プログラミング言語 11－2
（１９８７．５．２２）

Lispの動向について

湯淺太一
（京都大学 数理解析研究所）

Lispの標準化とも関連して，近年注目を集めている３つのLisp方言

- 実質的な標準となりつつあるCommon Lisp
- 教育・研究用として広く利用されているScheme
- 標準案として設計がすすめられているEulisp

をとりあげ，その概要と最近の動向を述べる。

State-of-the Art of Lisp

by Taiichi Yuasa
Research Institute for Mathematical Sciences, Kyoto University
Kyoto, 606, Japan

Overviews three Lisp dialects:

- Common Lisp, which is being a defacto standard
- Scheme, which is widely used for educational and research purposes
- Eulisp, which is being designed as a standard

1. はじめに

Lisp は人工知能向けの言語として 1950 年代後半に考案され、Lisp 1.5 [1]の言語仕様に基づいた処理系が 60 年代初頭に公開されている。60 年代半ばには、言語仕様と処理方式を大幅に改良した BBN-Lisp [2] (現在の InterLisp [3,4]) と MacLisp [5,6] の 2 つの方言が登場し、その後の Lisp 言語・処理系の開発に大きな影響を与えた。以来、数多くの Lisp 言語・処理系が開発されている。また、Lisp 専用機、いわゆる Lisp マシーン [7,8] や Lisp 専用の LSI チップ [9] なども開発されている。

Lisp 言語は対話型の処理系に組み込まれており、プログラムを記述するための言語であると同時に、プログラム開発の各側面において処理系と対話するための言語でもある。処理系に関連する一般的特徴としては

- 関数間結合が動的に行われる。
- リスト処理機能や動的なメモリー管理機能が組み込まれている。
- プログラムを Lisp のデータとして扱うことができる。

などがある。Lisp 処理系にはプログラム開発環境が充実しているといわれるが、これには以上の特徴が寄与するところが大きい。その他には、

- 評価・実行すべき式を実行時に生成できる。
- 関数をデータとして取り扱える。
- 高度なマクロ機能を有する。
- 変数や関数には型を指定する必要がない。

などの一般的特徴を有している。

以下では、Lisp の標準化とも関連して近年注目を集めている 3 つの Lisp 方言、Common Lisp、Scheme、Eulisp について、その概要と最近の動向などを簡単に紹介する。

2. Common Lisp

Common Lisp は、もともと MacLisp 系 Lisp (ZetaLisp [10]、NIL [11]、Spice Lisp [12] など) の言語仕様統一を目指して設計された。設計作業は 1981 年末に始まり、当初は上記 MacLisp 系 Lisp のメンバーによって議論が進められた。後に InterLisp をはじめとする他のいくつかの Lisp のメンバーも加わり、Lisp の国際標準化案として注目されている。

設計の基本方針は 1982 年までに固まり、第 2 回 Lisp 会議でその概要が報告されている [13]。その後いくつかの試案 [14,15] を経て、1984 年には仕様書が出版されている[16]。

Common Lisp が標準化案として注目されている理由として、次のような特性をもっていることがあげられる。

- 多くの研究者、処理系作成者の協力のもとに設計された。

・既存の Lisp 処理系との互換性が高い。
・OS やハードウェアに依存しない仕様である。
・実行効率の高い処理系が実現可能である。
・仕様が比較的厳密に記述されている。

Common Lisp の言語仕様の大きな特徴としては次の 4 点があげられる。

・静的スコープの採用。
・Lisp の基本文法を与えるスペシャル・フォームの導入。
・強力な関数閉包による完璧な downward funarg と upward funarg の実現。
・キーワード・パラメータの導入による関数の汎用化

Common Lisp は、その設計段階からすでにいくつかの処理系開発が進められきた。それらの中には、処理系の移植性を重視したものもあり、Common Lisp の普及に貢献してきた。例えば、Common Lisp 処理系の1つである Spice Lisp のグループでは、CMU パッケージなるものを用意している。Lisp (Common Lisp) で書かれた Spice Lisp のソース・プログラムがはいっており、マイクロ・コードの部分を書くか、あるいはマイクロ・コードをシミュレートするプログラムさえ書けば移植できるように配慮されている [17,18]。Kyoto Common Lisp [19]は C 言語と Common Lisp で記述されていて、簡単な修正だけで移植ができるようになっている。さらに、ポータブル・コンパイラをベースにした Common Lisp 処理系の商品化も行われている。

すでにかなりの数の Common Lisp 処理系が実用に供している。Lisp の実用的なアプリケーションも、最近はその多くが Common Lisp で記述されている。特に、米国のビジネス・ベースのアプリケーションに関してはそのほとんどが Common Lisp で記述されているようである。このように、Common Lisp は実質上の国際標準となりつつある。米国においては、ANSI のサポートのもとに、X3J13 とよばれる Common Lisp 委員会が、ANSI 標準化に向けて活動を開始している [20]。

さきに、Common Lisp の仕様が「比較的」厳密だと述べた。これは、他の Lisp のマニュアル類と比較してのことであり、処理系が数多く現われるにつれて、その仕様のあいまいな点が指摘されはじめている。また、コンパイラの仕様など、はじめからほとんど明記されていない個所もある。標準化を進める上で、これらの問題点を解決していくことが必要である。

現在の仕様にはグラフィックス関係の機能や、プログラミング・ツールなどの、処理に強く依存する機能は含まれていないが、ウィンドウ・システムとのインターフェイスの検討が進められている。また、オブジェクト指向機能の導入も検討されており、Common Loops [21,22] とその後継である CLOS (Common Lisp Object System) [23]も提案されている。

3. Scheme

Scheme [24,25,26,27,28] は 1975 年ごろ MIT で設計された Lisp 方言である。言語そのものがコンパクトであることと、概念的にまとまった仕様であることなどから、教育・研究用として広く利用されている。特に、Scheme を記述用言語として使った Abelson と Sussman のプログラミングの教科書 [29] の出版以来、

ソフトウェア教育に大きく貢献している。

コンパクトな仕様のために、処理系の作成も他の Lisp と比較すると容易であり、数多くの処理系が存在する。一方、Scheme は標準化を意図して設計されたものでないこと、仕様がコンパクトなために、処理系ごとに固有の拡張が行われてきたことなどから、処理系間の互換性が良くないようである。最新の仕様書 [28] は、Scheme 処理系間の言語仕様統一を目指したものと思われる。ここでは、Scheme の基本構造を denotational semantics で記述するなど、仕様を明確に与えようという試みがなされている。

Scheme の特徴的な点として、次のものがあげられる。

- 静的スコープの採用。
- コンパイラだけでなく、インタープリタも末尾再帰性をもつ。
- 強力な continuation の機能をもつ。
- 変数と関数の名前空間が同一。
- ラムダ式が関数閉包を生成する。

特に、静的スコープの採用と関数閉包の概念は、その後の Common Lisp や Eulisp の設計に大きな影響を与えている。

Scheme は簡潔な言語ではあるが、continuation による制御機能は強力で、これによって、Common Lisp の非局所的飛び出し (non-local exit) 機能 (block と return-from、catch と throw など) が実現できし、コルーティンも実現できる。これは、教育・研究用としては非常に興味深い。その一方で、処理系の実行効率の問題がある。このために、米国では、Lisp の教育は Scheme で行い、実用的なアプリケーションは Common Lisp で記述するという傾向があるようである。

4. Eulisp

Eulisp は、Lisp の国際標準化案として、欧州の研究者たちが現在設計をすすめている Lisp 仕様である。設計作業が始まったばかりで詳細な仕様はまだ決まっていないが、基本的な設計方針や仕様の大筋が 1986 年の Lisp 会議で報告されている[30]。

Eulisp は次の三つのレベルから成っている。

- Level 0 (The Semantic Foundation)
- Level 1 (The Kernel)
- Level 2 (The Development Language)

level 0 は level 1 の、level 1 は level 2 のそれぞれサブセットである。即ち、level 0 で記述されたプログラムは level 1 の処理系でも level 2 の処理系でもそのまま動くし、level 1 で記述されたプログラムは level 2 の処理系でそのまま動く。

level 0 はプログラミング言語というより、level 1 と level 2 の言語仕様を記述するための仕様記述言語として使われる。すなわち、最も基本的な機能のみ

を level 0 に入れ、これを使って level 1 と level 2 を記述する。level 0 の仕様自体は denotational semantics を使って記述する。言語水準としては Scheme に近い。リストや記号、関数といった基本的なデータ型はすでに level 0 に含まれ、さらに上位レベルを記述するためにデータ型の追加機能も備わっている。Common Lisp との大きな相違に、変数の名前空間と関数の名前空間の区別がないことがあげられる。これは Scheme の影響を強く受けていることを意味する。

次の level 1 は、PSL (Portable Standard Lisp [31]) 程度の水準である。機能的には不足があるものの、実用的なアプリケーションが十分書ける言語である。専用のプロセッサがなくとも十分効率の良い処理系が実現できると考えられる。比較的コンパクトなので、いわゆるパソコン上でも実現できそうで、教育に向いている。また、人工知能ソフトウェアなど Lisp で記述するソフトウェアのドライバとしての役割も担っている。

level 2 は Lisp 上の大規模ソフトウェア開発・実行用であり、強力な専用プロセッサと高性能のコンパイラの存在を前提としている。したがって Common Lisp 程度の言語水準になる。しかし、基本構造は level 1 とあまりかわらず、組み込み関数の追加・機能強化が level 1 との主な相違である。level 2 の仕様は現在検討中で、この程度の大筋しか決まっていない。最終的には、基本構造はずいぶん異なっていても、機能的には Common Lisp に似たものになるだろうといわれている。

5. おわりに

Common Lisp、Scheme、Eulisp を簡単に紹介した。Common Lisp に関しては、すでに多くの処理系が利用され、またアプリケーションの開発もさかんに行われているために、今後、従来の仕様と著しく互換性を損なうような変更はなさそうである。一方、Eulisp の仕様は現時点ではまだまだ流動的である。設計にあたっては、Common Lisp の短所・長所も議論されており、これらの結果が最終仕様に反映されるものと思われる。今後の成果を期待したい。

参考文献

[1] J.McCarthy 他 "LISP 1.5 Programmer's Manual", The MIT Press, 1962.

[2] W.Teitelman, D.G.Bobrow, A.K.Hartley, and D.L.Murphy "BBN-LISP:TENEX Reference Manual", Bolt, Beranek, and Newman, Inc., 1971.

[3] W.Teitelman 他 "INTERLISP Reference Manual", Xerox Palo Alto Research Center, 1978.

[4] R.R.Burton 他 "Overview and Status of InterLISP-D", Conference Record of the 1980 LISP Conference, Staford University, 1980.

[5] D.Moon "MacLisp Reference Manual, Revision 0", MIT Project MAC, 1974.

[6] K.Pitman "The Revised MacLisp Manual", MIT/LCR/TR 295, 1983.

[7] R.Greenblatt "The LISP Machine", Working Paper 79, MIT AI Lab, 1974.

[8] T.Knight "The CONS Microprocessor", Working Paper 80, MIT AI Lab, 1974.

[9] G.L.Steele Jr. and G.J.Sussman "Design of LISP-Based Processors or, SCHEME: A Dialectric LISP or, Finite Memories Considered Harmful or, LAMBDA: The Ultimate Opcode", MIT AI Memo 514, 1979.

[10] D.Moon, R.Stallman, and D.Weinreb "LISP Machine Manual, Fourth Edition", MIT Artificial Intelligence Lab, 1981.

[11] G.S.Burke, G.J.Carrette, and C.R.Eliot "NIL Reference Manaual", MIT/LCS/TR-311, 1984.

[12] S.E.Fahlman, J.Large, and M.J.Cellio "Spice Lisp User's Guide", Carnegie-Mellon University, 1983.

[13] G.L.Steele Jr. 他 "An Overview of Common LISP", Conference Record of the 1982 ACM Symposium on LISP and Functional Programming, Pittsburgh, 1982.

[14] G.L.Steele Jr. "Common Lisp Reference Manual, Laser Edition", Carnegie-Mellon University, 1982.

[15] G.L.Steele Jr. "Common Lisp Reference Manual, Mary Poppins Edition", Carnegie-Mellon University, 1983.

[16] G.L.Steele Jr. "Common Lisp the Language", Digital Press, 1984.
（邦訳）井田昌之訳 "Common Lisp", bit 別冊, 共立出版, 1985.

[17] S.E.Fahlman 他 "Internal Design of Spice Lisp", Carnegie-Mellon University, 1983.

[18] S.Wholey and S.E.Fahlman "The Design of Instruction Set for Common Lisp", Conference Record of the 1984 ACM Symposium on LISP and Functional Programming, Austin, 1984.

[19] T.Yuasa and M.Hagiya "Kyoto Common Lisp Report", 帝国印刷, 1985.

[20] 井田昌之 "ANSI X3J13", Common Lisp アラカルト 7, bit Vol.19 No.3, 1987.

[21] D.G.Bobrow, K.Kahn, and G.Kiczales "COMMONLOOPS", IJCAI 85, 1985.

[22] D.G.Bobrow 他 "CommonLoops", OOPSLA 86, 1986.

[23] D.G.Bobrow 他 "Common Lisp Object System Specification", X3 Draft 87-003, 1987.

[24] G.J.Sussman and G.L.Steele Jr. "Scheme: an interpreter for extended lambda calculus", MIT AI Memo 349, 1975.

[25] G.L.Steele Jr. and G.J.Sussman "The Revised Report on Scheme, a dialect of Lisp", MIT AI Memo 452, 1978.

[26] W.Clinger 編 "The Revised Revised Report on Scheme, or an uncommon Lisp", MIT AI Memo 848, 1985.

[27] J.Rees and W.Clinger 編 "Revised^3 Report on the Algorithmic Language Scheme", 1986.

[28] 幕足譲 "Scheme", Common Lisp アラカルト 6, bit Vol.19 No.2, 1987.

[29] H.Abelson and G.J.Sussman "Structure and Interpretation of Computer Programs", The MIT Press, 1985.

[30] J.Padget 他 "Desiderata for the Standardisation of Lisp", Proc. of the 1986 ACM Conf. on Lisp and Functional Programming, 1986.

[31] M.Griss, E.Benson, and G.Maguire Jr. "PSL: A Portable LISP System", Proceedings of the 1982 ACM Symbosium on Lisp and Functional Programming, Pittsburgh, August, 1982.