「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］
2002 年 1 月頃 ［並列・分散処理］
2002 年 3 月頃 ［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と副担当の編集委員 2 名が割り当てられ，投稿論文の査読プロセスを主導する．

本号の編集にあたって

小川瑞史，西崎真也

2000 年度第 3 回プログラミング研究会は，2000 年 11 月 16 日より 18 日まで公立はこだて未来大学にて開催された．特集テーマは「理論」であったが，特集テーマのみに制限せず，それ以外の発表また論文投稿をともなわない発表についても歓迎した．結果として 16 日はプログラミング言語の設計や概念の提案，17 日は理論，18 日は実装がそれぞれ中心となるプログラムとなった．

当初の予定では 11 月 16 日と 17 日の 2 日間の開催で最大 13 件の発表（各発表時間は発表 25 分，質疑 20 分）を想定していたが，申込み期限の 8 月 30 日までに 15 件の発表申込みがあり，発表者の都合を確認したうえで急遽 11 月 16 日より 18 日までの 3 日間の開催に延長した．会場の手配などについては公立はこだて未来大学の大沢英一氏のお世話になりました．厚く御礼申し上げます．

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．各論文を議論するのに十分な時間を確保するために，各開催日研究会終了後ならびに昼休みに計 4 回の会合を設定した．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．

最終的に，投稿を希望したもののうち 6 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．

「ループアンローリングの特徴抽出とそのモデル化」で，著者らは，近年のスーパスカラプロセッサにおいてしばしば有効な最適化手法であるループアンローリングをモデル化し，その効果の定量的な解析を試みている．その結果，演算の実行時間とキャッシュミスによるコストの 2 つの要素がループアンローリングに影響を与えていること，特にキャッシュミスに関してはロードと参照の距離を考慮することが重要との解析結果を述べている．

「代入を用いた再帰除去と再帰導入（Recursion Removal and Introduction Using Assignments）」で，著者らは，構造データを返す関数に対するラムダ抽象化に相当する操作を用いて，再帰除去および（累積変数を用いる末尾再帰関数に対する）再帰導入の手法を提案している．ラムダ抽象化はコンストラクタの性質を用いた代入操作で実現し，再帰除去や再帰導入は代入操作を含む拡張言語上で抽象化および変換を行うことで実現する．これにより，変換の容易さと実行速度の向上が得られたとしている．

「高階変数を持つ項書換え系の停止性証明について（On Proving Termination of Term Rewriting Systems with Higher-Order Variables）」で，著者は項書換え系に高階変数を導入し，さらに項書換え系における停止性証明の標準的手法である再帰経路順序，および依存対解析と切り落とし法の拡張を提案している．高階変数を持つ項書換え系は λ 抽象化を持たない高階書換え系に相当し，通常の関数型プログラムなどを包含する．従来より高階書換え系の停止性証明手法はいくつか提案されているが，切り落とし法の適用が難しいなどの困難があった．ここでは高階変数を持つ項書換え系を対象とすることで，実用的な範囲の停止性証明を容易にしたとしている．

「Java 向け静的コンパイラによる仮想メソッド呼出しの高速化」で，著者は，仮想メソッド呼出しの高速化手法である I-call if 変換を 4 種類に分類し，Java 向け静的コンパイラにおけるそれらの得失を定量的評価に基づき比較している．I-call if 変換はクラスチェックとメソッドチェック，および限定的と非限定的の 2 つの軸の分類に基づき，それぞれコードの実行速度で勝る前者とリンクの遅延について勝る後者との組合せについて SPECjvm98 による評価を行っている．その結果，メソッドチェックによる限定的 I-call if 変換が最も実行速度が速い傾向があり，またメソッドチェック変換のみで最適化すると，クラスチェック変換のみで最適化する場合より最大で 9.83%高速化できると結論づけている．

「Java 上の Scheme 処理系「ぶぶ」における単一のクラスローダを用いたオブジェクトシステムの実装」で，著者らは，Java 上のオブジェクト指向型の Scheme 処理系「ぶぶ」のオブジェクトシステムの実装法について述べている．「ぶぶ」の従来のオブジェクトシステムでは Scheme のクラスの実装に Java のクラスそのものを用いていたため，セキュリティ上の制約からアプレット上でクラス生成ができない，クラス再定義やメソッドの追加・再定義の効率が悪いなどの問題があった．本論文では，Java のクラスそのものの代わりに Scheme のクラス情報を保持する Java オブジェクトを用いることで Java VM が提供するデフォルトのクラスローダが使用可能となり，それらを解決したとし

ている．

「融合型言語 TAO における構造データと未定義値の扱い」で，著者らは，Lisp に論理型言語の機能を融合した TAO86 および TAO という 2 つの言語における内部データ構造における未定義値（UNDEF）と参照（REF）の扱いの違いについて実装の詳細に踏み込んで述べている．TAO86 の実装では，UNDEF を即値とし，REF を処理系が自動的に辿る方法を用いる．これにより言語は単純となるが，REF や UNDEF を含む構造データの同一性判定に問題が生じ，アドホックな関数を導入している．TAO の実装も同様であるが，論理変数を述語の値として返す関数型の機能，パターンマッチによる同一性の判定の機能などを準備することで，これらの問題を解決したとしている．

最後に，研究会開催，論文誌編集に御協力を賜りました皆様に感謝いたします．