知識工学と人工知能 58 － 4
（１９８８．５．１１）

高階節形式論理を用いた項書換え型計算系と論理型計算系の融合

三浦欽也，堂下修司
京都大学　情報工学教室

本稿では，高階節形式論理の枠組みの中で，単項述語変数を用いて，論理型計算系と，関数型計算系の一種である項書換え型計算系との融合を試みる．これは，１つのプログラム中で，用途に応じて双方のスタイルのプログラミングを行えることを意味し，各々の長所を生かしたプログラミングが可能になる．

本稿の方法の特徴は，通常の第一階述語論理による論理型計算系を完全に含んだ拡張になっていることと，プログラムが高階節形式論理の論理式となるため，理論的な明瞭性が得られることである．

また本稿では，双方のスタイルが混在するプログラム中での，サブプログラム間のインターフェイスについても，考察する．

Amalgamation of Term Rewriting System and Logic Programming on Higher-Order Clausal Logic

Kinya MIURA and Shuji DOSHITA
Depertment of Information Science, Kyoto University
Yoshidahonmachi, Sakyo, Kyoto, 606 Japan

In this paper, we amalgamate logic programming system and term rewriting system in higher-order clausal logic, using monic predicate variables. It means that you can use both programming styles in one program. Therefore, you can obtain benefits from both programming systems.

This system fully includes the standard logic programming system based on first-order predicate logic. In this system, programs are logical expressions of higher-order clausal logic. Therefore, there is theoretical clarity in this system.

In this paper, moreover, we consider the interactions between both programming styles.

1. はじめに

近年，Prologを代表とする論理型計算系が多くの局面で利用されている．論理型計算系の特徴には，入出力が不分明であること，強力なパターンマッチの能力を持つこと，バックトラックによる非決定的な実行等があり，さまざまな局面で有効であると思われる．一方，プログラムの入出力がはっきりしている場合や，非決定的な動作が不用であるような場合は，特に論理型計算系を用いる必然性は少なく，むしろ関数型計算系等を用いる方が，実行効率上でも有利であると思われる．

本稿では，論理型計算系と関数型計算系の双方の利点を生かしたプログラミングが可能になるような計算の枠組みを提案する．本稿で提案する方法は，高階節形式論理[1][2]を用いて，論理型計算系の中に，関数型計算系の一種である項書換え型計算系を取り入れるもので，第一階述語論理による論理型計算系に単項述語変数を付け加えたものになっている．また，この方法におけるプログラムは，高階節形式論理の論理式となるため，プログラムに関する演繹を，自然な形で行えることが期待できる．

以下，２．では高階節形式論理について簡単に述べ，その部分系として，３．で，本稿で提案する計算系を示す．４．では，項書換え系の表現と実行について具体的に述べ，単一化の問題と簡約戦略の関係について触れる．５．では，本稿の計算系における，論理型の計算機構と項書換え型の計算機構の間のインターフェースについて考察する．６．はまとめである．

2. 高階節形式論理

論理型計算系を高階論理に拡張する試みは，これまでにさまざまなものがあるが[3][4]，著者らも高階節形式論理という形で研究を行ってきた[1][2]．

高階節形式論理とは，原子式として型付きのλ式を用いた節集合による論理であり，一般にはωオーダーの論理である．この高階節形式論理においては，等号に関する公理を付け加えることで，導出原理がある種の完全性を持つことが示され，第一階述語論理を拡張したある種の計算系と見なすことができる．[2]

ところが，このような高階論理による論理型計算系において，特に導出原理を用いる場合，高階の項（型付きのλ式）の単一化を行う必要があるが，高階の項における単一化の手続きは，一般には停止性が保証されず，また，最汎単一化作用素（ｍｇｕ）が有限とは限らないという問題点があり，このことが，高階論理による論理型計算系を実現する上での大きな問題点となっている．

そこで，目的を論理型計算系と関数型計算系の融合という点に絞り，一般の高階節形式論理ではなく，目的を満たす程度に適当な制約を加えた論理を扱うことで，単一化を簡単にし，より実用的な計算系を構成することが考えられる．本稿では，そのために，通常の論理型計算系（第一階ホーン節集合）に，単項述語変数のみを高階の要素として付け加えたものを提案する．

3. 単項述語変数を含んだ論理型計算系Ｌ

この節では，第一階述語論理による論理型計算系（ホーン節集合）に，単項述語変数を付け加えた計算系Ｌを定義する．まず，Ｌの文法を以下のように定義する．

［記号］
個体変数：X，Y，Z，W，…
関数定数：a，b，c，…，f，g，…
述語定数：p，q，r，…
単項述語変数：P，Q，R，…

なお，便宜上，変数は大文字で，定数は小文字で表記するとする．

［項］
1）個体変数は項である．
2）fが関数定数，$t_1 \cdots t_n$が項ならば，$f(t_1,\cdots,t_n)$は項である．（$n \geqq 0$）

［原子式］
1）pが述語定数，$t_1 \cdots t_n$が項ならば，$p(t_1,\cdots,t_n)$は原子式である．（$n \geqq 0$）
2）Pが単項述語変数，tが項ならば，P(t)は原子式である．

［節］
1）A_0が原子式なら，A_0:-.は節である．（宣言節）
2）$A_0,A_1 \cdots A_n$が原子式なら，A_0:-$A_1,\cdots,A_n$.は節である．（手続き節）
3）$A_1 \cdots A_n$が原子式なら，:-$A_1,\cdots,A_n$.は節である．（ゴール節）

［プログラム］
プログラムは，宣言節と手続き節の有限集合である．

プログラムの例を以下に示す．

［例］
```
P(a(z,Y)):-P(Y).
P(a(s(X),Y)):-P(s(a(X,Y))).
e(X,X):-.
m(z,Y,z):-.
m(s(X),Y,Z):-
    m(X,Y,W),e(Z,a(Y,W)).
```

このプログラムの実行は，通常の第一階述語論理による論理型計算系と同様に，あるゴール節を頂上節とする深さ優先の線形入力演繹によって，反駁を探索することによってなされる．

このとき，Lが通常の論理型計算系と異なるのは，単項述語変数が現れた場合の単一化であるが，この場合は，一般の高階論理における単一化ほど複雑ではなく，以下のように考えることができる．

［単一化手続き］
単一化するべき二つの原子式をA,Bとする．
1）A,Bのいずれにも述語変数が現れない場合，通常の（第一階の）単一化と同様．
2）AがP(t)の形をしているとき，tと単一化可能な任意のBの真部分項sに対して，s,tのmguと，{P←λW.B[s/W]}との合成がA,Bのmguとなる．（B[s/W]は，B中の部分項sをWで置き換えたもの）
また，{P←λW.B}もmguとなる（自明なmguと呼ぶ）
3）BがP(t)の形をしているとき，2）に準じる．

任意の原子式は，大きさが有限であり，真部分項の数も有限であるから，mguが複数になることがあっても，この手続きは必ず停止する．

単項述語変数を含む原子式を単一化した結果として複数のmguが得られた場合，どのmguを採用するか（あるいはしないか）は，プログラムの実行全体に大きな影響を与えるが，項書換え型計算を行う場合は，自明なmguは通常無視した方がよく，どのmguを採用するかは，簡約の戦略に深く関わってくる．このことの詳細は次節で触れることにする．

4．Lにおける項書換え型計算の実現

項書換え型の計算は，書換え規則の集合として抽象化されるが，Lにおいては，これを単項述語変数を用いた手続き節の形で表現する．

［例］

書換え規則：

```
{ a(z,Y)→Y,
  a(s(X),Y)→s(a(X,Y)) }
```

Lのプログラム：

```
P(a(z,Y)):-P(Y).
P(a(s(X),Y)):-P(s(a(X,Y))).
```

この例は，zをゼロ，sを後者関数と見なすと，加算を表すプログラムになっている．

これを実行するには，

```
e(X,X):-.
```

という定義を付け加えて，例えば，

```
:-e(X,a(s(s(z)),s(s(s(z))))).
```

をゴール節として，反駁が得られれば，変数Xに解が得られる．

［実行例］

```
1: P(a(z,Y)):-P(Y).
2: P(a(s(X),Y)):-P(s(a(X,Y))).
3: e(X,X):-.
4: :-e(X,a(s(s(z)),s(s(s(z))))).
5: :-e(X,s(a(s(z),s(s(s(z)))))).
   (from 2,4 with
    { P← λ W.e(X,W),
      X←s(z),
      Y←s(s(s(z))) } )
6: :-e(X,s(s(a(z,s(s(s(z))))))).
   (from 2,5 with
    { P← λ W.e(X,s(W)),
      X←z,
      Y←s(s(s(z))) } )
7: :-e(X,s(s(s(s(s(z)))))).
   (from 1,6 with
    { P← λ W.e(X,s(s(W))),
      Y←s(s(s(z))) } )
8: □
   (from 3,7 with
    { X←s(s(s(s(z)))) } )
```

このような実行の過程では，単項述語変数を含んだ原子式の単一化手続きが行われるが（上の例の，5,6,7を導くところ），このとき，自明なmguを用いると，新しく得られるゴール節は，直前と同じものになってしまう．従って，単項述語変数を項書換え型の計算に用いる場合には，自明なmguは，なんら計算に寄与しない上に，無用な手間を増加させ，無限ループに陥る危険性を持つがゆえに，無視する方がよいと思われる．

また，上の例では，自明でないmguはたまたま1つしかないが，これが複数あったとき，単項述語変数の引数と対応付られる部分項をどう選ぶかが，項書換え型計算の簡約化戦略に対応する．従って，常に最左最外のものを選ぶならば，もとの書換え規則の集合が停止性を持つかぎり，Lにおける計算も停止する．また，もとの書換え規則の集合が合流性を持つならば，あるmguで失敗しても，バックトラックして他のmguを試す必要はない．

5．Lにおける論理型計算と項書換え型計算の融合

Lは通常の第一階述語論理による論理型計算系を包含しており，また，前節でみたように，項書換え型の計算も行うことができる．しかしながら，これら2種類の計算が並存して，互いに呼び出し合う方法は必ずしも定かではない．本節では，主に，この方法について考察する．

まず，論理型の計算から項書換え型の計算を呼び出す場合を考察すると，

いくつかの問題点が存在する.

まず第１の問題点は，前節のような形で項の書換えを行う場合，それは常にゴール節の上で行われるため，手続き節の頭部の書換えができないことである．ゆえに，例えば，

```
P(a(z,Y)):-P(Y).
P(a(s(X),Y)):-P(s(a(X,Y))).
m(z,Y,z):-.
m(s(X),Y,a(Y,W)):-m(X,Y,W).
```

というようなプログラムの場合，最後の節の部分項a(Y,W)は決して書き換えられないことになる．この問題を解決するためには，新しい変数Zと前節のeという述語を用いて，最後の節を

```
m(s(X),Y,Z):-
    m(X,Y,W),e(Z,a(Y,W)).
```

のように直せばよい．なお，このプログラムはzをゼロ，sを後者関数とすると，加算を表す関数としてaを定義し，乗算を表す述語としてmを定義しているとみなすことができる.

また，第２の問題点は，書換え規則が適用されるとき，即ち，書換え規則に対応した手続き節によって導出が行われるとき，書き換えられる側に変数が含まれている場合，不都合が生じることがあるということである．これは，書換え規則が適用できるように，その変数を適当な値に置換するときに生じる．このような場合，ゴール中で同じ変数を共有する他の原子式で導出が行われるときに，その変数に既に値が与えられていることにより，単一化に失敗する可能性が生じるからである．理論的には，このような場合には，バックトラックによって最終的に解にたどり着くことが予想されるが，実際的には実行効率の低下は否めない．このような場合にも，適当な変数と述語eを用いることで，項の書換えを，変数が十分に置換されるまで遅らせて，この効率の低下を防ぐことができる.

次に，項書換え型の計算から論理型の計算を呼び出すことを考える．これは，書換え規則に対応する手続き節の中に，適当にサブゴールを埋め込むことによって可能になる．例えば，先の例に加えて，

```
P(f(z)):-P(s(z)).
P(f(s(X))):-m(s(X),f(X),Z),P(Z).
```

なる節を持つプログラムを考えると，これは，乗算の述語mを使って階乗の関数fを定義していると見なすことができる．この場合でも，項の書換えを適当な時期まで遅らせるために，述語eを用いてもよい.

また，同じような形式で，いわば条件付き書換え規則と言うようなものも定義することができる．例えば，

```
R(b(X)):-g(X),R(c(X)).
R(b(X)):-R(d(X)).
```

という節は，b(X)という形の項に対して，g(X)が真ならばc(X)に書換え，偽ならばd(X)に書き換えるような規則であると考えられる.

６．おわりに

前節までに見てきたように，第一階述語論理による論理型計算系に単項述語変数を付け加えることにより，項書換え型の計算も同じ枠組みで実行できる．このときの問題点は，単項述語変数を含む原子式の単一化である．厳密には，得られる全てのmguについて，（バックトラックによって）導出を行

うべきであるが，項書換え型の計算系を実現するのが目的であるならば，自明なmguは排除するのが望ましい．また，合流性を仮定して，複数のmguのうち，ただ一つのものを選んで，他は捨ててしまうのが，実行効率の点で有利であると思われる．この場合，どういう方針でmguを選ぶかが，項書換え型計算系の簡約化戦略に対応する．正規な戦略とするためには，最左最外簡約に対応するようにmguを選べばよいが，実行効率を優先して，最内簡約とする考え方も可能である．いずれを用いるか，あるいは切り替えられるようにするかは，目的に応じて定めればよい．

今後の課題としては，この体系Lにおけるプログラムは，高階（２階）節形式論理の論理式となるので，この高階節形式論理の上で，プログラムの検証や，等価性の判定などを，論理的な演繹によって行う方法を模索することである．

また，単項述語変数を，この稿で行ったような定型的な用い方でのみ用いる場合には，ある種のコンパイルによって実行効率を向上させることも可能ではないかと思われる．

参考文献

[1] 三浦，他：高階節形式論理におけるエルブラン解釈について，第１回人工知能学会全国大会論文集，(1987)，33-36．

[2] 三浦，他：等号を含む高階節形式論理における完全性について，情報処理学会第35回全国大会講演論文集，(1987)，71-72．

[3] Pietrzykowski, T., et al: A Complete Mechanization of (ω)-order Type Theory, ACM national conf., 1(1972), 82-92.

[4] Miller, D.A., et al: Higher-Order Logic Programming, Lecture Notes in Computer Science, 225, Springer-Verlag, (1986), 448-462.