ソフトウェア評論

関数型言語 ML

武市 正人

近年，関数型のプログラム言語が注目されてきているが，筆者は言語 ML の処理系に親しむ機会があったのでこれを紹介するとともにいくつか気付いた点を指摘してみたい．

LCF と ML

ML (Meta-Language) という奇妙な名前で呼ばれている関数型言語は，まだ広く知られているというものではないようである．そこで，まず言語と処理系の歴史の紹介から始めることとする．

1971 年頃，Stanford 大学で定理の証明支援系 LCF (Logic for Computable Functions) が開発された．そのプロジェクトに参画した R. Milner が 1974 年に Edinburgh 大学で LCF をもとにして，Edinburgh LCF のプロジェクトを始めた．その際に，証明支援系に証明の手順を与えるのに用いることを目的として設計されたものが ML である．つまり，証明支援系の超言語として生まれたものである．Edinburgh LCF に関しては[4]が出版されていて，その第 2 章 pp. 13-61 が完結した ML の利用者手引きになっている．この ML の処理系は LCF と一体となっていて DEC-10 上の Lisp 1.6 で作成されているとのことである．残念ながら筆者は使ったことがない．

この ML は，一言で“ML is a functional language in the tradition of ISWIM and GEDANKEN”[4]とのことであるが，現在に至るまで ML の特徴として挙げられる概念や機能はこうした背景で考え出されたものである．ML の言語の説明は本稿の主題ではないので，上にあげた文献を参照されたい．

Masato Takeichi，電気通信大学電気通信学部計算機科学科．

1984 年 5 月 1 日受付．

Cardelli の VAX ML

1981 年に，Edinburgh 大学にいた Luca Cardelli (現在，AT & T Bell Laboratories) は LCF とは独立に，関数型言語としての ML の処理系を VAX11 の操作系 VMS 上に実現した．その後，慶応大の斎藤信男氏が Carnegie-Mellon 大学に滞在中，1982 年に VAX11 の Unix 上への移植に携わったとのことである．筆者は斎藤氏を通じて ML の処理系を手にすることになった．Cardelli 氏，斎藤氏の御好意によりその処理系 VAX ML (LCF のものと区別するため，Cardelli ML，Luca ML 等とも呼ばれていた) を東京大学大型計算機センターの VAX11/780 の Unix 上で使用した．筆者の近くには VAX がないため，300 BPS で電話線経由というのんびりした使い方であったが，関数型言語の基礎的な概念を理解することを目標とした大学院学生向けの講義の資料の作成という当初の目的は達成された．筆者の知っている実用的な関数型言語は多くないが，ML には他の関数型言語には見られない重要な概念が反映されているところに興味をおぼえたのである．代表的なものとして，

(1) 多様型 (polymorphic type) の考え方に基づいて“型”のついた対象を扱う．

(2) 抽象データ型 (abstract type) を定義する機能がある．

ことがあげられる．もっとも，副作用を伴う代入などの“不純な”ものも含まれているが，本稿ではこれらに対して評価することは避けたい．

VAX ML の処理系にはいくつかの“虫”が棲んでいて，安心して使うというわけにはゆかなかったが筆者は楽しむことができた．この ML の文法は LCF のものとそれほど変わっていない．いくつか拡張された機能もあるが図 1 に示した例を理解するには[4]にある説明で十分である．抽象データ型 *sequence (これは多様型の一

```
-let rec type
=       * sequence <=> * # * stream
=   and
=       * stream <=> . -> * sequence
=   with
=       hds s = fst(repsequence s)
=   and
=       tls s = repstream(snd(repsequence s))()
=   and
=       prefixs x s = abssequence(x, absstream s)
=   ;
> type *a sequence = -
| type *a stream = -
| hds = \ : *a sequence -> *a
| tls = \ : *b sequence -> *b sequence
| prefixs = \ : *c ->   ((. -> *c sequence)  -> *c sequence)

-let rec
=   filter p s =
=       let x=hds s  and s'=tls s  in
=           p x => prefixs x (\(). filter p s') | filter p s'
=   ;
> filter = \ : (*a -> bool)  ->  (*a sequence -> *a sequence)

-let rec
=   take n s =
=       n=0 => [ ], s |
=           let x=hds s  and s'=take (n-1) (tls s) in x_fst s', snd s'
=   ;
> take = \ : int ->  (*a sequence ->   ((*a list)  # *a sequence))

-let primes = sieve (from 2)
=       where {
=           rec from n = prefixs n (\(). from (n+1))
=       and
=           rec sieve x =
=                   let p=hds x  and x'=tls x  in
=                       prefixs p (\(). sieve (filter (\n.  n mod p <>0) x'))
=       }
=   ;
> primes = - : int sequence

-take 15 primes;
  ([2;3;5;7;11;13;17;19;23;29;31;37;41;43;47],-) : (int list)  # int sequence
Pars 14, Anal 10, Comp 6, Assm 8; Total 38 msec
Run 10, Collect 0 msec.
```

入力（抽象データ型の定義）

処理系の応答（抽象データ型とそれを操作する関数の型）

（関数の定義）

（関数の型）

関数の評価　結果　結果の型

処理時間

図 1　UEC ML の使用例

つでもある）を用いて，素数の列 primes を定義したものである．処理系による型の推論や多様型の扱いなどの点で ML の雰囲気を味わうことができるであろう．

やや細かいことになるが，この例でも推測されるように，関数の評価方式は“値呼び”である．また，言語自体に引数の評価を遅らせる機構は用意されていないので“\( ). e”（\はλの代用）の形で e の評価を遅らせている．この点は言語機能の面から議論されることの一つであろう．

UEC ML

1983 年 5 月頃，筆者らは三菱電機の MELCOM COSMO 700Ⅲ（現在は 800Ⅲ）と同年秋に導入されることになっていたミニコン MELCOM 70/250 に VAX ML を移植することとした．当時，大学院学生であった中条秀和君（現在，日本 IBM）の研究の一環としてこの作業を行なった[3]．あとで述べるように，いろいろ難問はあったが，同年 10 月頃には VAX 版と同様に動くようになり，機能の追加や，VAX 版に潜んでいた“虫”の駆除も行ない，安定した処理系になっている．

ところが，つぎに紹介するように，ML の主流派（？）は“標準”ML へ向っていて，VAX の上では上に述べた VAX ML ではなく Standard ML が動く情勢になってきた．筆者のところで作成した ML は VAX ML をもとにしたものではあるが VAX ML と呼ぶわけにもゆかないし，単に ML と呼ぶわけにもゆかないので，UEC ML（University of Electro-Communications ML）と呼ぶことにした．

実は，図 1 は MELCOM 70/250 の VOS 上で UEC ML によって実行したものである．

Standard ML

VAX ML のマニュアルは 1983 年 6 月頃には用意されると言われていた．しかし，1983 年 11 月，UEC ML

が動いて間もなく，Milnerによって“Standard ML”の提案が出され，同じ頃にCardelliによってVAX MLを変更したStandard MLの処理系(Pose 2)が送られてきた．この処理系は“標準”MLのすべての機能を実現しているわけではないが，VAX MLにはなかったモジュールの機構や関数を引数のパターンに従って場合分けで定義する形式(clausal form)が付加されている．構文も大幅に変更された．

Standard MLは，Edinburgh大学で開発されたHopeの特徴的な機能をVAX MLに融け込ませて整理したものとも考えられる．

CardelliはPose 2のあと，1か月ほどしてPose 2には“虫”がいたからといってPose 3を送ってきた[1]．この間に構文まで変わっているという忙しさである．Pose 3にも“虫”がいるようである．1984年夏頃までにStandard MLの処理系を(VAX11のUnixの上に)固定するとのことである．現在は処理系の利用者からのコメントを求めているところである．

以上が筆者がMLと接してきた状況である．

MLの処理系

CardelliによるVAX MLとStandard MLの処理系はいずれもVAX11上で動くものである．上に紹介したように，現在のものは開発途上にある処理系であり，利用者が(言語の基本的な機能とは別に)実用上要求すると考えられる機能——たとえば，デバッガ，トレーサなど——は今後，整備されるものと期待したい．

利用者は，図1の例のように対話形式でプログラムの実行を進めてゆく．もちろん，ファイルに用意したプログラムを読み込んで実行することも可能である．関数の定義は，その値を計算する機械語のコードに翻訳される．高階関数も扱うので，一般に環境とコードの対が関数の値である．異なる言語間で比較するのは難しいが，筆者の得た感触では，VAX11/780上でもMELCOMの上でも，MLによる関数の実行は，それぞれに代表的なLispの処理系Franzlisp, HLISPでコンパイルされたコードの実行よりやや遅い(数倍？)程度である．この点から，これらのMLの処理系は，今後の改良を期待して実用的なものであるといえよう．

MLが広く使われるためには処理系の移植性が重要である．上に触れたHopeはPOP-2で書かれていて移植性には問題がある．Cardelliは移植性も考慮して翻訳処理の部分はPascal(約10000行)で書き，翻訳の際には仮想のスタック機械のコード[2]を生成し，それを，対象とする機械語に変換するという方式を用いている．UEC MLのもとになったVAX MLでは，実行時支援系はアセンブリ語で3000行程度であった．UEC MLではアセンブリ語の部分を少なくして，1200行とし，あとはすべてPascalで書いてある．翻訳処理の部分のPascalのプログラムのなかで機械に依存する部分は約2000行であった．しかし，それ以上にPascalの方言に依存する部分が問題となろう．標準的ではない機能として，分割翻訳，外部参照，ファイルの取込みなどが使われているし，名前は25文字以上識別する必要がある．UEC MLの移植に際しては，天海良治君がUEC Pascalに手を加えてこれに対処した．これによってUEC Pascalの機能も充実したという副産物も得られた．

Standard ML(Pose 3)では，翻訳部分のPascalのプログラムがいくらか増え，実行時支援系はCで8000行程度で書かれている．

この程度の大きさのものになると，処理系の作成者が移植性を十分に考慮しておかないと移植は極めて難しくなってしまう．

筆者は，電電公社武蔵野通研の梅村恭司氏によるLispの処理系[6]を試験的にVAX11からMELCOMに移植してみたが，割込み処理などの部分を除いて数時間という仕事であった．“標準の”Pascalの機能しか使っていないからである．Lispの翻訳結果もまたPascalのプログラムなので，機械に依存した部分が存在しないということも移植には効果的であった．実行時の効率を重視した処理系では翻訳結果のコードが機械に依存することは止むを得ないであろうが，機械に依存しない部分についてはもとの処理系の作成者の配慮が極めて重要であることの実証である．

MLの応用

MLは実用的な関数型言語として，広範囲に利用できるものと思われる．ことに，多様型や抽象データ型の概念によって，広くプログラミングの方法論にも影響を与えるものと思われる．

筆者のところでは，UEC MLの処理系作成と併行して大学院学生の大平剛君(現在，日本IBM)が，表示的

意味記述で与えたプログラム言語の意味関数をMLのプログラムに変換し，その言語の翻訳系を実現しようとする実験を行なった[5]．MLの高階関数，抽象データ型の機能とML処理系の高速な実行機能を利用したものである．UEC MLの開発中に行なったものであるので，多くの部分がLispで書かれているが，全体をMLで書くことも可能である．

現在，筆者の知るかぎりでは，VAX上のML処理系は，上に述べた東大大型計算機センターと慶応大の斎藤信男氏，東工大の米澤明憲氏のところで使われているようである．また，斎藤氏のところでは他機種への移植も行なわれているとのことである．

おわりに

筆者の経験からMLの処理系の評論を試みた．Standard MLの処理系が完成したときには，移植性の問題も解決され，広くMLが使われるように期待するとともに，関数型言語の処理系の実現法の研究をさらに進める必要性を感じている．

参考文献

[1] Cardelli, L.: ML under Unix, *Bell Labs Technical Memorandum*, Bell Laboratories, 1983.

[2] Cardelli, L.: The Functional Abstract Machine, *Bell Labs Technical Report*, TR-107, Bell Laboratories, 1983.

[3] 中条秀和，武市正人：多様型をもつ関数型言語MLの処理系の移植について，第28回情報処理学会全国大会講演論文集，1984, pp. 427-428.

[4] Gordon, M., Milner, R. and Wadsworth, C.: *Edinburgh LCF*, Lecture Notes in Computer Science, No. 78, Springer-Verlag, 1979.

[5] 大平剛，武市正人：言語設計システムについて，第28回情報処理学会全国大会講演論文集，1984, pp. 329-330.

[6] 梅村恭司：移植可能なTAO-LISPにおけるLispとPrologの融合法，同上，1984, pp. 419-420.