Vol. 37 No. 5　　情　報　処　理　　May 1996

解　説　第五世代コンピュータプロジェクトの成果と残された課題

2. ICOTにおけるプログラム言語とその実装に関するコメント
—偉大なる先人に敬意を込めて†—

松　岡　　聡††

1. はじめに

ICOTの数々の研究分野の中で，論理型プログラム言語とその実装系，およびそれらの理論的研究は1つの中核をなす．その「功罪」に関して事後評価するのはたやすい．特に「論理型」という言語族の特殊性から，そのような評価はややもすると言語パラダイム間の安易な評価や比較になりがちである（これは，ICOTに関する海外の新聞記事にも見受けられた傾向である）．しかし，プログラム言語のデザインや実装系の研究は，一朝一夕でなるものではなく，膨大な試行錯誤や技術の蓄積が必要であり，単なるパラダイム論ではそれらの価値ある経験が無視されてしまう．むしろ，現状の言語のデザイン技術や，実装技術から見て，ICOTの言語研究の新規性やその問題点を，限られた誌面の中で客観的に評価するように試みてみたいと思う．

2. GHC-KL1の言語開発とそのデザイン

ICOTでは，Prolog，GHC（Guarded Horn Clauses，並列論理型言語），KL 1（Kernel Language 1，GHCを発展させた一般プログラミング用並列論理型言語），Quixote（データベース言語），MGTP（定理証明言語），GDCC，CAL（制約解消言語）などの数々のプログラム言語が，研究用途に応じてデザイン・実装された．その中でも，いわゆる並行論理型でDependent-AND並列を主体とするCommitted Choice Language（CCL）であるGHC－KL 1が中核言語となり，実行ハードウェアであるMulti-PSIおよびPIM（Parallel Inference Machine）の上で直接実行され，OSであるPIMOSの記述や，遺伝子応用や法例推論などの数々のアプリケーションの実装，およびその他の言語の実装の礎となった．

ここではまず，「KL 1のようなCCLをベースとする選択」が，言語および実装上のfeasibilityとしてどうであるか，という点を考える．GHCはPrologと同様に，ホーン節の集合をプログラムとする一階の述語論理をベースとした並行論理型言語であるが，特徴としては(1)並行実行のためにCommitted ChoiceのDependent-AND並列実行モデルを採用したことと，(2)同期のために節の冒頭にGuard部分を設け，ゴールとの単一化の際にGuard部からゴールの論理変数への具体化を禁止することによって，ゴール間の同期・通信を達成することにある．このきわめて単純な機構により，汎用の並行プログラム言語として，十分な柔軟性を発揮し，多くの並行プログラムが容易に記述されることが上田や竹内らによって示された[1]．(GHCによるプログラム例は参考文献2）に詳しい．)

GHCはCCLであるがゆえに証明系としての完全性を失ったが，そもそも完全性を保存するのは，汎用言語としてはいくつかの技術的問題がある．たとえば，証明探索時のOR並列性または並列バックトラックは実装にとって複雑性を大幅に増し，効率の点で問題が生じる．また，ユーザプログラミングにとっても，OR並列性による非決定性は扱いにくい場合が多い．したがって，汎用プログラム言語としてはこのような選択が行われたのは妥当であるといえよう．後に，ICOTでは定理証明のための専用言語であるMGTPなどが開発されたが，それらはすべてKL 1の上で，あるいはKL 1にコンパイルされて実行される．

CCLの中でも，Concurrent Prolog，Parlogや

† A Commentary on Program Language Design and Implementation Research at ICOT—Honoring our Great Predecessors—by Satoshi MATSUOKA (The University of Tokyo).

†† 東京大学大学院工学系研究科情報工学専攻

Flengなどの「ライバル」が存在したが，GHCは言語として「ちょうど良い」デザインをしていたともいえよう．実際，ガード部とボディ部から分離する記号'|'は，Prologのcut演算子との類似性が高く，Parlogなどのように特殊なアノテーションが必要ない．また，Guard部に節全体の起動条件を書き，論理変数をストリームとしてゴール間通信を行うというプログラミングスタイルにより，比較的見通しの良い並行プログラムが記述できるといわれている*．（並行論理型一般のサーベイは参考文献3）を見られたい．）

GHCは理論的な観点からは「美しい」言語であるが，実用的な並行言語としては欠ける機能があった．それらを「荘園」と呼ばれる一種のメタ機能を中心として補って拡張したのがKL 1である．荘園のメタ機能を用いることにより，ユーザはスケジューリングやロードバランシングなどの様々な並列計算の資源制御を行うことができる．

しかし，KL 1は，近代的なプログラム言語としていくつか欠ける点があるのも事実である．

- ●静的な型システムがない．
- ●データ構造，特に抽象データ型やオブジェクトのサポートが弱い．
- ●数値計算に不可欠な配列の扱いが煩雑．
- ●構造化プログラミングのサポート，特にモジュール化機能が欠けている．
- ●他言語インタフェースが弱い．

これらのうち，いくつかは後日のModed Flat GHCや次に述べるKLICにより解決が模索されている．また，ICOTでは数万行にも及ぶアプリケーションが実際に開発されていることを考慮すると，その開発規模のレベルの実用的な言語としてKL 1はかなり良好な性質を示していたといえよう．しかし，より広く流布するには，ソフトウェア工学上，大規模プログラムを作成するのに有効な機構を取り入れるのみならず，プログラム設計法の確立や様々なツールを開発してゆく必要があろう．（この点は後に述べる．）

* 幾人かの熟練KL 1プログラマらの証言によると，KL 1のプログラムでは同期によるバグはほとんど生じないそうである．

3. 実装技術に対する評価—PIM vs. KLIC, 他の言語

ICOTはKL 1を直接実行するハードウェアとして，専用マシンであるPIM（Parallel Inference Machine）を開発した．確かにPIM上のKL 1は一定の性能は得られ，ICOT内のKL 1上の並列処理の研究に大いに貢献したが，KL 1自身の普及に大いに貢献したとはいいがたい．現在では専用マシンはRISCプロセッサの性能向上に追従するのは困難であると見なされており，また，その上で開発されたソフトウェアの可搬性・携帯性にも欠ける．

専用マシン上でのKL 1の一種のアンチテーゼとして開発されたのがKLIC（KL 1 in C）である．KLICは，(1)UNIXなどの通常のOSの上で動作する，(2)コンパイラのターゲットとしてC言語を採用する，(3)分散並列実行や，入出力などの機種依存性を吸収するためのGeneric Object機能を有する，(4)Cの文がKL 1中に埋め込める，などの特徴を有し，KL 1ユーザプログラムのRISCワークステーションやMPP（超並列計算機）への可搬性を達成した．KL 1は現在でも改良が続けられており，多くの場合PIM上のKL 1に同等か，それ以上の実行性能を示す．

しかし，最初からいきなりKLICの開発が可能であったかは疑わしい．並列オブジェクト指向言語や，並列関数型などの他の言語パラダイムにおいても，細粒度～中粒度並列をRISC上に実装する技術が確立されたのは80年代末からである．実際，PIM上のKL 1のランタイムシステムはソフトウェア上の技術的レベルが大変高く，特に数百ノード規模で分散したメモリ管理・単一化・ガーベジコレクション，プロセス（ゴール）移送・負荷分散・停止判定，などの基本的技術は，PIM上のKL 1で多くの基本方式が開発され，KLICにそのまま，あるいは改良されて受け継がれている．

逆に，PIM-KL 1やKLICの大きな弱点は，静的解析系の弱さである．確かに上田氏によるMode解析や，GHCの型解析などの技術は提案こそされてはいるが，実際のコンパイラには組み込まれていない（近山氏談）．また，並列計算用

の種々の解析，たとえばIdなどで行われているスレッド融合や，最適なゴールのマッピングなども行われていない．世界的には，論理型言語の静的解析はいくつか行われており，特に最近ではAKLの処理系に多くの成果が集約されているようである．

いずれにしろ，KLICの開発により可搬性の確保は達成された；今後はKLICをベースにした様々な静的・動的最適化技法の開発が望まれる．特に，現段階ではメッセージ送信の速い商用のMPPでさえ，分散メモリ版の並列プログラムの実行速度はPIMのものにはまだまだ及ばないのが現状である．これは，KLICが遠隔参照をすべてGeneric Object経由で行い，PVM (Parallel Virtual Machine)，MPI (Message Passing Interface) など細粒度通信には向かないメッセージ通信ライブラリを用いているのが一因である．これらをソフトウェア技術により個々の計算機に特化したものに変換できれば，大幅な速度向上が見込まれる．

4. 世界的貢献ーできたこと，できなかったこと

KL1に代表されるICOTのプログラム言語の研究開発は，論理型言語の領域に関しては世界的に大きな影響を与えたのは明らかである．しかし，並列言語や並列処理の世界に対しての影響は，残念ながら低いといわざるをえない．単に伝統的な数値計算の分野のみならず，非数値的な記号処理分野でも，ICOTの貢献が世界的に認知されているとはいいがたい．個々のKLIなどのプログラム言語の普及はいうに及ばず，論理型のパラダイムに限定されない他のパラダイムに適用可能なソフトウェア技術も，他の言語分野からICOTの研究がオリジネータと参照されることは少ないのが実状である．

これはなぜであろうか？ 研究の質自身が問題ではないとまず断言しよう．先に述べたとおりに，ICOTのランタイムの処理系のソフトウェア実装技術は当時としては世界的なレベルに達していた[4]．また，確かに数値計算応用には実数やその配列のサポートがあまりにも貧弱すぎたが，後期のICOTでは，回路シミュレーションや法例推論など，興味深い非数値応用が並列化されていた．

1つの原因は，ICOTにおけるプログラム言語研究が，あまりにも「論理型言語の研究」というパッケージに包まれており，その本質的な原理が他の言語に応用できたとしても，他の分野の言語研究者に認識されていないことにあろう．たとえば，KL1はゴールの分散停止判定に関してはかなりの汎用的な成果をあげており，これは他の言語の分散プロセスの停止判定に直接応用可能なのだが，国際的な認知度が低いのが実状である．

また関連するが，KL1の実装自身，PIMに特化された形で外部に発表されることが多く，他の言語にも応用可能な汎用性のあるソフトウェア技術であるか否か判読が難しかった，という原因もあろう．RISCの登場により，80年代には世の中はすでに汎用アーキテクチャに向かって動き出しており，特定のハードウェアを前提とした言語研究は正当な評価を受けるのが困難な時代になってしまった．

この現象は，何もICOTに限られたものではない．Andorraに代表される並列論理型言語やその専用マシンの研究は一般的に実際の並列処理にインパクトを与えられなかった．これは，実行効率の低さ，実際の処理系の可搬性のなさなど様々な技術的要因に加え，論理型言語などの新しい言語やパラダイムへのアレルギーなどの社会的要因も考えられる．しかし，オブジェクト指向言語のようにほぼ同時期に研究が行われ，一般のプログラム言語として広く普及しているものもあるのも事実である．やはり科学技術計算やビジネス計算における並列アプリケーションプログラマは，問題を早く容易に解くことが計算機を使う一義的な目的であり，現在までにはまだ彼らを満足させるだけの言語やその処理系ができていないということが主な要因と考えられる．今後のKLIC上の研究は，論理型のフォーラム（のみ）に留まらず，たとえばIEEE Supercomputing Conferenceなどの並列処理一般のフォーラムに提示されていくべきであろう．

5. 将来に向けて：パラダイム主導からアプリケーション主導へ

プログラム言語は本来計算を表現する道具である．その道具に関する技術的蓄積は，コンピュー

ティングの最も基礎的でかつ重要な技術であるといえよう．ところが，我が国のプログラム言語研究，特に実装方面やOSに近い部分は世界的に遅れをとってきた．これは，我が国の近年のプログラム言語研究では比較的パラダイム主導型の理論的研究が重宝される一方，実装研究に対する妥当な工学的視点の評価が欠けていたのが1つの原因ではないか，と考える．

本来は，言語研究，特に実装の研究はパラダイム主導型ではなく，アプリケーション主導であるべきである．これは理論やパラダイムを否定するものではないが，それらは常にアプリケーションからの厳しい評価の目にさらされなくてはならない，ということである．その点からすると，理論と実装は密接な関連があるべきで，たとえばたいそうな理論があってもトリビアルな実装しか可能ではないのは意味がない．また，特定のパラダイムの有効性は，常に実現性とアプリケーションに対する潜在的な有効性を秘めていなくてはならない．研究結果はすぐに現実の言語処理系に応用できるものでは必ずしもないにせよ，将来においては現行の言語やその処理系に対して「新たに学ぶ」ハードルを越えるに足る魅力あるものに至ることがある程度明確ではなくてはならないのである．

オブジェクト指向が成功したのも，まさにこのようにアプリケーションの要求に応え得るパラダイムであったからで，今では逆にアプリケーションが主導となってオブジェクト指向自身が改良・発展されている段階である．まだまだ研究の続く並列論理型言語においても，「論理型」の錦の御旗を異議なしに受け入れるのではなく，常に他のパラダイムや言語に対する優越性をアプリケーションからの客観的な視点で自己評価し，場合によっては他のパラダイムの特質を取り入れる必要がある．たとえば，現在のKL1ではオブジェクト指向の継承機構がないが，ソフトウェア工学的観点からはその導入は有効であろう（実際，いくつかの提案はなされている）．

ICOTでは理論と実装，およびアプリケーションをより密接に結ぶ上で大いなる貢献をしたが，まだまだ十分ではない．むしろ，ICOTにより，初めてそのような（欧米では昔から行われてきた）本質的なプログラム言語の研究法が大規模に行われたといえ，我々の若い世代の研究者はその資産に基づいて世界と勝負していくべきであろう．

参 考 文 献

1) Fuchi, K. et al.: The Fifth Generation Project: Personal Perspectives—Launching the New Era, Communications of the ACM, Vol. 36-3 (Mar. 1993).
2) Tick, E.: Parallel Logic Programming, The MIT Press, pp. 49-99 (1991).
3) Shapiro, E.: The Family of Concurrent Logic Programming Languages, ACM Computing Surveys, Vol. 21-3 (Sep. 1989).
4) Rokusawa, K., Ichiyoshi, N., Chikayama, T. and Nakashima, H.: An Efficient Termination Detection and Abortion Algorithm for Distributed Processing Systems, Proceedings of the 17th International Conference on Parallel Processing, The Pennsylvania State University Press, pp. 18-22 (1988).

（平成7年11月1日受付）

松岡 聡（正会員）

1963年生．1986年東京大学理学部情報科学科卒業．1989年同大学院博士課程中退．同年同大学助手．現在，同工学部情報工学専攻講師．理学博士．オブジェクト指向言語，並列システム，リフレクティブ言語，制約言語，ユーザ・インタフェースソフトウェアなどの研究に従事．ACM，IEEE-CS各会員．ACM日本支部書記．