ロジック・プログラミング シンポジウムに参加して

中島秀之
（電子技術総合研究所）

はじめに

1984 International Symposium on Logic Programmingは IEEEの主催で米国アトランティックシティー（後覧的市）で1984年 2月 6-9日に開催された。このうち、6日はチュートリアルで、 7-9の 3日間が会議に当てられた。

アトランティックシティーは最初、ニューヨークからのリゾート地として栄えていたそうである。それが航空機の発達に伴い、人々が上空を飛び越してフロリダまで足を伸ばすようになってから寂れ、10年前にはゴーストタウンのようだったそうである。それが最近賭博で盛り返し、売上げでラスベガスを抜いたとかいう話である。そういうわけで海岸沿いにカジノが立ち並ぶという何とも奇妙な風景の町である。「初めてアメリカを訪れる人達に、これが典型的アメリカだと思われると困る［筆者訳］」とは、ニューヨーク在住の一女性の弁である。シンポジウムの会場もカジノのあるホテルであった。参加者のうち何人が勝ち、何人が負けたのかは定かでない。ちなみに、筆者は当然にも前者である。

主催者側の話では、最初 125人くらいの参加者を見込んでいたのが、実際には 350人以上が集まったということである。正確な比率はわからないが大学関係の研究者の他にソフトウェア企業からの参加が目立った（半数以上？）。ディナーの席上でIBM のワトソン研究所・長が日本の第五世代プロジェクトは侮るべきではない、との話をしたが、出席者一同の反応は、何をいまさらという感じであった。この反応の分析はむつかしいが、筆者が考えるに、彼の話は日本嫌いの企業向けの話であって、研究者には向かなかったのではないか。いずれにしても、日本人研究者が聞いて有益な話ではなかった。

以下に、発表の概略を主観的に紹介させていただく。発表内容の面白さと紹介行数をほぼ比例させようと試みたのであるが、単語の長さなどに差があって、実際には誤差のほうが大きくなってしまったようである。

1.アーキテクチャ

アーキテクチャはふたつのセッションに分かれ、全部で７件の発表があり、Prologの並列実行に関するものが大部分を占めた。

Borgwardt(Univ. of Minesota)は共有メモリ型マルチプロセッサ上でのPrologの並列実行について述べた。これは現在の、一台のプロセッサで行なう逐次実行の拡張になっている。

D.S.Warren, Ahamad, Debray, Kale (SUNY* at Stony Brook) はPrologをデータベス質問用言語とみなし、そのOR並列実行をブロードキャスト・ネットワーク上で実現する話を述べた。

中川（横国大）は述語のアサーションの一組ごとにユニフィケーション用のプロセッサを用意したシステム上でのPrologの AND並列実行について述べた。

Tick, D.H.D.Warren(SRI) はPrologのパイプライン処理について述べ、100ns のサイクルタイムを仮定して、450 KLISP の実現が可能であると述べた。ちなみに現在のDEC-10 Prolog のコンパイルド・コードのスピードはDEC-2060で40 KLIPSである。アーキテクチャのセッション中では、この話が一番現実性が高く感じられた。［た

＊State University of New York

だし、筆者はアーキテクチャに関しては全くの素人である。]

田村、金田（神戸大）はマルチ・プロセッサ上でのPrologのOR並列（パイプライン）実行について述べた。この方式では並列度は３くらいまでしか上がらないとのことである。

Ciepielewski, Haradi(Royal Institute of Technology, Sweden)は限られた数のプロセス上でのOR並列をトークンで制御する方法を述べた。

Taylor, Lowry, Maguire, Stolfo(Columbia Univ.)はDADOと呼ばれる木状の巨大並列アーキテクチャ（現在1023プロセッサのプロトタイプを製作中）上でのPrologのAND/OR並列実行について述べた。ここでは全ての解（変数の可能な値）の集合を操作する方法をとっている。

2.並列言語

こちらはPrologの並列化を言語面から見たセッションである。３件の内、２件はConcurrent Prolog に関するものである。

Khabaza(Univ. of Sussex, England) はnegation as failure による、否定情報を含む解の集合を操作するロジック・プログラミング言語について述べた。

Gelernter(Yale Univ.) はConcurrent Prolog によるシステム記述を、C やLinda によるものと比較し、Concurrent Prologは強力であるが、データの流れと字面上の前後関係が一致しないので理解しにくいと述べた。これに対し、会場からPrologプログラムをデータの流れとして理解する読み方が間違っているとの声があった。

当のShapiro (The Weizmann Institute of Science, Israel) は予稿集の内容（Concurrent Prolog による公正なマージ）ではなく、Bagel と呼ばれるアーキテクチャについて述べた。彼は、データ構造をコンパイラが決めるのは無理なのと同様、プロセスの構造決定（プロセスのプロセッサへの割当て）をコンパイラに求めるのは無理であるから、プロセスの構造を記述する言語をConcurrent Prologに追加すべきであるとし、それを実現するアーキテクチャを示した。

3.アプリケーション

このセッションでの発表は２件で、いづれも大規模アプリケーションの話ではなかった。時期尚早か？

Aponte, Fernandez, Roussel(Universidad Simon Bolivar, Venezuela) は人間が定理証明を行なうのをサポートするシステムを示した。

Hellerstein(Harvard Univ.)はConcurrent Prolog で最大フローの問題を、破壊的代入なしに効率良く解く方法を示した。

4.知識表現とデータベース

別名、日本人セッション［チェアマン、Shapiro ］。

宮地、国藤、北上、古川、竹内、横田(ICOT)はデータベースに新しい知識を取り込む方法に関して述べた。これは、Bowen とKowalskiのdemo述語を基礎として、新しい知識の証明可能性、矛盾性、冗長性、独立性をチェックする方法である。

中島（電総研）はProlog/KR の多重世界機能を用いた、概念の階層構造およびそれらの間の属性の受継ぎの実現法について述べた。この方法では多重受継ぎも操作することができる。

北上、国藤、宮地、古川(ICOT)は知識ベースにおける、知識獲得システムの構成と実現法とについて述べた。知識獲得システムには、メタ推論(demo)・演えき(de- duce) 、帰納、知識の同化、知識の調整の機能が要求される。

5.ロジック・プログラミングと関数型プログラミング

この分野の研究も盛んで、２セッション６件の発表があった。ただし、中にはLispによるPrologのインプリメンテーションについて述べたものも含まれている。

Subrahmanyam, You(Univ. of Utah)はセマンティック・ユニフィケーションとして、関数のリダクションを許す言語 Funlog について述べた。

Carlsson(UPMAIL,＊ Sweden)はLispでPrologをインプリメントする方法について述べた。

Barbuti, Bellia, Levi(Universita di Pisa), Martelli(CNUCE - C.N.R., Pisa)は宣言的部分と手続き的部分よりなり、両者のインターフェースを持つ言語について述べた。

Lindstrom, Panangaden(Univ. of Utah)らはストリームを基本とするホーン節の実行モデルについて述べたようだ（筆者は居眠りしていたらしく、記憶にない。以下、「らしい」を使う場合はこれに準ずる）。

Smith(Univ. of North Carolina)はBackusの関数型言語を基本にした FFPマシン上でのロジック・プログラミング言語のAND/OR並列実行について述べた。

Reddy (Univ. of Utah) は不定変数を含まないロジック・プログラムの関数型プログラムへの変換法を示した。

6.インプリメンテーション

Prologのインプリメンテーションに関するセッションであるが、そうでないものも含まれていた。

D.S.Warren(SUNY)はPrologの実行モードを深さ優先に限定しない場合の、効率の良いメモリ管理法について述べた。

Wise, Powers(Univ. of New South Wales, Australia)はPrologの節のインデキシングについて述べた。これは各節をバイナリーの語に落とし、それらのlogical and により粗いマッチングを取る方法である。

Stickel (SRI) はPrologに近い速さを持った、（制限付き）定理証明システムのデザインについて述べた。

7.文法と構文解析

Prologの得意とする構文解析に関するセッションである。

上原、落谷、角所、豊田（阪大）は述語論理を基本にしたボトム・アップパーサPAMPS について述べた。

Porto, Filgueiras(Universidade Nova de Lisboa)はロジック・プログラミングを基本とした、自然言語のセマンティクス表現言語を示した。この言語はPrologプログラムに翻訳可能である。

Abramson(Univ. of British Columbia, Canada) は Definite Clause Transition Grammarと呼ばれるDCG の拡張版を示した。これは属性文法のロジック・プログラミングによる実現とも考えられる。

8.ロジック・プログラミング言語

Aspects of Logic Programming Languages と名付けられたセッションであるが、その実miscellaneous の感がある。

Kahn(UPMAIL)はlazy bagと呼ばれる、解の集合を遅延実行によって求めるプリミティブの紹介と、それ一つで not を含むPrologの制御構造のほとんどがつくれることを示した。

Brough, Emden(Univ. of Waterloo, Canada)はPrologのプログラムをデータフロー、フローチャート、LUCID的に考えることにより、効率の良いプログラミングが可能になることを示した。

＊ Uppsala Programming Methodology and Artificial Intelligence Lab.

玉木（茨城大）はPrologにreducibilityという概念を追加することにより、関数型言語を取り込む方法を示した。

Zaniolo (Bell Laboratories) はPrologでオブジェクト主導型言語のインタプリタを書いた。

9.理論

筆者には理論コンプレックスがあり、理論をやるひとはインプリメンタより偉いと思い込んでいるのだが、現実にはそうでもない場合もあるようである。

Plaisted(Univ. of Illinois) はPrologのオカーチェックは必要であるが、コストが高いため、オカーチェックの不要な場合を分析し、ごく限られた状況でのみ実施すれば十分であることを示した。（発表は最後になった［単に著者のアリバイを示すための記述］。）

Jones(Datalogisk Institut, Denmark), Mycroft (Edinburgh Univ., England)はPrologのオペレーショナル／デノテーショナルセマンティクスをステップワイズに開発する手法を述べた。

Mishra(Univ. of Utah) はPrologプログラムのタイプ付けに関して述べた。引数のタイプは、それ以外の引数ではプログラムが成功しないような引数の集合として定義されている。（この定義では全体集合でかまわないのでは［筆者］？）

Gergely, Szots(Research Institute of Applied Computer Science, Hungary) は1階古典述語論理のサブセットをロジック・プログラミング言語として示したらしい。

まとめ

アーキテクチャでは猫もしゃくしも並列化という風潮が見受けられた。Prologの並列化は一見簡単そうで実は困難な問題であると思う。重要な課題であるだけに、もう少し基本からの見直しが期待される。Shapiro のアプローチが一番手堅いところか。

知識表現関係では日本からしか発表がなかったのは意外である。少なくとも米国のソフトウェア会社は知識工学への応用に関心を持っていた。ICOTのアプローチは面白いのだが、Prologでメタ推論システムを作ればそれらのことができるのは、なかば当然であり、彼等の研究成果が今後の言語／アーキテクチャ仕様にどう影響を及ぼすのかのほうに注目すべきであろう。

関数型言語との融合に関しても、まだこれといった案が見られない。二つの言語を含み、そのインターフェイスを用意するというアプローチは筆者の感覚にそぐわない。最初から一つの枠組みで統一する玉木のアプローチが興味を引いた。

1981年のハンガリーのワークショップで見られたような、新しい言語の提案が減っている。そろそろ安定期に入ったのか。ただし、Prologを今のまま受け入れてしまうのも考えもので、現存の言語の改良は必要である。

文法と構文解析の記述に関しては、ロジック・プログラミングが有利であるのはDCG の例でも示されている。後は、効率に関してどこまで行けるかだが。

Prologの制御構造に関しては、もう少し見直しが必要であると感じる。Kahnのアプローチが成功するかが興味のあるところである。

謝辞

1984ロジック・プログラミングシンポジウムに参加の機会を与えてくださった柏木部長に感謝する。また、本報告の初期の版の不足を指摘していただいた諏訪室長を始めとするETL の方々、情報処理学会IC-2の方々に感謝する。