ソフトウェア自動合成シェルSOFTEX/S(3)

6F−3

−項書換えシステムの直接実行方式とその高速化−

友部 実　山之内徹　渡辺正信
NEC C&Cシステム研究所

1 はじめに

項書換えシステムは関数型言語や等式仕様の実行系、また定理自動証明系などにおける重要な計算モデルとなっている。ソフトウェア自動合成シェルSOFTEX/Sは、プログラム変換を項書換えシステムに基づく計算モデルで実現している。プログラム変換システムに与えられた入力仕様は項書換え規則によって書き換えが行なわれ、最終的にプログラムに変換される。本稿では、項書換え規則を従来LISP等で実現されていた間接実行方式[2, 3]と異なる、項書換えシステムに特化したデータ構造を持つ直接実行可能なCプログラムにコンパイルし実行する処理系の開発を行なった。直接実行方式による実現の結果、従来の実現に比べ、項書換えシステムに特化したデータ構造の記述を行なうことができ、高速に項書換えシステムを実行することが可能になると共に、実行時の高速化手法を用いることにより、更に大幅に書き換え速度を向上することができた。以下SOFTEX/Sシステムの基盤となる多ソート項書換えシステムについて述べ、項書換え規則を各関数記号毎にC言語の関数にコンパイルする書き換え規則コンパイラについて説明する。また書き換え実行時の高速化手法として、書き換え対象の中から書き換え可能な部分を選択することにより高速化をおこなう構成子マーキング手法について検討・評価を行なったので報告する。

2 多ソート項書換え系

2.1 多ソート項書換えシステム

定義2.1(多ソート指標) 多ソート指標Σとはソート記号集合Sと関数記号集合Fの組$\Sigma \stackrel{def}{=} < S, F >$である。ただし、$\forall f \in F; f_{s_1 \ldots s_n s}; n \geq 0; s_1, \ldots s_n, s \in S$である。ここで、ソート列$s_1 \ldots s_n$は関数記号$f$の定義域、$s$は$f$の値域を示す。□

定義2.2(変数) 変数集合Vは、ソート記号$s \in S$を持つ変数v_sの集合である。□

定義2.3(多ソート項) 多ソート指標を$\Sigma =< S, F >$、変数集合をVとするとき、ソートsの多ソート項集合$T_s(\Sigma, V)$は以下のように定義される。:

1. $v_s \in V, s \in S$ならば$v_s \in T_s(\Sigma, V)$
2. $f_{s_1 \ldots s_n s} \in F; t_1 \in T_{s_1}(\Sigma, V), \ldots, t_n \in T_{s_n}(\Sigma, V)$ならば$f(t_1, \ldots, t_n) \in T_s(\Sigma, V)$

多ソート項集合$T(\Sigma, V)$とは$T(\Sigma, V) \stackrel{def}{=} \cup_{s \in S} T_s(\Sigma, V)$である。□

定義2.4(多ソート書換え規則) 多ソート項集合を$T(\Sigma, V)$とし$l, r \in T(\Sigma, V)$かつrに現れる変数はlに現れるとするとき、$l \rightarrow r$を多ソート書換え規則という。□

定義2.5(構成子項) 多ソート書換え規則集合$R = \{l_i \rightarrow r_i \mid l_i, r_i \in T(\Sigma, V), i = 1, \ldots, n\}$があったとき、各書換え規則の$lhs$が$l_i = f_i(t_{i_1}, \ldots, t_{i_n})$で表されるとする。このとき、$\Sigma_c =< F - \{f_i \mid i = 1, \ldots, n\}, S >$とし、このような多ソート指標$\Sigma_c$から構成される多ソート項集合$T_c(\Sigma_c, \phi)$の要素を構成子項と呼ぶ。□

2.2 多ソート書換え規則の操作的意味

多ソート項集合上の書換え関係$\Rightarrow$を与えることによって多ソート書換え規則の操作的意味を与える。

定義2.6(文脈) 項$t = f(\ldots)$が、n個の特別な$\Box_1, \ldots, \Box_n \notin T(\Sigma, V)$を部分項の位置に持つ時、$t$を文脈(*context*)といい、$f[\Box_1, \ldots, \Box_n]$と表す。$n = 1$のときは添字を省略することがある。また$\Box_1, \ldots, \Box_n$をそれぞれ項$t_1, \ldots, t_n$で置き換えて得られる項を$f[t_1, \ldots, t_n]$と表す。□

定義2.7(多ソート置換) 置換$\sigma = \{v_1/t_1, \ldots, v_n/t_n\}$において、$v_i \in V; t_i \in T(\Sigma, V)$かつ$\forall s_v \in Sort(v_i); \exists s_t \in Sort(s_i); s_v = s_t$(但し$i = 1, \ldots, n$)であるとき、$\sigma$を多ソート置換という。ここで$Sort(t)$は項$t$の値域ソート記号集合である。また$t = f[v_1, \ldots, v_n]$のとき$t\sigma$を項$t$への多ソート置換$\sigma$の適用と呼び、$t\sigma = f[t_1, \ldots, t_n]$である。□

定義2.8(多ソート書換え関係) $t_i[x], t_j \in T(\Sigma, \phi)$と多ソート書換え規則$l \rightarrow r$と最汎多ソート単一化置換$\sigma$が存在し、$l\sigma = x\sigma; t_j = t_i[r\sigma]$となる時、その時に限り、$t_i[x]$と$t_j$は多ソート書換え関係にあるといい、$t_i[x] \Rightarrow t_j$または$t_i[x] \Rightarrow_{l \rightarrow r} t_j$と表す。□

3 書換え規則コンパイラ

図1にSOFTEX/Sシステムの概念図を示す。文脈自由言語によって記述された入力仕様はパーザによって構文解析が行なわれ、多ソート項集合$T(\Sigma, \phi)$にマッピングされる[1]。プログラム変換は項書換え規則コンパライラによってコンパイルされた多ソート項書換え規則によって行なわれ、変換終了後、プリンタによって多ソート項集合$T(\Sigma, \phi)$は再び文脈自由言語として出力される。

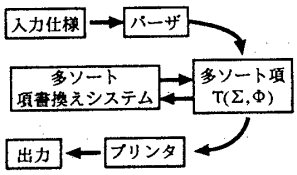


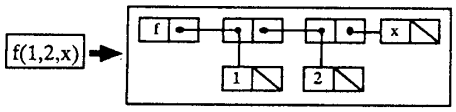
図1: システム概念図

3.1 コンパイラの概要

多ソート項書換え規則は書換え規則コンパイラを用いて、左辺の最外の関数記号毎にCの関数としてコンパイルされ

Software Synthesis Shell SOFTEX/S(3)
−Direct and Efficient Implementation of TRS−
Minoru TOMOBE, Toru YAMANOUCHI, Masanobu WATANABE
NEC Corporation

る。SOFTEX/S内部では項表現は図2に示すような構造体として表現されている。SOFTEX/Sでは多ソート書換え規則による書換え対象を基底項（変数をもたない項）に限定しているため、最汎多ソート単一化置換σは単純なパターンマッチングとして実現することができる。図3に書換え規則とコンパイルされたコードの例を示す。書換え規則の1行目～4行目まで多ソート指標の定義、5行目～7行目が書換え規則の定義となっている。コンパイルされた書換え規則はif文の条件部として展開され(4行目～10行目)、項表現のデータ構造のパターンマッチを行なう。変数への値の代入は実行時にソートの検査を行なった後(8行目)、実行される。パターンマッチに成功した場合、if文の本体に記述された項書換え規則のrhsに表された項表現と等価なデータ構造を生成する関数呼び出し(11行目)が実行され、書換えが行なわれる。




```
typedef struct term {
  union {
    unsigned int functor;
    struct term *car;
  } u;
  struct term *cdr;
  unsigned int flag : 1;
  unsigned int type : 3;
} term;
```

図 2: 項表現の内部形式

書換え規則

```
 1: sortdef int;
 2: termdef 'int('int) s;
 3: termdef 'int('int,'int) plus;
 4: vardef 'int x,y;
 5: relation r_1 eql (
 6: plus(s(x),y),s(plus(x,y))
 7: )
```

Cにコンパイルされた書換え規則

```
 1: int R_test_plus(term **spec,term **addr) {
 2:   term *var_121 /* y */ ,*var_120 /* x */ ;
 3:   term *tmp_0,*tmp_1,*tmp_2,*tmp_3;
 4:   if ((tmp_0 = *spec) &&
 5:      (tmp_1 = tmp_0->u.car) &&
 6:      (tmp_1->u.functor == 115) && /* 115 = 's'*/
 7:      (tmp_2 = tmp_1->cdr) &&
 8:      CheckBind(var_120,5234,tmp_2) && /*5234 = 'int'*/
 9:      (tmp_3 = tmp_0->cdr) &&
10:      CheckBind(var_121,5234,tmp_3)) {
11:      *addr = make_cell(115,1, /* 8387 = 'plus' */
             make_cell(8387,1,cons(var_120,var_121)));
12:       if(GCflag) gc(Spec);
13:       return(TRUE); }
14: return(NULL); };
```

図 3: コンパイルされたコードの例

3.2 実行時処理系

実行時処理系は、変換の前処理として後述する高速化処理部を呼び出す。実行時処理系は書換え対象の中から指定された戦略（現在は最左最外戦略と最左最内戦略の2種類）に基づいて、書換え対象の中からリデックスを選択し、選択されたリデックスの関数記号に相当する書換え規則を呼び出す。書換え規則はあらかじめC関数としてコンパイルされており、リデックスと単一化可能な場合、単一化をおこない、これを書き換える。最左最外戦略の場合、書換えが成功した後、書き換え対象の最左最外部から再びリデックスの選択を行ない、書換え可能な部分がなくなるまでこれを繰り返す。最左最内戦略の場合、コンパイラは、ある関数記号の書換えを行なう関数を生成する際に、あらかじめ関数の引数を評価する関数呼び出しのコードを生成し、最左最内部分から正規形を求め、書換えを繰り返す。

4 構成子マーキングによる高速化

SOFTEX/Sによる変換は、複数の書換え規則をモジュールとして順次書換え対象に適用していくことにより行なわれる。ここである変換規則のモジュールに対して、書換え対象の中で構成子項だけで構成されている項は書換え対象になることはない。そのため書換え実行前にあらかじめこの部分にマークをつけておくことにより、実行時処理系のリデックスのサーチ回数を大幅に減少することが可能になり、変換の高速化を行なうことができる。

5 評価

本システムの評価として、状態遷移図(SDL/GR)・状態遷移表(SEM)からCプログラムを合成する書換え規則のコンパイルを行ない、複数の入力仕様からCプログラムを合成した。実験結果を図4, 図5に示す[1]。書換え戦略として、どちらの例も最左最外戦略を用いている。図中、入力シンボル数は状態遷移図に記述されるシンボルの数、表のサイズは状態遷移表のサイズを示す。また括弧内の数字はリデックスのサーチ回数を示す。構成子マーキングによる最適化により、状態遷移表の例では平均2倍弱、状態遷移図における例では13倍から最大30倍程度の高速化が行なわれていることがわかる。

入力シンボル数	高速化手法無し	高速化手法有り	速度比
51	4.29(3,621,601)	0.32(122,140)	13.4
85	12.01(9,513,329)	0.64(216,323)	18.8
153	77.62(67,094,370)	2.67(847,536)	29.1

図 4: 変換速度の比較 (1) (SDL 変換規則) 単位:sec

表のサイズ	高速化手法無し	高速化手法有り	速度比
30x16	2.55(1,851,498)	1.44(612,781)	1.77
30x32	15.75(12,401,961)	8.43(3,895,584)	1.87
60x80	1047.7(721,027,503)	552.5(227,747,278)	1.90

図 5: 変換速度の比較 (2) (SEM 変換規則) 単位:sec

6 おわりに

多ソート項書換えシステムの直接実行方式のインプリメントを行なった。項書換えシステムに特化したデータ構造による実現の結果、項書換えシステムの高速な実行が可能になると共に、実行時に構成子マーキングによる書き換え候補探索の最適化をおこなうことができ、書き換え速度を大幅に向上することができた。

参考文献

[1] 佐藤, 山之内, 渡辺: ソフトウェア自動合成システム SOFTEX/S(2)-文脈自由言語と多ソート項集合の同型写像 -, 第 46 回情報処理学会全国大会,6F-2,1993.

[2] Stéphane Kaplan: A Compiler for conditional term rewriting systems, Pierre Lescanne(ed.), Proc. of RTA-87,May 1987.(LNCS 256)

[3] 澤田寿実、高橋孝一、戸村哲、二木厚吉:CafeIn の順序ソート項書換え系コンパイラ、第 9 回大会論文集、日本ソフトウェア科学会、9 1992.

[1] EWS4800/220, 実メモリ 32MB にて計測