トピックス

FPCA'93

吉田 展子

FPCA'93 (Functional Programming Languages and Computer Architecture) は 6 月 9 日から 11 日までデンマークのコペンハーゲンで ACM SIGPLAN/SIGARCH が主宰となり開催された．FPCA は隔年で開かれる関数型プログラミング言語に関する幅広い分野の研究を対象とする国際会議であり，今年は関数型言語の設計，型システム，形式意味論，最適化，プログラムの翻訳，コンパイル手法，プログラミング手法，ガーベージコレクション，データフローモデル，関数型言語のためのアーキテクチャの設計など，従来の関数型言語における分野の研究だけでなく，状態・制御・非決定性・並行性などの新しい概念を取り入れることによって関数型プログラミングパラダイムをより一般化する研究などについての発表が行なわれた．

会議の開催されたコペンハーゲンはデンマークの首都であり，童話作家アンゼルセンが数多くの物語を書き残した場所として知られている．この治安のよい整備された北欧一の都市は人口わずか 150 万人ほどで意外に小さく，初夏の日差しの中，一日も散歩すれば中心部全体を十分歩き回れるといった感じであった．今年は初めての試みとして，FPCA に連続して，同じ ACM 主宰で，6 月 12 日に国際ワークショップ SIPL'93 (State in Programming Languages)，及び 6 月 14 日から 16 日まで国際シンポジウム PEPM'93 (Symposium on Partial Evaluation and Semantics Based Program Manipulation) が開催されたため，筆者のように全日参加した人達は，毎日朝バスに乗って会場であるコペンハーゲン大学に向かい，夕方すぎにまたバスに乗りホテルに戻ってくるという規則正しくハードな 1 週間を過ごした．FPCA が提供したホテルは遊園地チボリや有名なショッピング街ストロイエなどのあるコペンハーゲン中心部に位置していたため，会議のあと日がなかなか沈まない街を出歩く余裕のある人にとっては，都合がよかったと思われる．しかし会議が 1 週間も続くと街に対する目新しさも減り，疲労もあわさって最後は観光といった雰囲気は全体的に感じられなかったように思う．

会議は Arvind (MIT) がプログラム委員長を，John Williams (IBM) が会議委員長を務めた．Arvind 以外のプログラム委員は Alex Aiken (IBM), Andrew Appel (Princeton), John Darlington (Imperial), Matthias Felleisen (Rice), John Hughes (Chalmers), Jean-Jacques Levy (INRIA), Harry Mairson (Brandeis DEC), Rishiyur Nikhil (DEC), Simon Peyton Jones (Glasgow), Rinus Plasmeijer (Nijmegen), Guy Steele (Thinking Machines), Mads Tofte (Copenhagen) の 12 人であった．今年の FPCA では 18 の異なった国，及び 95 の異なった機関 (うち 11 は企業) から，144 件の多方面にわたる論文が投稿され，その中から 33 件の論文が選ばれた．プログラミング委員が (共) 著者である論文は，他の 30 件の論文が決定したのち対象となる論文の著者が席を外す中で合否が決定されたと報告されている．このことは厳正な話し合いがプログラム委員会で行なわれたことを示している．参加者は日本人 6 人を含め約 180 人で，様々な関数型計算の分野から幅広く人が集まったばかりではなく，中には筆者のように，普段違う分野の研究を行なっているが，会議を通して関数型言語の分野を勉強しようという人も見受けられた．

Nobuko Yoshida, 慶應義塾大学計算機科学科
Keio University Department of Computer Science.
コンピュータソフトウェア, Vol.11, No.1(1994), pp.54–58.
1993 年 7 月 22 日受付.

一般発表

FPCA では 3 日間にわたり 33 件の発表が行なわれた．1 件の発表時間は質疑応答を含め 30 分であった．時間の関係のせいかチュートリアルもビジネスミーティングもなく，まさに一般発表中心の会議であった．筆者がこの一般発表を通じて学び取りたいと思ったことは主に 2 つある．1 つは，最近どのような分野の研究が関数型言語にとって主流になっているのかということと，2 つめは関数型パラダイムの枠組みで理論的・歴史的に確立している方法では取り扱えないような新しい問題に，最先端の研究者達がどのような姿勢で取り組んでいるのか知ることであった．特に筆者は普段並行計算に興味を持ち研究を行なっているので，広い意味で関数型の枠組みに並行性を融合する手法について学びたいと考えた．多岐にわたる全ての論文の内容について述べることは不可能なので，参考のため全論文のタイトルとその著者を本稿の最後に掲載するにとどめ，特に印象に残った研究の幾つかについて簡単に紹介していきたい．

まず，はじめの Applications: Experience のセッションでは，関数型言語を実際にアプリケーション構築の道具として使用した経験に基づく論文発表がなされた．[1] は，Amoco 株式会社で並列コンピュータを使用して科学技術計算を行なうプロジェクトを通して得た経験をもとにして書かれた論文である．石油タンクの状態変化のシミュレーション計算を行なう際，科学技術計算で従来用いられている言語 Fortran で直接プログラミングすると一万行〜十万行にも及び，プログラム全体を把握することが不可能になってしまう．一方，状態方程式はもともと関数の等式の集まりであるから，それをありのまま表現できる関数型言語 (ここでは Miranda) でプログラミングを行ない，目標とする並行計算機のコードに翻訳をすれば，複雑な計算もより自然に実行できる．ようするに関数型言語を「科学技術計算の方程式の仕様」として用いるという新しい発想の論文で，関数型ソフトウェアを作る人，関数型言語から並列計算機への一般的な翻訳系を作っている人，また並列計算機のアプリケーションを作っている人などにヒントとなる研究であった．[3] の Canon 株式会社の小里と Otto の研究では，画像処理のイメージプロセッシング計算を遅延関数型言語でプログラミングし，評価・考察を行なったものである．C 言語と Miranda，Haskell を使用し評価を行なった結果，一般のプログラマーレベルでプログラミングを行なうのであれば，やはり速さでは圧倒的に C 言語が勝るが，プログラムの書きやすさの面からみるとデータを型付きで取り扱える関数型言語が極めて優れている．よってユーザーの立場から見ると速い機械コードを生成するコンパイラーの作成が望まれるという主旨の発表であった．[1]，[3] 双方ともプログラムが公開されており，ベンチマークとして使用することができる．これらの研究はアプリケーションのアイデア自身が現実的に優れていたばかりではなく，関数型言語を使用する優位な点は何かを一般的な視点から改めて我々に気付かせてくれる点で，興味深い発表であった．

今回，特に発表数が多かった話題は「型」に関する研究であろう．これは理論面からと実装面からの発表が各々3 件ずつ行なわれた．筆者は特に型の専門家ではないので，この 6 件の中で特に面白かった型理論に関する [4] の研究について紹介する．この論文は型表現上の包含制約関係を解く推論アルゴリズムを提案したものである．関数型言語に関する型推論アルゴリズムで，最もよく使用されているものとして Hindley/Milner 型システムがある．よく知られているように型付け可能性は，型表現に対する等式の解があるかどうか (いわゆる単一化アルゴリズムを用いる) を調べることによりなされ，解があれば型付け可能であり，実行時のエラーが防げることになる．この等式の代わりに，より一般化した $X \subseteq Y$ という包含関係を解く型推論アルゴリズムを考えることで，従来型付け不可能であった表現にも型付けを行なうことができる．コンスタントを含む λ-項上の型を取り扱える彼らのシステムでは，全てのピュアな λ-項が計算可能な型を持つだけでなく，Hindley/Milner 型で型付き可能な項に関して，従来のシステムで推論した場合と同じ Hindley/Milner 型を割り当てることができ，その意味で従来のシステムの conservative な拡張になっていることが示された．発表では将来的にはこのシステムを関数型言語 FL に実装し，性能をみるつもりであると述べていた．この他，型に関する論文としては

[5] の subtype のための型システムから，conjunctive type のための型システムの翻訳を取り扱ったものや，型理論ではあるが前の 2 つの発表よりも具体的な関数型言語の型に関する研究で，関数型言語 Haskell の型階層をより一般化した型階層を持っている型付き言語に，オーバーローディングと高階多相型の概念を融合した型システムを提案した [6] があった．また，実装面では，[7] の Haskell のオーバーローデイングの速さの改善を図ったものや，ML のような静的に型検査を行なう多相型の言語に対して，コンパイル時に (ランタイムに必要な) 型情報を再構成する方法を示した [8] の発表があった．

並行性に関する研究としては，まず [13] の call/cc を含むλ-計算の意味論を取り扱ったものがある．call/cc オペレータを用いることで，継続 (continuation) を第一級オブジェクトとして取り扱える言語として Scheme や New Jersey の Standard ML があるが，このオペレータを含んだプログラムを並行評価した時の意味論を厳密に議論することは極めて困難であった．この論文はλ-計算に call/cc を加えた十分一般性のある形式系を考え，call/cc を含むことで不必要な逐次性を導入する必要がないことを観測等価性を使って述べたものである．観測等価性は，文脈の中で項 あるいはプログラムの動作を試行する概念を抽象したものであり，並行プロセス計算でもよく使用されることから，これから並行性を含んだ関数型言語の意味論を取り扱う上でも有用になると思われる．また，[14] の研究は従来の関数型言語の枠組みでは捉えにくかった遅延評価を行なう関数型言語の I/O-マシンの意味論を，Milner のプロセス計算 CCS で導入したラベル付き遷移系の方法を基礎とし与えたものである．ラベル付き遷移系とは，一般に $P \xrightarrow{\alpha} Q$ のように示され，これは「P が α というポートを通して外部と相互作用を行ない，その結果 Q に変化した」という相互作用を表現する．この論文では $P \xrightarrow{n} Q$ と書くと「プログラム P が文字 n をキーボードから入力しプログラム Q になった」という状態を，$P \xrightarrow{\overline{n}} Q$ と書くと「文字 n をプリンターから出力した」という状態を示していることとし，3 種類の異なった I/O-マシンにそれぞれ遷移則を与え，入出力の簡潔な定義を行なった．主な結果はこの 3 種類の I/O-マシンの表現能力が等しいことを，プロセス計算で一般的な相互模倣性 (bisimulation) という遷移則を基礎として導出される等式関係によって示したことである．この論文の意義は，並行プロセス計算の基本概念を関数型言語の枠組みに導入したことにあるが，並行プロセス計算の核である並行合成の演算子や，簡約に対応する内部遷移が定義されておらず，プロセス計算の概念を関数型言語に融合することの意味を十分に示したとはいえないのではないかと思った．

Compiler Optimization のセッションでは，まず Gill, Launchbury, Peyton Jones の共同研究 [22] の発表が行なわれた．関数型言語の書きやすさは，小さなプログラムを自由に組合せ，新しい関数を構築できる能力からきているといえる．しかし，このようなプログラムは一般的に多くの中間リストを生成してしまい，それがしばしば資源の問題につながってしまう．この研究は，単純で局所的なプログラム翻訳によって中間リストの生成を抑制するコンパイラーを提案し，その正しさの証明を行なったものである．この方法論はすでに Glasgow 大学の Haskell のコンパイラー上に実装されていることからも，極めて完成度の高い研究であったといえる．同セッションの筆者の研究 [24] は共有環境を用いた弱λ-計算の研究であり，λ-計算の β-簡約にあたる演算を項の複製と名前通信の 2 つの計算プリミティブに分解することによって，項の共有の単純な表現を可能にしたものである．この論文の主定理は，今までの環境を用いたλ-計算では実現されなかった Lévy の定義による 1 回簡約による「最適簡約」を最左戦略で実現できることである．

この他にも，アーキテクチャ部門ではデータフローに基づく実行系に関するもの [16][17] や，G-マシーンを素材に最適化理論の実マシーン上での性能をシステマティックに分析した [32] や関数型言語で表現した実時間 Deques [28] など，様々な分野からユニークな研究が寄せられ，関数型言語の分野がこのように広いとは筆者にとっては驚きであった．これは Partial Evaluation に関する研究が多かった前回の FPCA'91 とは対照的であったといえるだろう．特に今回は企業からの発表も多く，アーキテクチャ部門とアプリケーション部門に充実が感じられ，「使える関数型計算」を目指した研究が地

道に進められていることを知った．関数型言語としてはHaskell に関する論文が最も多く，ML などの破壊的代入を持った純粋でない関数型言語よりも，純粋な遅延型関数型言語の研究が主流を占めていた．筆者が期待した並行性に関しては，遅延型言語の意味を破壊することなく並行性を用いるという方向が主流で，プロセス計算的な意味での相互作用に基づく並行性を本質的に考える研究は意味論分野や言語レベルには見当たらなかった．しかしながら，並行性に限らず新しい概念を積極的に取り入れ，関数型計算の概念をより一般的に拡張しようとする雰囲気が会議全体にみられた.「関数」という共通語の中でどのような研究が発表されるのか，再来年の会議に期待したい.

最後に

以上，1993 年度 FPCA について一般発表の内容を中心に印象・感想などを簡単に紹介した．なお論文集はACM 出版より入手可能である (ACM Order Number: 407930)．また，次回の FPCA'95 は，1995 年カナダの Banff で開催される予定である.

発表論文

[1] *Experience with a Large Scientific Application in a Functional Language*. Rex L. Page (Tulsa Research Center, Amoco Production Company)

[2] *Some Issues in a Functional Implementation of a Finite Element Algorithm*. P.W. Grant, J.A. Sharp, M.F. Webster and X. Zhang (University of Wales)

[3] *Benchmarking Real-Life Image Processing Programs in Lazy Functional Languages*. Y. Kozato and G.P. Otto (Canon Research Centre Europe, Ltd.)

[4] *Type Inclusion Constraints and Type Inference*. Alexander Aiken and Edward L. Wimmers (IBM Almaden Research Center)

[5] *Conjunctive Subtyping*. D.J. Lillie (Imperial College of Science, Technology and Medicine)

[6] *A System of Constructor Classes: Overloading and Implicit Higher-Order Polymorphism*. Mark P. Jones (Yale University)

[7] *Implementing Haskell Overloading*. Lennart Augustsson (Chalmers University of Technology)

[8] *Compiler-directed Type Reconstruction for Polymorphic Languages*. Shail Aditya and Alejandro Caro (Laboratory for Computer Science, MIT)

[9] *Precedences for Cnoctypes*. Annika Aasa (Chalmers University of Technology)

[10] *Experiences with Compiler-Directed Storage Reclamation*. James Hicks (Motorola Cambridge Research Center)

[11] *Generational Garbage Collection for Haskell*. Patrick M. Sansom and Simon L. Peyton Jones (University of Glasgow)

[12] *Fixing Some Space Leaks without a Garbage Collector*. Jan Sparud (Chalmers University of Technology)

[13] *Sound Rules for Parallel Evaluation of a Functional Language with call/cc*. Luc Moreau and Daniel Ribbens (U. of Liege Sart-Tilman)

[14] *An Operational Semantics for I/O in a Lazy Functional Language*. Andrew D. Gordon (Chalmers University of Technology)

[15] *Delimiting the Scope of Effects*. Jon G. Riecke (Bell Laboratories, AT&T)

[16] *Generation and Quantitative Evaluation of Dataflow Clusters*. Lucas Roh, Walid A. Najjar and A.P. Wim Böhm (Colorado State University)

[17] *Efficient Implementation of Sequential Loops in Dataflow Computation*. Boon S. Ang (Laboratory for Computer Science, MIT)

[18] *Polling Efficiently on Stock Hardware*. Marc Feeley (University of Montreal)

[19] *Widening the Representation Bottleneck: A Functional Implementation of Relational Programming*. Dave Cattrall and Colin Runciman (University of York)

[20] *Towards Lazy Evaluation, Sharing and Nondeterminism in Resolution Based Functional Logic Languages*. Feixiong Liu (Oldenburg Universität)

[21] *On the Relation between Functional and Data Parallel Programming Languages*. Per Hammarlund and Björn Lisper (Royal Institute of Technology)

[22] *A Short Cut to Deforestation*. Andrew Gill, John Launchbury and Simon L. Peyton Jones (University of Glasgow)

[23] *A Fold for All Seasons*. Tim Sheard and Leonidas Fegaras (Oregon Graduate Institute of Science & Technology)

[24] *Optimal Reduction in Weak-λ-calculus with Shared Environments*. Nobuko Yoshida (Keio University)

[25] *Strictness Analysis using Abstract Reduction*. Eric Nocker (University of Nijmejgen)

[26] *Order-of-evaluation Analysis for Destructive Updates in Strict Functional Languages with Flat Aggregates*. A.V.S. Sastry, William Clinger and Zena Ariola (University of Oregon)

[27] *From Operational Definitions to Abstract Semantics*. S. Purushothaman (North Carolina State University) and Jill Seaman (Pennsylvania State University)

[28] *Real-Time Deques, Multihead Turing Machines, and Purely Functional Programming*. Tyng-Ruey Chuang and Benjamin Goldberg (New York University)

[29] *Parallel Implementation of Bags.* Herbert Kuchen and Katia Gladitz (RWTH Aachen)

[30] *Compiling Actions by Partial Evaluation.* Anders Bondorf (DIKU) and Jens Palsberg (Aarhus University)

[31] *FUDGETS: A Graphical User Interface in a Lazy Functional Language.* Magnus Carlsson and Thomas Hallgren (Chalmers)

[32] *Assessing the Evaluation Transformer Model of Reduction on the Spineless G-machine.* Sigbjorn Finne and Geoffrey Burn (Imperial College of Sc., Techn., Med.)

[33] *Benchmarking Implementations of Lazy Functional Languages.* Pieter H. Hartel and Koen G. Langendoen (Univ. of Amsterdam)