数理解析研究所講究録
第625巻 1987年 98-107

融合型プログラミング言語と処理系の実現について

三宅　延久，富樫　敦，野口　正一
(東北大学　電気通信研究所)

1　はじめに

今日、Prologは一階述語論理をベースとし、ユニフィケーションや非決定的な処理が出来る言語として、注目を集めているが、Prologは次のような点の処理を行う能力に欠如しています。その点は(1)関数を扱うこと (2) Equality relationを扱うことの二点です。その欠点を解決すべく、論理型の言語に関数のfeatureを取り入れた、すなわち、融合型言語が注目を集めるようになって来ました。そのような言語の中でも、数学的な基盤を持ったものは、セマンテクスが明確であるので、プログラムが分かりやすく、また、数学から得られた種々の結果をアプリケーションとして使うことも可能といった長所を持っているので、言語として望ましいと言える。我々は数学的基盤として等式論理を持った融合型言語[3]に関心を持っており、その言語の処理系(インタプリタ)を製作した。ここではその処理系実現に関することについて説明します。

2　融合型プログラミング言語

本章では、融合型言語のベースとして選んだ等式論理についての説明を行います。シンタックス、オペレショナルセマンティクスについて述べます。

2.1等式論理のシンタックス

定義2.1 (等式ゴール節)

等式ゴール節は次のような形をしている。

$\leftarrow M_1=N_1,....,M_q=N_q.$

ただし、$M_j, N_j (1\leqq i\leqq p, 1\leqq j\leqq q)$ はターム

定義2.2 (等式定義節)

等式定義節は次のような形をしている。

$L_1=R_1\leftarrow M_1=N_1,....,M_q=N_q.$

ただし、$L_1,R_1,M_j,N_j(1\leqq i\leqq p,1\leqq j\leqq q)$はターム

これらの等式節の左辺はimplicitに左から右への方向づけがなされていて、$L_1=R_1$は$L_1\rightarrow R_1$という意味である。すなわち、等式節の直感的意味は、$M_1=N_1$から$Mq=Nq$までの左辺の等式がすべて成立する時に、L_1からR_1へ書き換える事ができるという意味である。

定義2.3(プログラム)

等式論理プログラムは、等式定義節の有限個の集まりである。

2.2 等式論理のオペーショナル セマンティクス

等式論理プログラムでは、計算はリゾリューションではなく、SuperpositionとReflectionによって行われます。ここでは、それを示します。

定義2.4(**reflectant**)

ゴールG:$\leftarrow M_1=N_1,....,M_q=N_q.$が与えられているとする。もし、

m.g.u σが存在して、$M_1\sigma=N_1\sigma$であるならば、その等式を削除できてゴールG'←$(M_2=N_2,....,M_q=N_q)\sigma$が得られ、その時G'はGのreflectantと呼ぶ。

例

G:←rational(Z,W)×Y=rational(X,6)×4,2×W=Z×3

G':←2×6=X×3

定義2.5(**goal superposant**)

ゴールG: ←$M_1=N_1,....,M_q=N_q$と、

定義節C: L=R←$L_1=R_1,....,L_r=R_r$が与えられているとする。goal superposant G'は、次のようにして、G,C,m.g.u σから得られる。それは

M_1のオッカーレンスt_1にあるサブタームT_1が、Lとm.g.u σで、ユニファイ可能な時(すなわち$T_1\sigma=L\sigma$)、G'は

←$(M_1[t_1←R]=N_1,L_1=R_1,....,L_r=R_r,M_2=N_2,....,M_q=N_q)\sigma$

であるか、または、N_1のオッカーレンスu_1にあるサブタームU_1が、Lとm.g.u σで、ユニファイ可能な時(すなわち$U_1\sigma=L\sigma$)、G'は

←$(M_1=N_1[u_1←R],L_1=R_1,....,L_r=R_r,M_2=N_2,....,M_q=N_q)\sigma$

である。

例

C:rational(X,Y)=rational(Z,W)←X×W=Z×Y

G:←rational(2,3)×Y=rational(X,6)×4

G':←rational(Z,W)×Y=rational(X,6)×4,2×W=Z×3

定義2.6(**derivation**)

Qを等式論理プログラム、G,G'をゴールとする。Q∪{G}からG'のderivationとは(Q∪{G}⊢G'と表記する)、$G_0,G_1,....,G_n$の有限系列で

1.G_0はG,G_nはG'である

2.G_iはG_{i-1}のreflectantかgoal superposantである。

とくに、G'が空節の時には**refutation**と呼ぶ。

以上述べたオペレションによって、等式論理プログラムは計算が実行されます。

定義2.7(解)

Qを等式論理プログラム、Gをゴールとする。その時、解とは、

1.Q∪{G}からのrefutation(Q∪{G}⊢空節)が存在する。

2. 各reflectionまたはgoal superpositionに、使ったm.g.uを$\theta_1,\theta_2,...,\theta_n$とし、$\theta_1,\theta_2,...,\theta_n$の合成置換θをGに現れる変数に制限して考え,ゴールGに現れる変数にθを施し、その正規項を解とする。すなわち、$x\theta\downarrow$(ただし、$x\in Var(G),\theta=\theta_1,\theta_2,...,\theta_n|Var(G)$)を解となる。

この解を求める手続きの最後に正規項を求める理由は、例えばFact(s(s(0)))=*Ans.の解に、論理的には、*Ans=Fact(s(s(0)))も含まれるが、通常我々が答えとして欲しいものは、Fact(s(s(0)))を計算した(正規化した)*Ans=s(s(0))であるからである。

3 融合型言語の処理系の設計と実現

3.1 はじめに

本章では実際に我々が製作したシステムについて述べるが、まず上で与えたオペレショルセマンテクスと実際に作成したシステムと異なる部分について述べます。それは処理効率を向上するための言語に設けた制限と、先に示したオペレーショナルセマンテクスが持つ非決定性をシーケンシャルマシンで実行するための戦略について議論します。次にその制限と戦略の基で作成した処理系のソフトウエアデザインについて示します。その特徴は4本のスタックを使った構造である言うことが出来るでしょう。そして最後に処理速度と、支援環境の一つであるトレーサ用の実行モデルについて示します。

3.2 言語系の制限

本言語系はPROLOG以上の複雑な実行メカニズムを持っているために、PROLOG以上に計算機にとって重くなることが容易に予想される。そこで、処理速度の向上のために、言語系に制限を付けた。その制限は以下の点である。

(Ⅰ)関数記号と構成子記号をシンタックスで区別する

(II)ルール節の制限

ルール節を次のようなSyntaxに制限をする

$F(t_1,....,t_n)=R\leftarrow M_1=N_1,....,Mq=Nq.$

ただし、Fは関数記号、t_iはアトム又は変数又は構成子のみからなるターム、R,M_i,N_iはターム

これは次節でのOpt-inner-most戦略のために必要な制限である。

(III) Goal Superposantの制限

ゴール中の変数は任意のアトム、項とユニファイ可能であるので、変数それ自身を部分項と見て、すべてのルールとのSuperpositionが可能である。しかし、本システムでは変数を部分項と見てのsuperposeを禁止している。

3.3 戦略

Superpositionを行う時に部分項をどの様な順番で評価するかについて議論する。この順番により、Outer-most,Inner-most,Opt-Inner-most等の戦略がある。

Outer-most,Inner-mostは、それぞれ、最外、最内のリデックス(書き換え可能な項)を選び計算を進める戦略である。複数の候補がある場合には、本システムでは最左の項より実行を試みる。

Opt-Inner-mostは、Inner-mostより計算能力が落ちるものの(Inner-mostで解が得られるのに、Opt-Inner-mostで解が得られない問題がある)、計算時間、スタック使用量に関して効率の向上がある戦略である。その方法は、最左最内の関数部分項を選び(これは制限(Ⅰ)によって関数と構成子の区別ができるためにできる)、その項に対して、ユニファイを行い、その項の書き換えができない時には別の項を選ばずに、Superpositionの失敗とする方法である。このことは、関数の計算の立場から見れば、「最左最内の関数が未定義ならば、全体の関数の計算をしない」ことに相当する。本処理系で

は現在、Outer-most、Opt-Inner-mostの戦略が実現さてお り、現在Inner-most戦略を開発中である。

3.4 全体構成

システムの全体構成を示すと図1のようになる。

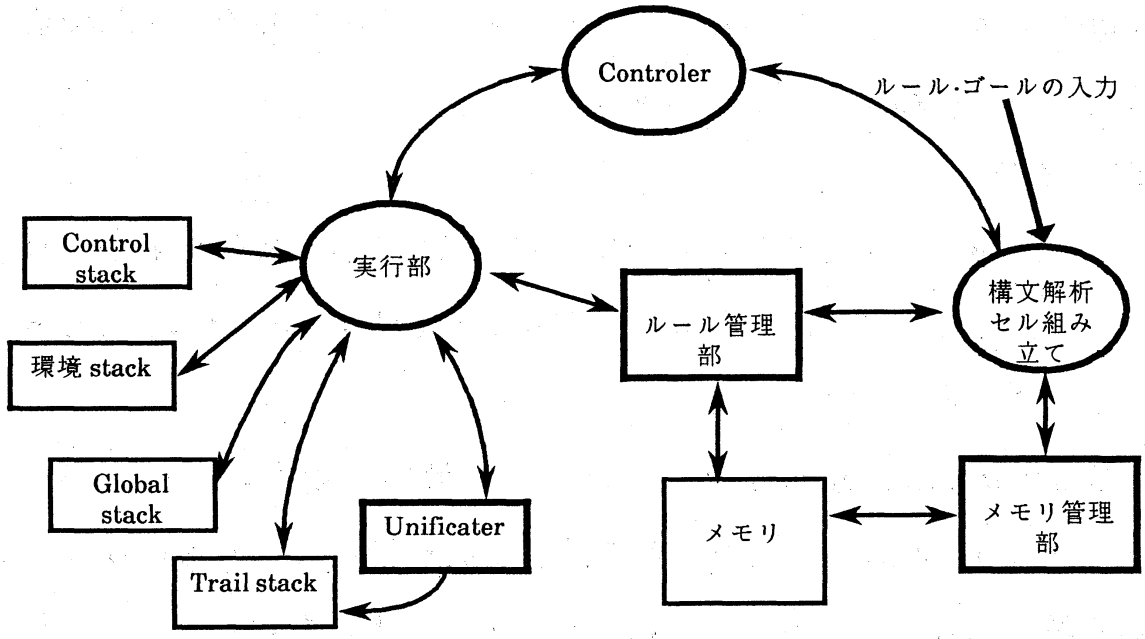

図1 システムの全体構成

3.5 構文解析·セル組み立て部

構文解析部で受理されたルールやゴールはメモリにダグを付きの二進木セル構造で格納される。

3.6 実行部

計算を実行する上で必要な主な機能は次の4点である。

1.Derivation

正しく等式、部分項、ルールを選んで計算実行をしてゆく機能

2.Backtrack

計算が失敗したときに、ある時点まで計算の過程を戻り、別の選択をする機能。PROLOGと同じ。

3.Work memory

本システムは構造共有方式でなくコピー方式を採用しているので、コピーしたものを格納する領域

4.環境の制御

本言語系では、異なるルールに現れる変数は、同じ名前でも、同じ変数ではない。そのため、ある変数の値を評価する時に、どの環境の変数であるかが問題になる。その環境の切り替え等を制御する必要がある。

これらの機能を4本のスタックを使って実現をした。

(Ⅰ) Global stack:機能3を実現する。そのためメモリと同じデータ構造を採る。

(II) Control stack:計算の実行環境を格納していて、機能の1,2のために使われる。計算が一ステップ実行される毎にpushされる。Backtrack時にpopし環境を元の状態に戻す。

(III) Trail stack:機能2のうち、変数の束縛、項の書き換えに関することのみ扱う。PROLOG処理系のTrail stackと異なり項の書き換え情報も格納している。

(Ⅳ) 環境stack:機能4を実現している。

3.7 計算時間

処理系の性能の一つである、計算時間について示す。計算時間はプログラムにも依存するので大まかな目安として欲しい。

問題	Outer-most	Opt-Inner-most
5の階乗	0.4	15.9
ソート(10ヶ)	0.3	0.9

計算時間(秒) 東芝AS-3300

参考までに、本システムとC言語で以前私が作ったPROLOGインタープリタと比べると次のようになる。

問題	本システム(Opt-inner-most)	PROLOG
6の階乗	8.5	1.7

計算時間(秒) ソードM685

3.8 支援環境

デバッグの方法は大きく二つのアプローチがある。一つはユーザーに実行過程を示すことによる方法(トレーサー)、もう一つは、本言語のように数学的基盤のあるものに限られるが、その数学的性質を使ってユーザと会話をしながら半自動的にデバッグを行う方法である。ここでは前者のアプローチを取る。そのアプローチにおいては、本言語系のように実行メカニズムが複雑な場合には、そののまま実行の過程を出力したのではユーザは実行の流れの把握は困難であり、如何にトレース情報を整理して分かりやすいかたちで出力するかが問題点となる。そこで実行過程を次の三種類に分類し、それに基づきトレース情報を出力する方法を提案する。また現在これに基づいたトレーサを製作している。その分類は次の通りである。

(I)等式列を証明する系列

(II)関数の計算をする系列

(III)部分項やルールを選び出しユニフィケーションをする。

4 まとめ

まず、等式論理をベースとした融合型言語について簡単に紹介をした。詳細については文献[3]を参照されたい。次に、実際に処理系を実現するために(1)言語系の制限 (2)部分項の選択法 を明確にし、その上でC言語を使って実行機構が具備すべき4つの機能を4つのスタックをシミュレーションすることにより実現した。さらに、ここでは詳しく述べなかったが本処理系はコピー方式を採用、トレールスタックにPROLOG処理系で見られた変数の束縛情報だけでなく、項の書き換え情報も蓄えた、などの特徴を持つ。

また、今後の課題として、

-戦略として、現在Inner-mostの実現とユーザの指定により評価法を変えられるメカニズム

-プログランミング支援環境･デバッグ環境の整備

-組込み関数の充実

を、考えている。

参考文献

[1] 山本 他:“逐次型推論マシン”,第31回記号処理研究会誌、(1984)

[2] H. Nakazima,“Prolog/KR Implementation Notes”,Feb. 1983,Department of mathematical engineering and instrumentation physics,University of Tokyo

[3] A.Togashi,S.Noguchi,“On a Deductive Basic for Equational Logic Programs”,1986,Research Institute of Electrical Communication,Touhoku University