「情報処理学会論文誌：プログラミング」の編集について

プログラミング研究会論文誌編集委員会

情報処理学会では，研究会の活性化を目指して様々な改革を進めている．プログラミング研究会はこの流れを受けて，研究会のあるべき姿について徹底的な討論を行ってきた．その帰結として，研究会独自の論文誌の編集にいち早く踏み切ることを決定した．

研究会論文誌「情報処理学会論文誌：プログラミング」の特徴と意義は大きく3つある．第1は，従来の「論文」に対して想定されてきた対象分野や査読基準では必ずしもカバーしきれない，多様な成果の公表の場を提供することである．第2は，投稿論文の内容を研究会で発表することを義務づけることによって，迅速で的確な査読を実現するとともに，議論の結果の最終稿へのフィードバックを可能にすることである．第3は，研究内容の表現に必要であると認められれば，長大な論文も採録可能としている点である．

本論文誌を通じて，日本のプログラミング分野の研究活動を盛り上げていきたい．読者諸氏からの多くの論文投稿を期待する．

1. 対象分野

プログラミングは，コンピュータの誕生と同時に生まれた伝統的な分野であるが，コンピュータがある限り不可欠な技術である．並列分散処理やマルチメディア応用など処理内容が高度になるにつれて，プログラミングの重要性は増すことがあっても減ることはないであろう．

「情報処理学会論文誌：プログラミング」は，プログラミングに関するテーマ全般を専門に扱う論文誌である．具体例として次のようなテーマがあげられる．

- プログラミング言語の設計，処理系の実装
- プログラミングの理論，基本概念
- プログラミング環境，支援システム
- プログラミング方法論，パラダイム

これらを応用したシステムの開発事例も対象に含まれる．また，上記以外でも，プログラミングに関する面白い話題であれば対象となる．

2. 編集方針

本論文誌は，プログラミング研究会における発表と論文誌投稿が密接にリンクされている点に特徴がある．論文誌への投稿者が用意する研究会発表用の資料が，そのまま本論文誌への投稿論文となる．

研究会発表をせずに本論文誌に投稿することはできないが，逆に，本論文誌への投稿をともなわない研究会発表は可能である．そのような発表や，論文が不採録となった発表については，アブストラクトが本論文誌に掲載される．従来のプログラミング研究会の研究報告は廃止し，その代わりとして，研究会登録者には本論文誌が配布される．

本論文誌に掲載する論文は，通常のオリジナル論文と，サーベイ論文の2種類とする．どちらの種類であるかは，著者自身の指定によって決まる．論文の記述言語は日本語，英語のいずれかとする．論文の長さに制限は設けない．

3. 査読基準

基本的に，減点法に陥ることを避け，論文の良い点を積極的に評価するという方針を貫く．具体的には，新規性，有効性などの評価項目のうち，どれか1つの点で特に優れていると認められれば採録する．体裁のみが整った論文より，若干の不備はあっても技術的な貢献の大きい論文を積極的に受け入れる．

このような観点から，たとえば次にあげるような，従来は論文としてまとめることが難しかった内容について論じた論文もできるだけ受け入れる．

- プログラミング言語の設計論
- システムの開発経験に関する報告
- 斬新なアイディアの提案
- 概念の整理，分類法，尺度の提案
- 複数のシステムその他の比較

4. 投稿から掲載までの流れ

本論文誌への投稿希望者，および研究会での発表希望者は，発表会開催日の 2〜3 カ月前までに発表申込みをする．具体的な方法は研究会ホームページ `http://www.ipsj.or.jp/sig/pro/` を参照していただきたい．申し込みの際には，本論文誌への投稿の有無，オリジナル論文とサーベイ論文の種別指定を明記する．また，アブストラクト（和英両方，和文は600字程度）を添付する．

論文投稿を希望した場合は，研究発表会の 3 週間前までに，別に定めるスタイル基準に従ったカメラレディ形式で論文を提出する．

毎回の研究発表会の直後，編集委員会が開催され，各論文について 1 名の査読者が決定される．査読報告をもとに，編集委員会は採録，条件付き採録，不採録のいずれかの判定を行い，発表会開催後 3 週間程度で発表者に採否通知を行う．照会の手続きはないが，論文改善のための付帯意見が添付される場合がある．この場合は，3 週間以内に改良版を作成する．

5. 研究発表会

プログラミング研究会では，発表会ごとに特集テーマを設けている．ただし各発表会では，特集以外の一般の発表もつねに受け付けている．

2001 年度の発表会予定は次のとおりであり，各発表会の特集テーマは，今後数年間はそのまま維持する予定である．

6 月 21 ～ 22 日［プログラミング言語の設計と実装］
7 月 25 ～ 27 日［SWoPP—並列/分散/協調プログラミング言語と処理系］
10 月 22 ～ 23 日［理論］
1 月 29 ～ 30 日 [並列・分散処理]
3 月 [プログラミング言語一般]

6. 編 集 母 体

本論文誌は，下記のプログラミング研究会論文誌編集委員会の責任で編集を行う．各研究発表会ごとに担当編集委員が割り当てられ，投稿論文の査読プロセスを主導する．2000 年度より論文誌編集委員会メンバを増やし，各研究発表会を 2 名の編集委員で担当している．

本号の編集にあたって

久野　靖，前田敦司

2001 年度第 1 回プログラミング研究会は，2001 年 6 月 21 日より 22 日まで情報処理学会会議室にて開催された．特集テーマは「プログラミング言語の設計と実装」であったが，特集テーマのみに制限せずそれ以外の発表についても申し込みを歓迎している．

今回は 16 件の発表が行われ，発表 25 分，質疑 20 分の発表時間を考慮すると 2 日間の日程としてはやや多い件数となった．結果として，コンパイラの最適化技法や実行時システムなど言語処理系の実装手法を中心に，プログラミング環境や言語の提案，プログラム解析のアルゴリズムなど，ほぼテーマに沿った領域を広くカバーするプログラムとなった．

投稿原稿の査読を議論する編集委員会会合は，編集委員ならびに編集委員会が出席を依頼したメンバで現地にて開催した．ただし，投稿論文の共著者となっているメンバは，その論文についての議論の間は退席している．各開催日研究会終了後に計 2 回の会合を行った．十分な議論の後，各投稿論文について担当の査読者を決定し，査読を依頼した．発表件数が多かったこともあり，遅くまで議論いただいた委員会参加者の皆様にはその労を多としたい．成果である本誌の内容によって，その労苦が報われているものと信じる．

最終的に，投稿を希望したうち 11 件の論文（すべて通常論文）が採録となった．これらの論文の掲載に続き，それ以外の発表については 1 ページの概要を掲載してある．掲載順序は，論文，概要それぞれについて当日の発表順に従うことにした．

掲載した論文について，以下，簡単に紹介する．

「圧縮方式による世代別ガーベッジコレクションの実装について」（Implementation of Generational Garbage Collection Based on Mark-and-compact）で著者らは，データの配置順序を保存する圧縮方式に基づいた世代別ガーベッジコレクションの効率的な実装とその評価について述べている．また，圧縮方式の特性を活かして多世代管理に似た効果的な世代管理を行う手法や，対象領域の動的変更機能等の手法の提案も行っている．

「メモリ管理機能のモジュラーかつ効率的な実装手法」（Modular and Efficient Implementation Scheme for Memory Management Systems）では，著者らは，メモリ管理機能のモジュラーな実装を可能にするために，実行時システムとメモリ管理機能との間に抽象的なインタフェースを定義し，さらに効率化のために，実行時システムの特化手法を提案している．提案した手法にしたがって既存の処理系を再実装し，提案した特化手法がモジュラーな実装にともなうオーバヘッドを解消し，既存の処理系と同程度の性能を達成することを示している．

「オブジェクト指向スクリプト言語 Ruby への世代別ごみ集め実装手法の改良とその評価」（Improvement of the Implementation of Generational Garbage Collection in Object-oriented Scripting Language Ruby and Its Evaluation）では，著者らは，オブジェクト指向スクリプト言語 Ruby に著者ら自身が以前に導入した世代別 GC を改良し，以前の手法の問題点であったメモリ使用量の改善と拡張ライブラリ作成者への負担の軽減を図っている．さらに，速度の向上を図るために，タイプ別ルート選択の手法を提案し，評価を行うことで，改良手法の有効性を示している．

「ドラッグ&ドロップを用いたビジュアルプログラミングシステム」（Drag and Drop Based Visual Programming System）は，ノードとエッジで表現されたプログラムをドラッグ&ドロップにより直接的に操作できるビジュアルプログラミングシステム CafePie に関する報告である．著者らが実装した CafePie は，プログラム構造とプログラムの実行を同じアイコンを用いて視覚化する．また，プログラムの識別を容易にするために視覚的表現のカスタマイズができる仕組みを与え，その操作も同様なドラッグ&ドロップを用いて可能としている．

「非循環グラフにおける支配関係の簡潔な検出算法」（A Simple Algorithm to Identify Dominance Relations in DAG）で著者らは，非循環グラフについて支配木や支配辺境を求めるアルゴリズムを与え，計算量の考察と実プログラムを材料とした評価結果を報告している．スタックを導入することにより，従来の手法に比して簡潔なアルゴリズムとなり，また速度も約 3 倍に向上している．実プログラムの制御フローグラフのほとんどを占める可約グラフは，実質的に非循環グラフと等価と見なすことができ，本手法は実用上の貢献も大きいといえる．

「冗長な符号拡張命令の除去手法」（Elimination of Redundant Sign Extensions）は，Java 言語で書かれたプログラムなどのように 32 ビットの整数データを扱うプログラムを 64 ビットアーキテクチャ上で実行させる際，効果的に符号拡張命令の除去と移動を行う新しいアルゴリズムを提案している．この手法は，より多く実行される場所から順に，変数の依存関係を利用して除去を行う．また，配列に関する言語仕様を符号拡張命令の除去に利用する手法や，符号拡張命令をより実行頻度の少ない場所に移動させる手法もあわせて，Java JIT コンパイラ上に実装し評価を行っている．その結果，符号拡張命令の実行回数を平均で 87%削減できることを示している．

「動的コンパイラのための実行時分岐予測情報を用いた最適化手法」（An Optimization Technique Using Profile Based Branch Prediction for Dynamic Compilers）は，Java プログラムを対象として，関数内の局所的な実行時情報を利用した最適化手法について議論している．動的コンパイラが収集する実行時情報に課される，収集量と収集時期の大きな制約のもとで，分岐予測の精度を向上させる手法について考察している．また多分岐命令において，履歴情報を用いるだけでなく，分岐確率が分散しているときにも速度低下を引き起こさないコード生成手法を提案している．これらの手法をいくつかのベンチマークプログラムを用いて評価し，その有効性を示している．

「組込み機器向け Java2C トランスレータにおける 2 返戻値法を使った例外処理の実現」（Implementation of Exception Handling by Two Return Value Method in a Java2C Translator for Embedded Machines）で，著者は，Java to C トランスレータが生成する C ソースのコードサイズを抑止するために，例外処理の実現方法に配慮する必要があることを指摘

し，冗長な例外発生検査の除去や，例外オブジェクトの生成を `catch` まで遅延する技法を提案している．SPECjvm98 による評価の結果，提案技法は，下方移動による例外発生検査の集約とあわせて，コードサイズを相加平均で 4.71% 抑止できることを示している．

「命令レベル並列計算機上で並列実行する領域の選択を高速かつ効率的に行う方法」（A Fast and Efficient Method to Generate Hyper Block for Instruction Level Parallel Processors）は，条件分岐を削除して分岐先命令群に相補的なプレディケートを付けたり（IF 変換），制御依存やメモリ依存を超えた投機的な命令移動によってクリティカルパスを縮めたりする最適化の適用領域を適切に選択する手法の提案と評価を行った論文である．適用領域を定めた場合の実行効率を，その領域を構成する部分の実行効率から再帰的に見積もれると仮定してモデル化し，動的計画法を適用することで，広範囲の問題に対して高い実行効率を実現する高速な手法を示している．

「プロセッサグループの動的分割による並列再帰プログラムの実現手法」（Implementation of a Parallel Recursive Program Based on Dynamic Division of Processor Groups）では，著者らは，並列再帰呼び出しを可能とする並列言語処理系の多くが用いているマネージャ・ワーカ法と呼ばれる動的負荷分散方式において，並列実行する再帰処理にプロセッサグループを割り当て，プロセッサ管理を各グループごとに行うことで，プロセッサ管理負荷を分散する方式を提案している．さらに，並列再帰呼び出し時のプロセッサ管理に要する通信回数を減らすことで，プロセッサ管理負荷の軽減を図り，また，再帰関数中のデータ並列性も利用することで，より高い性能向上を目指している．実機上で詳細な性能評価を行って有効性を示している．

「共有メモリ向けプリミティブとその GCC を使った実現」（Primitives for Shared Memory and Its Implementation with GCC）の著者らは，最近の多くのプロセッサのメモリモデルや不可分命令に関する詳細な調査を行っている．この調査の結果をもとに，アセンブリコードの代わりに C 言語を生成する高水準な並列処理言語のコンパイラにおいて，移植性や実行効率を高めるため，不可分操作やメモリアクセス完了順序などを記述できるように C 言語を拡張する，共有メモリ向けプリミティブの設計を示している．さらに，既存のシステムでもできるだけ利用可能なように考慮を加えて変更したプリミティブを，GCC の拡張機能を用いて実現している．

最後に，研究会開催および論文誌編集にさまざまなご協力をたまわった皆様に深い感謝を捧げたい．