6F−2

ソフトウェア自動合成シェルSOFTEX/S (2)
−文脈自由言語と多ソート項集合の同型写像−

佐藤 明良　山之内 徹　渡辺 正信
NEC C&Cシステム研究所

1 はじめに

従来、プログラム変換理論としてunfold/fold変換や再帰関数言語の変換など様々な提案がされているが、これら理論は主にプログラムの意味的正当性が興味の対象であり、プログラムの構文的正当性についてはほとんど調べられていない。この理由は、これら対象言語はS式などの比較的単純な構文を持ち、構文的正当性の保証は自明であることが多いからである。

しかしながら、CやC++,COBOLなどをターゲット言語とするプログラム変換理論を考えると、これら言語構文は複雑であるので、変換されたプログラムの構文的正当性(つまり、変換結果が文法に受理されること)を保証することは自明ではなくなる。これら言語構文は、通常、BNFなどを用いて文脈自由文法として記述される。従って、CやC++などを対象としたプログラム変換系を構築するためには、文脈自由言語変換系を形式化し、変換結果が構文的正当であることを保証する理論を与えることが重要となる。

本稿では、CやC++,COBOLなどをターゲット言語とするプログラム変換系を構築するための理論として、文脈自由言語から多ソート項集合への写像ξを定義し、無曖昧文法に対して、構文的正当性(項集合上ではwell-formed terms)に関して同型であることを示す。写像ξの性質として、写像された多ソート指標の多相性と文法が曖昧である場合の問題について考察する。また、文法構造と書換え規則記述能力の関係について述べる。最後に、写像ξに基づいてSOFTEX/Sパーザ部とプリンタ部[5]を作成したことを述べる。

写像ξにより、文脈自由言語変換系は多ソート項書換え系によって形式化される[2][4]。また、ξの同型性のために、互換な書換え規則によって得られる結果は常に構文的正当である(文法に受理される)ことが保証される[2][1]。

2 文脈自由言語から多ソート項集合への同型写像

定義 2.1 (文脈自由言語) 文脈自由文法Gは$< Ns, Ts, Ss, Pr >$である。ここで、Nsは非終端記号集合、Tsは終端記号集合、Ssは開始記号、$Pr : Ns \rightarrow (Ns \cup Ts)+$は生成規則集合を表す。$G$上の文脈自由言語$L(G)$は、生成規則$r \in Pr$によって開始記号$Ss$から生成される言語列の集合である。□

定義 2.2 (多ソート指標) 多ソート指標Σは、組$< S, F >$である。ここで、Sはソート記号集合、Fは関数記号集合である。それぞれの関数記号fはそれぞれの型$s_1 ... s_n \rightarrow s$をもち、$f : s_1 ... s_n \rightarrow s$と表す。ここで$s_1, ..., s_n, s \in S$であり$s_1, ..., s_n$は$f$の定義域、$s$は$f$の値域を表す。□

Software Synthesis Shell SOFTEX/S (2) −An Isomorphism between Context-free Language and Many-sorted Terms−
Akiyoshi SATO, Toru YAMANOUCHI, Masanobu WATANABE
NEC Corporation

定義 2.3 (多ソート項集合) 多ソート指標をΣ、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. $v_s \in T_s(\Sigma, V)$ *iff* $v_s \in V$; $s \in S$,
2. $f(t_1, ..., t_n) \in T_s(\Sigma, V)$ *iff* $f : s_1 ... s_n \rightarrow s$ *and* $t_i \in T_{s_i}(\Sigma, V)$ *for* $1 \leq i \leq n$.

また、$T(\Sigma, V) \stackrel{\text{def}}{=} \bigcup_{s \in S} T_s(\Sigma, V)$は$\Sigma$と$V$上の多ソート項集合を表す。□

定義 2.4 (写像ξ) 文脈自由言語$L(G)$から多ソート項集合$T(\Sigma_G, \phi)$への写像ξを以下のように定義する:

1. 指標$\Sigma_G =< S, F >$は文脈自由文法$< Ns, Ts, Ss, Pr >$から以下のように定義される:
   - $s \in S$ *iff* $s \in (Ns \cup Ts)$,
   - $s : s_1 ... s_n \rightarrow s \in F$ *iff* $s \in (Ns \cup Ts)$ *and* $(s \rightarrow s_1 ... s_n) \in Pr$
2. 写像$\xi : L(G) \rightarrow T(\Sigma_G, \phi)$は以下のように定義される:
   - $\xi(l) \stackrel{\text{def}}{=} t$ *iff* t は言語列 $l \in L(G)$ の構文解析木

□

例 2.5 (写像ξ)

以下に示す文脈自由文法をG_1とする。(G_1はANSI C言語文法の一部を表しており、BNFで記述されている)

```
Ss :          declaration | statement                    ;
statement :   ID "(" arguments ")" ";"
            | ID "(" ")" ";"                             ;
arguments :   ID
            | ID "," arguments                           ;
declaration : type_name ID "(" arg_decls ")" ";"
            | type_name ID "(" ")" ";"                   ;
arg_decls :   type_name ID
            | type_name ID "," arg_decls                 ;
type_name :   /* nothing */
            | "int" | "char" | "float"                   ;
```

この例において、言語列l_1 = `int func(char c, float f);`は写像ξによって以下の項t_1に写像される。

```
t1 = Ss( declaration( type_name( int ), ID( func ), "(",
         arg_decls( type_name( char ), ID( c ), ","
         arg_decls( type_name( float ), ID( f ))), ")", ";" ))
```

曖昧な文法の下では、ひとつの言語列から複数の解析木に写像される。上述の文法G_1は曖昧であるので、言語列l_2 = `func();`は以下のような複数の項t_{2a}, t_{2b}に写像される。

```
t2a = Ss( statement( ID( func ), "(", ")", ";" ))
t2b = Ss( declaration( type_name(), ID( func ), "(", ")", ";" ))
```

□

命題 2.6 (写像 ξ は 1 対 1 の対応) G を無曖昧な文脈自由文法とするとき、写像 $\xi : L(G) \rightarrow T(\Sigma_G, \phi)$ は 1 対 1 の対応である。□

定理 2.7 (写像 ξ の構文的正当性に関する同型性) G を無曖昧な文脈自由文法とするとき、写像 $\xi : L(G) \rightarrow T(\Sigma_G, \phi)$ は、構文的正当性および整合的項 (*well-formed terms*) に関して同型である。つまり、

$$\xi(l_1 \in L(G)) = t_1 \implies t_1 \in T(\Sigma_G, \phi), \text{ and}$$
$$\xi^{-1}(t_2 \in T(\Sigma_G, \phi)) = l_2 \implies l_2 \in L(G).$$

証明. 命題 2.6 より、無曖昧文法の基で、写像 ξ は 1 対 1 の対応である。言語列 l は、$l \in L(G)$ の時、その時に限り構文的正当であり、(変数を含まない) 項 t は、$t \in T(\Sigma_G, \phi)$ の時、その時に限り整合的である。□

3 多ソート指標の多相性

命題 3.1 (多ソート指標の多相性) 文脈自由言語から写像 ξ によって写像された多ソート項集合の指標において、各関数記号の定義域は *overloading* 多相であり、値域は単相である。□

各関数記号の定義域が overloading 多相 [6] であるとは、同じ関数記号に対して複数の定義域ソート列を持つこと、つまり、$\exists f : w_1 \rightarrow s_1,\ f : w_2 \rightarrow s_2 \in F;\ \ w_1, w_2 \in S+;\ \ s_1, s_2 \in S;\ \ w_1 \neq w_2$ であることである。また、値域が単相であるとは、同じ関数記号ならば値域ソートは唯一であること、つまり、$\forall f : w_1 \rightarrow s_1,\ f : w_2 \rightarrow s_2 \in F;\ w_1, w_2 \in S+;\ s_1, s_2 \in S;\ s_1 = s_2$ であることである。

写像 ξ によって得られる指標は、関数記号値域が単相であるため、書換え規則の互換性検証 [3] が、一般的な多ソート書換え系や順序ソート書換え系の場合 [8] [9] に比較して、単純化されている [1]。

4 文法の曖昧性

命題 4.1 (文法の曖昧性) 文脈自由文法が曖昧であるとき、その時に限り、写像 ξ は 1 対多の対応となる。□

文法の曖昧性は、構文的正当性の観点からは問題にはならない。なぜならば、ある言語列 l が写像 ξ によって複数の項 $t_1, ..., t_n$ に写像されるとしても、どの項 $t_1, ..., t_n$ も well-formed であり、かつ、それら項は写像 ξ^{-1} によってもとの言語列 l に写像されるからである (図 1(a))。

しかしながら、文法の曖昧性は変換の意味にとっては問題となりうる。つまり、複数の項 $t_1, ..., t_n$ のどの項を選択するかによって、書換え結果は異なることがある (図 1(b))。

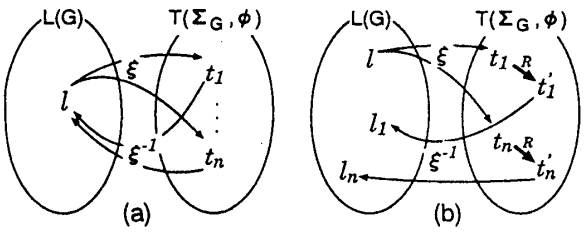

図 1: 写像 ξ と文法の曖昧性

5 文法構造の違いによる書換え規則記述能力

任意の文脈自由文法の言語に対して、写像 ξ は構文的正当性を保証する。しかしながら、同じ言語であっても文法構造が異なることは、どのような変換規則が記述可能であるかに影響する [2]。つまり、文法の非終端記号集合をどのようにとるかによって、書換え規則の記述能力が決定されるのである。例えば、例 2.5の文法 G_1 において、ソート *type_name*, ID, *arg_decls* の変数をそれぞれ v_{tn}, v_{id}, v_{ad} とすると、ANSI C-like なプロトタイプ宣言を K&R C-like なものに変換する規則は以下のように記述できる:

$$declaration(v_{tn},\ v_{id},\ "(",\ v_{ad},\ ")",\ ";")$$
$$\rightarrow declaration(v_{tn},\ v_{id},\ "(",\ ")",\ ";")$$

一方、文法中に非終端記号 *arg_decls* が存在しない文法 G_2 の場合 (例えば、*arg_decls* の生成規則を全て fold してしまった場合など)、たとえ $L(G_1) = L(G_2)$ であっても、有限の書換え規則では同様の規則を記述することはできない (その fold されたソートの台が無限である場合、有限の書換え規則では不可能になる)。このように、文脈自由言語変換系の記述能力は (たとえその言語が同じであっても) 文法の構造に依存するのである。

6 DSL/C++ パーザとプリンタの実現

本稿で述べた写像 ξ に基づいて、ソフトウェア自動合成シェル SOFTEX/S [5] の DSL/C++ 言語のパーザ部とプリンタ部を作成した。パーザ部は写像 $\xi : L(G_{C++}) \rightarrow T(\Sigma_{G_{C++}}, \phi)$ に相当し、プリンタ部は写像 $\xi^{-1} : T(\Sigma_{G_{C++}}, \phi) \rightarrow L(G_{C++})$ に相当する。DSL/C++ 言語文法は、ARM[7] の C++ 文法サマリを基に、ソート記号・関数記号・項書換え規則を追加記述可能な機能を備えている [5]。ARM の C++ 文法サマリは文脈自由文法で記述されているが、曖昧な文法である。従って、写像 ξ は 1 対多の対応となるが、DSL/C++ パーザは LR 構文解析における shift 優先により曖昧性解消処理を行なうことによって構文解析を行ない、文法曖昧性問題を解決している。

参考文献

[1] 三木, 佐藤, 山之内, 渡辺: ソフトウェア自動合成シェル SOFTEX/S (4) – 項書換え規則の互換性検証システム –, 第 46 回情報処理学会全国大会, 6F-4, 1993.

[2] 佐藤, 山之内, 渡辺: ソフトウェア自動合成システム SOFTEX における書換え規則の正しさについて, 通信サービス品質とネットワーク品質ワークショップ予稿集, 信学会, pp.57-67, 1992.

[3] 佐藤, 山之内, 渡辺: 多ソート書換え規則の互換性: 日本ソフトウェア科学会第 9 回大会, C4-4, pp.309-312, 1992.

[4] 友部, 山之内, 渡辺: ソフトウェア自動合成シェル SOFTEX/S (3) – 項書換えシステムの直接実行方式とその高速化 –, 第 46 回情報処理学会全国大会, 6F-3, 1993.

[5] 山之内, 佐藤, 山田, 渡辺, 岡: ソフトウェア自動合成シェル SOFTEX/S (1) – 設計思想とシステム構成 –, 第 46 回情報処理学会全国大会, 6F-1, 1993.

[6] Cardelli, L. and Wegner, P.: On Understanding Types,Data Abstraction,and Polymorphism, *ACM Computing Surveys*, Vol.17, No.4, pp.471-522, 1985.

[7] Ellis, M. and Stroustrup, B.: *The Annotated C++ Reference Manual*, Addison-Wesley, 1990.

[8] Smolka, G., Nutt, W., Goguen, J.A. and Meseguer, J.: Order-Sorted Equational Computation, in *Resolution of Equations in Algebraic Structures Volume2 Rewriting Techniques*, Academic Press, 1989.

[9] Waldmann, U.: Compatibility of Order-Sorted Rewrite Rules, *Lecture Notes in Computer Science 516*, pp.407-416, 1991.