4R−9

新ELISのプログラム開発支援系*

高田久靖†　　鶴巻宏治‡
tkd@ntt-elis.ntt.jp　　tsuru@ntt-elis.ntt.jp
NTTヒューマンインタフェース研究所§

1　はじめに

新ELIS(ELIS8200)[1]は実用的な記号処理ソフトウェアの開発実行マシンとして開発されたLispマシンである。新ELIS上では、TAO/ELIS[2]のマルチパラダイム性を受け継ぎつつ、Common Lispに完全に準拠したELIS Common Lispが実現されている。

本稿ではELIS Common Lispのプログラム開発支援系(以下、開発支援系)の設計方針・機能を述べると共に、複数のプログラミングパラダイムを含む融合型プログラムの実行追跡機構の実現法について報告する。

2　開発支援系の設計方針

本章では、開発支援系の設計方針について述べる。

(1) インタプリタ環境の重視

Common Lispの言語仕様は非常に豊富であるため、従来の処理系はまずコンパイルされたCommon Lispプログラムの実行性能を高めることに重点が置かれてきた。これに対し、ELIS Common Lispでは専用マシンの特質を生かして、まずインタプリタの高速化による、インタプリタ環境での大規模プログラムの効率的実行を可能にしている。このインタプリタ環境の利点を最大限生かして、開発支援系を実現する。

(2) プログラミングパラダイムに依存しない開発支援機能を提供

使用するプログラミングパラダイムの種類によらずに、融合型プログラムの開発を支援する機能を提供する。

*Program Development Tools of new ELIS system
†TAKADA, Hisayasu
‡TSURUMAKI, Koji
§NTT Human Interface Laboratories

3　開発支援系の特徴と実現上の問題点

本章では、前章で述べた設計方針に基づき、開発支援系を構成する各種ツール群に要求される機能的特徴と、その実現上の問題点について述べる。

(1) 融合型プログラムの実行過程を追跡する機能の提供

ELIS Common Lispが提供する3つのプログラミングパラダイム（関数型、オブジェクト指向、論理型）に対し、一定の方法でプログラム実行過程を追跡する機能が必要になる。しかし、各々計算モデルが異なり、プログラム実行の実現形態も全く異なる。それを考慮して、統一的な実行追跡機構を実現する必要がある。

(2) 豊富なデータ構造の表示/変更機能の提供

Common Lispが提供する豊富なデータ型の他、オブジェクト指向/論理型プログラミング機能が提供するオブジェクト/節といったデータを手軽に表示/変更する機能が必要になる。そこでオブジェクト/節も特殊な型のデータとして扱えるようにLisp機能を拡張し、表示/変更機能をツールとして実現する。

(3) デバッガの提供

融合型プログラムの実行過程でスタック上に作り上げられるフレーム情報を基に、パラダイムを意識してプログラムの実行状態を表示/変更する機能が必要になる。これは処理系でフレーム情報取得用作業関数を用意し、表示/変更機能をデバッガとして提供する。

4　融合型プログラムの実行追跡機構の実現

本章では、3 (1)で述べた機構の実現法について述べる。

Common Lispプログラムの実行追跡機能を実現するために、Common Lisp言語仕様にはプログラム実

行をS式の評価毎または関数の呼出し毎に中断する機能(フック機能)がインタプリタ環境では用意されている。一方、Elis Comomn Lispが継承したTAO/ELISのマルチパラダイム性は、LispおよびS式と調和するように構築されている。従って、オブジェクト指向/論理型プログラミングパラダイムを含む融合型プログラムの実行追跡機構もCommon Lispのフック機能を拡張することで実現することが自然であると考えられる。その際、フックすべきプログラム実行単位を各々メッセージ送出とユニフィケーションとする。

4.1 メッセージ送出に対するフック機能

メッセージ送出を含むプログラムを評価する際にフック条件が成立すると(即ち、当該プログラム評価時にスペシャル変数*evalhook*/*applyhook*[3]にnil以外が束縛されている場合)、関数呼び出し時点と同様にメッセージ送出時点でも実行が中断され、ユーザ定義されたフック関数を評価する。フック関数は*evalhook*/*applyhook*の束縛値として与えられ、実行追跡を行なう。

4.2 ユニフィケーションに対するフック機能

論理型プログラミング機能を使用したプログラムを評価する際に4.1で述べたフック条件が成立すると、処理系は4種類のユニフィケーション開始/終了点においても、プログラムの評価を中断し、ユーザ定義されたフック関数を評価する。フック関数はスペシャル変数*revalstep*の束縛値として与えられ、実行追跡を行なう。ユニフィケーション開始/終了点は、箱(box)モデルに基づくcall(ユニフィケーション開始), exit(ユニフィケーション成功),redo(バックトラック),fail(ユニフィケーション失敗)である[4]。

論理述語concat(図1参照)およびCommon Lisp関数foo(2を返す無引数関数)を含む融合型プログラムの実行追跡例を図2に示す。図2において、先頭行がプログラムである。また、"(foo)"の前のコンマ(,)は「ユニフィケーションの直前に"(foo)"を評価する」ことを指定する。従って、まず"(foo)"の実行が追跡され、その後にconcatのユニフィケーションが追跡されている。

```
(assertz (concat (_a . _x) _y (_a . _z))
         (concat _x _y _z))
(assertz (concat () _x _x))
```

図1 論理述語concat

```
>(goal (concat _x _y (1 ,(foo))))
  <IN .1> foo
  <OUT.1> foo -->(2)
<in .1> concat[Call] [ _x = undef]
        [ _y = undef] [ _z = (2)]
<in .1> concat[Call] [ _x = undef]
        [ _y = undef] [ _z = nil]
_x = (1 2)
_y = nil
t
```

図2 融合型プログラムの実行追跡例

5 おわりに

本稿では新ELISシステム上の開発支援系の設計方針・機能を述べると共に、特に融合型プログラムの実行追跡機構の実現法について報告した。

Common Lispのフック機能をオブジェクト指向/論理型プログラミング機能に対して拡張したことで統一的な構成でプログラム実行追跡機構が実現できた。

今後は、使用経験のフィードバックを行なうことによって、より使い易い開発支援系の開発を行なう。

参考文献

1) 鈴木、家吉、杉村:「新ELISシステム概念」、信学全大、1989秋季

2) Takeuchi,I.et al.,"A List Processing Language TAO with Multiple Programming Paradigms", New Generation Computing, 1986

3) Steele, G. L., "Common Lisp : the Language", 2nd Edition, Digital Press, 1990

4) Clocksin, W. F. and Mellish, C. S., "Programming in Prolog", Springer-Verlag, 1981