7F−4　コンビネータ項書換え系にもとづく関数・論理融合型言語の設計

西岡 知之　井田 哲雄
筑波大学

1　はじめに

関数型言語と論理型言語は、いずれも宣言型言語の枠組でとらえることができることから、近年これらを融合させた関数・論理融合型言語についての研究が盛んに行なわれている。関数・論理融合型言語とは、関数型言語、論理型言語のそれぞれの分野での理論／実現上で得られた知見を損なうことなく融合させた言語である。これらの言語においては、遅延評価などに代表される関数型言語の利点と論理変数などの論理型言語の利点とを同時に用いることができる。この種の言語の代表的な例としては、K-LEAFやBABELなどがあげられる。

我々の研究グループも、関数・論理融合型言語のための遅延ナローイング計算系 *LNC*[1] や *LNC* に基づくプログラミング言語Evなどをこれまでに発表してきた。

本発表では、理論 *LNC* を元とする言語Evの拡張について述べる。

2　Evの概要

プログラミング言語Evは、条件付き項書換え系(CTRS)に基づく関数・論理融合型言語である。クラス III_n に属するCTRSの書換え規則をプログラムとみなし、計算機構として遅延ナローイング計算系 *LNC* を用いることにより、論理変数を含む計算を遅延評価で計算することができる。Evにおいては、プログラムはまず基本式といわれる形式に変換された後に *LNC* の推論規則を使って実行される。遅延評価の能力により、無限のデータ構造を扱うような計算も可能である。

図1に Ev によるプログラムの例をあげる。言語Evについては、すでに抽象マシンLNAMによる実装がなされている[2]。

```
ones = [s(0) | ones].

nth( 0,[H|T] ) = H.
nth( s(X),[H|T] ) = nth( X, T ).
```

図1:Evのプログラム例

3　Evの拡張

言語Evは、*LNC* の試験的インプリメンテーションとしての側面が強く、実用的なプログラミング言語としての能力は十分だとはいえない。そこで、以下のような拡張を行なうことを検討している。

- 高階関数の導入
- 多相型システムの導入
- 制約解消系の導入

これらの拡張のため、構文を含めた言語全体にわたって見直しを行なった。その際、理論的な性質の検証のしやすさを失うことのないようにするため、CTRSとしての枠組は維持する。

3.1　構文

Haskell等の関数型言語の特徴を大幅に取り込む形で新たな構文を定めた。取り込んだ関数型言語の特徴としては、静的なスコープを持つ局所変数の導入、可読性向上のためのレイアウトルールの導入、書換え規則の適用順序に優先順位をつける、などがあげられる。また、条件部に

A Functional-Logic language based on Combinatory Term Rewriting System
Tomoyuki Nishioka, Tetsuo Ida
University of Tsukuba

ガードを導入し、条件部分を簡単に計算できるものとそうでないものをプログラマが指定できるようにして、実行時の規則の選択の効率化を計る。この目的のため、ガード内部においては、組込みの関数／述語のみが使用できるものとする。

3.2 高階関数の導入

関数型言語に見られる高階関数の機能は非常に有用であるが、一階のCTRSをベースとする現在のEvにその機能はない。そこで、関数計算の部分に高階の機能を導入する。単純に高階の機能を導入すると、高階のE単一化が必要となり、計算量が膨大なものになってしまうので、実用性という観点から、制限つきで高階の機能を導入する。具体的には、ゴール中に現れる論理変数は一階のものしか許さない。つまり、ゴール中の論理変数に高階関数を束縛するようなことは許さないことにする。

3.3 制約解消系の導入

LNC においては、計算解の領域としてHerbrand領域のみを考えているので、算術演算を行ないたいときには、数項によるコーディングなどの技法が必要になる。しかし、この方法は実際的ではないので、有理数などの特定の領域における制約問題については、これを解決するための制約解消系を用意し、数などを直接扱うことにより効率の向上を目指す。

3.4 型システムの導入

関数型言語の分野では、プログラムの信頼性の向上、実行時の効率の向上などに型システムが大きく貢献している。一方、新しいEv においては、制約解消系の導入にともない、制約解消系が扱うべきデータを、従来からのデータと区別する必要が生じる。また、停止性や合流性などの理論的な性質の検証に当たっては、型理論の存在が重要である。これらの理由から、Ev に型システムを導入する。型付けに関しては、関数型言語Haskellなどで用いられているMilnerの多相型を基本とし、論理変数も取り扱えるような型システムを作る。

4 プログラム例

拡張後のEv のプログラム例を図2に示す。関数condmapは、与えられたリストの要素のうち、条件P を満たす要素にのみ関数F を適用させたリストを返す。述語positiveの条件部のX > 0は、制約解消系によって解かれる。

```
condmap :: (*->bool)->(*->*)->[*]->[*].
condmap P F [] = [].
condmap P F [H|T] = [ F H | condmap P F T ]
                                        :- | P H.
condmap P F [H|T] = [ H | condmap P F T ].

positive :: num->bool.
positive X :- | X > 0.

add1 :: num->num.
add1 X = X + 1.
```

図2: 新しいEv によるプログラムの例

このプログラムのもとで、次のようなゴールを与えると、自動的に型が推論され、答とその型が得られる。ここでX が論理変数であることに注意。

```
?- condmap positive add1 X = [2,0,3].
X = [1,0,2] : [num]
```

図3: プログラムの実行例

5 おわりに

関数・論理融合型言語Ev と その拡張について述べた。現在、型推論系、コンパイラの一部を製作中である。今後の課題としては、理論的な性質の検証や、モジュール化等について考察などがあげられる。

参考文献

[1] Satoshi Okui and Tetsuo Ida. Lazy narrowing calculi. Technical report, Institute of Information Science and Electronics, University of tsukuba, 1992.

[2] 中村敦司, 鈴木太朗, 中川康二, 井田哲雄. 遅延ナローイング抽象機械のアーキテクチャ. 情報処理学会第44回全国大会予稿集, No. 6G-1, 1992.