7K-6

多様型をもつ関数型言語MLの処理系の移植について

中条秀和　　武市正人
(電気通信大学計算機科学科)

0. はじめに

MLは、Edinburgh LCF[3]と呼ばれるプログラムの自動証明システムのメタ言語(Meta Language)として開発された関数型言語である。このML処理系はLispで記述され、DEC-10で動くように作られた。その後、LCFとは独立にPascalで記述されたMLがVAX-11のVMSやUNIXオペレーティングシステム上に移植された。MLは次のような特徴を持っている。

- 高階関数(higher order function)を扱うことができる。
- 豊富な構造型(list, pair, union, .., etc.)が用意されている。
- 抽象型(abstract type)が備わっている。
- 多様型(polymorophic type)の概念に基づいている。
- トラップを扱う機能が用意されている。

また、MLの処理系の特徴として次のようなことがあげられる。

- 会話型の処理系である。
- 与えられた関数の定義や式を、機械語までコンパイルするため実行が速い。
- 式や関数の実行まえに型検査を行なう。そのとき、利用者が特に型を定義しなくても、自動的に与えられた式に対して型を類推する。

VAX-UNIX上で動くML処理系(VAX ML CMU版)をもとに、Melcom Cosmo 800-IIIと、Melcom 70/250のUTS/VS & VOSオペレーティングシステム上に、簡単な入出力機能を追加されたUEC MLの処理系を作成した。

1. 移植作業

UEC ML処理系はそのほとんどの部分がPascalで記述されていて実行時支援系などの機械に依存する部分はアセンブラで記述されている。そのソースプログラムは、MLのプログラムをコンパイルする各段階に分けてモジュール化されている。

各モジュールは次の通り。

- globh(大域変数宣言)
- main(主プログラム)
- debg(デバッグおよびFAMのブート)
- scan(辞句解析)
- pars(行文解析)
- anal(型検査)
- comp(中間コード生成)
* assm(機械語生成)
* rout(Pascalで記述された実行時支援系)
* rtim(アセンブラで記述された実行時支援系)
* stor(記憶管理)
* fchk(Pascalで使う機械依存部分)

(・は多少の修正、*は新しく作り直したモジュール)

Pascalで記述されている部分については次のような変更と修正を加えた。1) Pascal処理系についての修正。

これは名前の長さに関する修正とファイルの取り込み(include)機能の追加である。

2) ML処理系のソースプログラムについての修正。

これは、モジュール分割の方法や、入出力のタイミング等のPascalの方言に関する部分と、取り扱う文字コードに依存している部分(ASCIIコードとEBCDICコードの違い)の修正と、他の機械に移植をすることを考慮してVAX-11への依存が強い部分の変更である。

このような変更と修正を行なうことにより、中間コードを生成するところまでが実行可能となった。その後、中間コードから機械語を生成する機械語生成系と、セルを割り当てたりごみを回収する記憶場所の管理を含む実行時支援系を作成した。

1.1. FAM

ML処理系では与えられた式や定義をそのまま機械語に変換するのではなく、仮想機械FAM (Functional Abstract Machine)のコードを中間コードとし、それをもとに機械語を生成している[1]。FAMの状態は<AS,RS,FR,PR,TS,ES,M>という7つ組で表現される。AS(Argument Stack)は局所的な定義や、演算用のスタック、RS(Return Stack)は戻り番地等を積むスタック、FR(Frame)は現在評価中の関数の閉包(closure)を指す。閉包は関数のコードとその環境をもつ。PR(Program)は現在評価中の関数のコード、TS (Trap Stack)はトラップが起きたときにFAMの状態をもどすためのスタック、ES(Environment Stack)は大域的な環境のスタック、M(Memory)は現在の記憶の状態である。

FAMコードはこれらの状態遷移により定義されている。

1.2. 機械語生成系

FAM は一種のスタック機械であり、そのコードのほとんどは AS に対する状態遷移である。そこでFAMコードのASに対する操作について簡単なスタックシミュレーションを行ないコードを生成するものを作成した。生成されるコードは、ごみ回収のときに再配置されることも考慮して設計した。

1.3. 実行時支援系

UEC ML では、実行速度を考慮しなければならない部分、アセンブラでなければ記述できない部分以外はすべてPascal で記述し、移植性を高めた。

VAX ML では、ほとんどアセンブラで記述されていた。

1.3.1. 記憶管理

ML で用意されている構造型などは、そのほとんどが可変長のセルであることから、このセルの割り当ては次のような方法をとった。

2つの領域A,Bを確保し、領域Aからセルを割り当ててゆき、領域Aでセルが割り当てられなくなったとき、有効なセルをすべて領域Bに移す。そしてこんどは領域Bからセルを割り当てていく。領域Bで割り当てることができなくなれば同様に領域Aに有効なセルを移す。有効なセルは、AS、ES、RS、FR、PR、TS から指されているものであり、現在評価中の関数のコードも含んでいる。

ML のさまざまな型のデータはそのデータのセルを指すポインタとしてスタックに積まれる。整数(int)や、論理値(bool) などは、そのままスタックに積まれる。このことから有効なセルを回収するときに、スタックに積まれているものがポインタなのか、そうでないか、またどのような型を指すポインタなのかを区別するために、UEC ML ではポインタにその情報を持たせている。

VAX ML では、領域をさらに、ページに分割し、ページないには同一の型のデータをおさめるようにしている。

2. 機能の追加

2.1. 入出力

VAX ML CMU版は試験的なものであり、入出力の機能は実現されていなかったが、入出力機能がなければ ML でプログラムを記述することの大きな制限となる。そこで UEC ML 処理系ではとりあえず簡単な入出力機能を追加した。

入力は関数 getfile を評価することによって行なう。関数 getfile はファイル名を受け取り、そのファイルの内容をすべて文字のリスト(token list)として返す。

出力は、関数 reset、putch、puttok を評価することによる。関数 reset は、ファイル名を受け取り、そのファイルを開き unit という値を返す。関数 putch は文字列を受け取り、それを関数 reset で開いたファイルに出力し unit という値を返す。関数 puttok は、文字列のリストを受け取り、その最初の要素を関数 reset で開いたファイルに出力し、残りのリストを値として返す。このように、出力はこれらの関数の副作用として行なう。

3. おわりに

現在の UEC ML は、簡単な入出力機能を加えたが、そのほかにも機能的に不足しているところがある。たとえば、会話型の処理系ではあるがその機能は非常に乏く、エディタなどを使わないで ML のプログラムを記述することは困難である。また、定義をした関数や型などをファイルなどに保存することもできない。また、生成される機械語にも改善の余地がある。しかし、多様型や抽象型、高階関数などを用いて手続的な言語により表現と対比することなど実用的な関数型言語の処理系として有用である。

なお、Cardelli によって標準 ML の文法が1983年末に定められ、その処理系がVAX上に作成されている[2]。

[1] L. Cardelli."The Functional Abstract Machine",Bell Labs Technical Memorandum TM-83-11271-1, Bell Labs Thechnical ReportTR-107,1983.

[2] L. Cardelli."ML under Unix", Bell Laboratories,1983.

[3] M. Gordon,R. Milner,C. Wadsworth." Edinburgh LCF", Lecture Notes in Computer Science,n.78,Springer-Verlag, 1979.