4 年半のプロジェクトを振り返ると，さまざまな思いがこみあげてくる．2006 年 1 月の成果報告会で各プロジェクトの最終報告をしてもらったとき，多くの発表者が「感無量の思いがある」と述べたのには，まったく同感だった．第 1 節に書いたように，この間 12 回の研究会を開いたが，そのほかにもさまざまな形で，A01 の多くの参加者が集まる機会があった．それらを通じた研究の交流とひとびとのつながりが，今後のわれわれの大きな財産になるであろう．そういう意味でも，研究代表者の安西祐一郎氏初め多くの関係者の方々に，心より感謝したい．

参 考 文 献

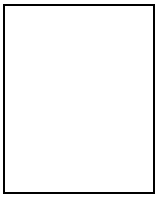
1) 松井俊浩, 関口智嗣. マルチスレッドを用いた並列 euslisp の設計と実現. 情報処理学会論文誌, Vol.36, No.8, pp. 1885 – 1896, 1995.
2) 湯浅太一, 中川雄一郎, 小宮常康, 八杉昌宏. リターン・バリア. 情報処理学会論文誌, Vol.41, No. SIG9(PRO 8), pp. 87–99, 2000.
3) 稲葉雅幸, 岡田慧, 水内郁夫, 稲邑哲也. ヒューマノイドロボットのシステム実現 - ロボットシステム記述言語 EusLisp による実装. コンピュータソフトウェア, Vol.23, No.2, pp. 45–61, 2006.
4) Y. Chiba, T. Aoto, and Y. Toyama, Program transformation by templates based on term rewriting, *Proceedings of the 7th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP 2005)*, 59–69, 2005.
5) Yuki Chiba and Takahito Aoto, RAPT: A Program Transformation System based on Term Rewriting, *Proceedings of the 17th International Conference on Rewriting Techniques and Applications (RTA 2006), Lecture Notes in Computer Science, Vol.4098*, 267-276, 2006.
6) 安西祐一郎 (発行責任), 玉井哲雄 (編集),「IT の深化の基盤を拓く情報学研究 研究成果報告書 A01 新しいソフトウェアの実現」
7) 安西祐一郎 (発行責任), 安達淳 (編集),「情報学を創る」, 2006 年 3 月 8 日発行.
8) http://research.nii.ac.jp/kaken-johogaku/

(平成 年 月 日受付)
(平成 年 月 日採録)

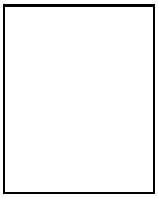


玉井 哲雄（正会員）

1948 年生．1972 年東京大学大学院工学系研究科計数工学専攻修士課程修了．同年 (株) 三菱総合研究所入社．1985 年同社人工知能開発室室長．1989 年筑波大学大学院経営システム科学専攻助教授．1994 年東京大学教養学部教授，2000 年同大学院情報学環教授，2003 年同大学院総合文化研究科教授，現在に至る．工学博士．ソフトウェアの仕様・検証技術，モデル化技術，進化プロセスの分析，協調計算モデルの開発，等の研究及びそれらの技術の実際的な問題への適用に従事．著書に「ソフトウェア工学の基礎」(岩波書店，2004，大川出版賞受賞),「ソフトウェアのテスト技法」(共立出版，1988) など．日本ソフトウェア科学会，情報処理学会，電子情報通信学会，日本オペレーショズリサーチ学会，人工知能学会，ACM, IEEE 各会員．



稲葉 雅幸（正会員）



外山 芳人（正会員）