項書換えシステムに基づく自己反映計算の実現*

1G－2

沼澤 政信†　栗原 正仁†　大内 東†

北海道大学工学部‡

1 はじめに

自己反映計算(reflection)とは，計算システムが「自分自身」に関するいわゆるメタ的な情報を参照したり変更したりすることを含むような計算である．

歴史的にはこの試みは関数型言語，論理型言語，オブジェクト指向言語などをベースとして行われてきた．しかし，書換え型の計算モデルをベースとする言語である項書換えシステムに自己反映計算を導入する試みは現在のところ提案されていない．

本稿では，項書換えシステムに自己反映計算機能を導入し，REPS (Reflective Equational Programming System) という言語の枠組みを提案し，現在構築中のREPSの応用例を示す．

REPSは，計算の実行環境の一部であるメタオブジェクトをユーザプログラムからアクセス可能なデータに変換するreification，およびデータをメタオブジェクトに変換して自らの実行環境を変更するdeificationを基本機能として提供している．

2 REPSにおける自己反映計算

2.1 REPSの構文

関数記号の集合を$\mathcal{F}$，変数記号の集合を$\mathcal{V}$とする．各関数記号$F \in \mathcal{F}$には項数(arity)と呼ばれる非負整数が付随している．項数0の関数記号を定数記号という．

項の集合$\mathcal{T}(\mathcal{F},\mathcal{V})$は次の条件を満たす最小集合である．

- $x \in \mathcal{V}$ならば，$x \in \mathcal{T}(\mathcal{F},\mathcal{V})$
- $f \in \mathcal{F}$が項数nの関数記号，
  $t_1, t_2, \ldots, t_n \in \mathcal{T}(\mathcal{F},\mathcal{V})$が項ならば，
  $f(t_1, t_2, \ldots, t_n) \in \mathcal{T}(\mathcal{F},\mathcal{V})$

次に，自己反映計算を引き起こすための構文を導入する．t, c, $r \in \mathcal{T}$が項のとき，その3組$t : c : r$を自己反映式という．自己反映式は項ではない．項が書換えの対象になるのに対し，自己反映式はそうならない．自己反映式は書換え規則の左辺または右辺，すなわちプログラムの一部としてのみ現われる表現であり，システムの自己反映機能とのインタフェースの役割を果たす．一方，項も通常の項書換えシステムと同様に，書換え規則の左辺または右辺を構成できる．そのため，便宜上，項と自己反映式を総称して式と呼ぶ．

$e \to e'$ *if* $t_1 = t'_1, \ldots, t_n = t'_n$. を書換え規則，式$e, e', (t_1 = t'_1, \ldots, t_n = t'_n)$をそれぞれその規則の左辺，右辺，条件という．ただし，書換え規則には次のような構文的な制限が課せられる．

- 左辺は変数ではない．
- 右辺に出現する変数は必ず左辺にも出現する．

書換え規則の形は，左辺，右辺がそれぞれ項であるか自己反映式であるかによって，4つに分類できる．左辺が自己反映式である書換え規則はreification，右辺が自己反映式である書換え規則はdeificationという自己反映計算特有の操作を引き起こすために用いられる．

2.2 reificationとdeification

reificationはメタレベル・オブジェクトをベースレベル・オブジェクトに変換するプロセスである．項，文脈，プログラムは計算の実行環境の一部(すなわち，メタレベル・オブジェクト)であり，通常はユーザプログラムからデータとしてアクセスすることはできない．各関数により，メタレベル・オブジェクトはユーザプ

---

*Implimentation of Reflective Computation Based on Term Rewriting Systems

†Masanobu Numazawa, Masahito Kurihara and Azuma Ohuchi

‡Faculty of Engineering, Hokkaido University

ログラムがアクセスできるデータに変換される．要するに，reification は計算の実行環境をベースレベルからメタレベルへ移行させる操作である．あるいは，相対的に，別な見方をすれば，メタレベルの実行環境をベースレベルでアクセスできるデータに変換する操作である．こうして得られたデータを，それぞれ，項，文脈，プログラムのメタ表現という．

deification は，reification の逆操作で，項，文脈，プログラムの (適切な) メタ表現をそれぞれメタレベル・オブジェクトである項，文脈，プログラムに変換するプロセスである．メタ表現が「適切な」データでないときは，この変換は定義されない．

3 REPS の応用

外山 [1]，山田 [2] のメンバーシップ条件付き項書換えシステムにおいては，各書換え規則にメンバーシップ条件と呼ばれる条件を付けることができる．ただし，メンバーシップ条件は一般にメタレベルの情報を必要とするため書換え規則では記述できず，実現は，メンバーシップ条件を判定するプログラムをユーザが Lisp で記述するようになっていた．メタレベルの情報を扱える REPS においては，このメンバーシップ条件を書換え規則で記述できるように拡張した．つまり，条件付き書換え規則が扱え，その条件部分にメタ情報も与えられるようにした．

REPS はプログラム (書換え規則) の操作が可能であり，書換え規則の追加，削除によりプログラムを変更できる．その REPS の能力に注目し，Leler[3] の提案する augmented term rewriting system (ATRS) に基づいた制約プログラミング言語処理系を作成できる．ATRS の特徴の一つは，大域的な「変数」(項書換えシステムの構文規則からすれば「定数記号」) に値を束縛できることである．x に 2 を束縛することはプログラムに動的に書換え規則 $x \to 2$ を付加することと等価であるので，REPS のプログラム変更操作で簡単に実現できる．

4 REPS の効率化

現在の REPS の問題点は，実用的な面から言えば，メタオブジェクトとそれに対応する項 (メタ表現) との間の変換が主要なオーバーヘッドとなってあまり効率の良い処理系とはいえないことである．そこで，その問題点を解決するために，現在，遅延評価 (delayed evaluation) の考え方を導入し効率化を試みている．

reification 操作は，項，文脈，プログラムをそれぞれに対応する項 (メタ表現) に変換する．このとき，与えられた項やプログラムが大きければ大きいほどその変換時間は長く，後に deification 操作が行われない時やそれが行われてもそのデータにアクセスがなければその変換は無駄になってしまう．そこで，メタ表現への変換要求があった場合は，即座に変換せず，「これは本当はメタ表現である」ということがわかるような情報を持たせておき，その (本当はメタ表現である) 項へのアクセス要求があった場合にのみ，その要求に応じて一部分をメタ表現に変換するという方法で無駄な変換を避けるようにする．

本稿では，REPS を CLOS(Common Lisp Object System) を用いて構築しており，この方法は CLOS のクラス (class) 定義とメソッド定義によって実現することが可能である．

5 おわりに

本稿では，項書換えシステムに自己反映計算機能を導入し，REPS (Reflective Equational Programming System) という言語の枠組みを提案し，現在構築中の REPS の応用例を示した．また，自己反映計算特有のメタ表現への変換の効率改善を試みた．

今後は，REPS の言語としての実用性を高めるために，更に効率改善や機能向上の再検討を行う．

参考文献

[1] Toyama, Y.: Confluent term rewriting systems with membership conditions, *Proc. of Intern. Workshop on Conditional Term Rewriting Systems, Lecture Notes in Computer Science*, Vol.308, 1987, pp.228–241.

[2] Yamada, J.: Confluence of terminating membership conditional TRS, *Proc. of Intern. Workshop on Conditional Term Rewriting Systems, Lecture Notes in Computer Science*, Vol.656, 1992, pp.378–392.

[3] Leler, W.: *Constraint Programming Languages: Their Specification and Generation*, Addison-Wesley, 1988.

[4] 佐藤，栗原，大内：自己反映計算機能をもつ等式プログラム処理系の実現，信学技報 COMP92-92, pp69-76, 1993-03.