1L-05

プログラム逆化のための文脈自由表現の等価関係に基づく項書換え系決定化手法の実装

熊谷 妃美香 †　西田 直樹 ‡
名古屋大学 †　名古屋大学 ‡

1 はじめに

関数 f を定義するプログラムから f の逆関数 f^{-1} を定義するプログラムを生成することを（プログラム）逆化という．一般的に，逆化システムは n 引数関数 f の定義を入力とし，次の性質を満たす関数 f^{-1} の定義を出力する：f に入力可能な任意の値 $v_1,\ldots,v_n$ と任意の出力可能な値 v に対して，$f(v_1,\ldots,v_n)=v$ のとき，かつそのときに限り，$f^{-1}(v)=(v_1,\ldots,v_n)$ となる．逆化システムは暗号化／復号化プログラムなどの実装において一方のプログラムからもう一方を自動生成することに利用できる．

上述の性質を満たす f^{-1} を $f^{-1}(v)$ と $(v_1,\ldots,v_n)$ の関係として定義することは容易である [5]．一方で，f が単射関数であっても，そのように生成された f^{-1} の定義は決定的でない場合，すなわち関数定義になっていない場合がある．生成した定義を決定的な定義，すなわち関数定義とみなせる形式に変換することを決定化という．これまでに提案された決定化手法では決定化に成功しないプログラムが存在し，逆化において決定化は最も重要な課題である．

先行研究の 1 つに，決定性条件付き項書換え系（DCTRS）を対象とし，文法プログラムと文脈自由表現（CFE）を用いた決定化手法 [3] がある（図 1）．決定化に CFE の等価性を利用しており既存のベンチマークすべての決定化に成功する一方で，変換は手作業であり，自動変換の指針は示されていない．

本研究では，図 1 の枠組みを自動化することをめざす．そのために，CFE の等価性を利用した決定化を自動化する指針を示し，文法プログラムと CFE の相互変換，CFE の等価性を利用した決定化（図 1 の太字の箇所）を実装する．そして，ベンチマークに対する実験から自動化の指針を評価する．

Implementing Determinization of Term Rewrite Systems via Equivalence of Context-Free Expressions Toward Program Inversion
† Himika Kumagai, Nagoya University
‡ Naoki Nishida, Nagoya University

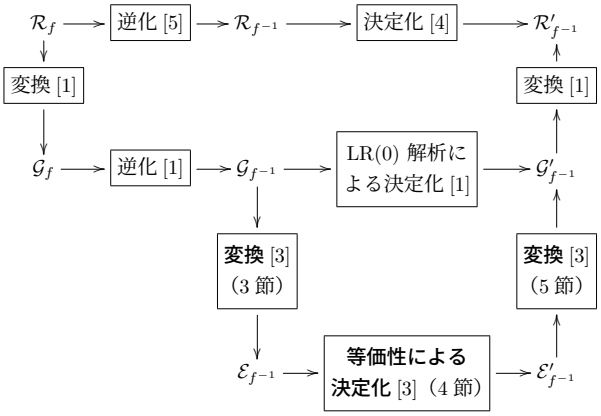


図 1　先行研究 [5, 1, 4, 3] の枠組みと本研究の位置付け

2 入出力ファイルの形式

本章では本ツールで扱う入出力ファイルについて説明する．以降，リストの末尾に要素を追加する関数 snoc を例として使用する．

$$\mathcal{R}_{\mathsf{snoc}}=\left\{\begin{array}{r}\mathsf{snoc}(\mathsf{n},y)\to\mathsf{c}(y,\mathsf{n})\\ \mathsf{snoc}(\mathsf{c}(x,xs),y)\to\mathsf{c}(x,ys)\\ \Leftarrow \mathsf{snoc}(xs,y)\twoheadrightarrow ys\end{array}\right\}$$

本ツールは逆化した文法プログラム $\mathcal{G}_{\mathsf{snoc}^{-1}}$ [1] を入力として受け取る．

$$\mathcal{G}_{\mathsf{snoc}^{-1}}=\left\{\begin{array}{l}\mathsf{snoc}^{-1}\to \mathsf{c?}\ (2\ 1)\ \mathsf{n?}\ \mathsf{n!}\\ \mathsf{snoc}^{-1}\to \mathsf{c?}\ (2\ 1)\ \mathsf{snoc}^{-1}\ (3\ 1\ 2)\ \mathsf{c!}\end{array}\right\}$$

文法プログラムは左辺が非終端記号，右辺がオペレータと非終端記号（関数呼出しに相当）からなるリストで構成される生成規則の集合である．各オペレータはスタックへの操作を表現する．例えば，n! はデータ n をスタックの先頭に追加し，n? はスタックの先頭と n を照合し，(2 1) はスタックの先頭二つのデータを入れ替える．

$\mathcal{G}_{\mathsf{snoc}^{-1}}$ を表す入力ファイルは以下の形式となる．

```
(SIGNATURE snocInv/1/2 n/0/1 c/2/1)
(RULES
  snocInv -> c? (2 1) n? n!;
  snocInv -> c? (2 1) snocInv (3 1 2) c!
)
```

1 行目は使用する関数記号の名前と引数個数，出力個数を宣言している．本ツールは文法プログラムを

Copyright ©2022 Information Processing Society of Japan.
All Rights Reserved.

$$
\begin{aligned}
&\textbf{E1} && E\cdot E_1\cdot E' + \cdots + E\cdot E_n\cdot E' \approx E\cdot(E_1+\cdots+E_n)\cdot E'\\
&\textbf{E2} && (E\cdot E_1\cdot\Box\cdot E_2)^*.(E\cdot E_3) \approx E\cdot((E_1\cdot E\cdot\Box\cdot E_2)^*.E_3)\\
&\textbf{E3} && ((\Box\cdot E_1)^*.E_2)\cdot E_3 \approx E_2\cdot((E_1\cdot\Box)^*.E_3)\\
& && \vdots\\
&\textbf{E8} && E + E_1\cdot((E_1\cdot\Box\cdot E_2)^*.E)\cdot E_2 \approx (E_1\cdot\Box\cdot E_2)^*.E
\end{aligned}
$$

図 2　決定化に利用する（CFE の）等価関係 [3]

同様の形式で出力する．

3　文法プログラムから CFE への変換

CFE は文脈自由言語で正規表現を利用できるように拡張した表現である [2]．文法プログラムは等価な CFE に変換可能であり，生成規則中の左辺に現れる非終端記号が □ に置き換えられる [3]．$\mathcal{G}_{\mathsf{snoc}^{-1}}$ は以下の CFE に変換される．

$$\mathcal{E}_{\mathsf{snoc}^{-1}} = (\mathsf{c?}\ (2\ 1)\ \Box\ (3\ 1\ 2)\ \mathsf{c!})^*.(\mathsf{c?}\ (2\ 1)\ \mathsf{n?}\ \mathsf{n!})$$

以降，E を任意の CFE を表す記号として用いる．

4　CFE の等価性による決定化手続き

先行研究 [3] では図 2 に示す CFE の等価関係を用いて CFE を決定化する枠組みが提案された．しかし，どのように等価関係を用いるかの指針が示されていない．そこで，本研究では等価関係に基づいた決定化として，以下の手続きを提案し，実装した．

(1) **E1** を左から右への変換として最左最外戦略の下でできる限り適用する．
(2) **E2** を用いて (1) と同様に CFE を変換する．
(3) **E3** を用いて (1) と同様に CFE を変換する．
(4) **E4** を用いて (1) と同様に CFE を変換する．
(5) **E5** を用いて (1) と同様に CFE を変換する．
(6) **E6** を用いて (1) と同様に CFE を変換する．
(7) **E7** を用いて (1) と同様に CFE を変換する．
(8) (1)–(7) をもう一度実行し，CFE を変換する．
(9) 部分 CFE $(E_1)^*.(E_2)$ の E_1 中に最初に出現する $c?$ よりも前に現れる $(E_3)^*.(E_4)$ を **E8** を右から左に用いて変換する．これを最左最外戦略の下でできる限り適用し，CFE を変換する．E_2 に対しても E_1 と同様の変換を行う．
(10) 部分 CFE $E_1\cdot E_2$ に現れる部分 CFE $E_3 + E_4$ を **E1** を右から左に用いて変換する．これを最左最外戦略の下でできる限り適用し，CFE を変換する．

上記の (1) と (4) においてはオペレータの列は構文的だけでなく，意味的に等価であれば等しいとみなして等式の左辺に照合する．ここで，意味的に等価であるとは，スタックに対する操作が等しいことを意味する．例えば，c? n! (2 3 1) と n! (2 1) c? はどちらも $\mathsf{c}(a,b):ts$ の形式のスタックを $a:b:\mathsf{n!}:ts$ に遷移させるため，意味的に等価である．

上述の手続きにより，$\mathcal{E}_{\mathsf{snoc}^{-1}}$ は以下の決定的な CFE $\mathcal{E}'_{\mathsf{snoc}^{-1}}$ に変換される．

$$\mathcal{E}'_{\mathsf{snoc}^{-1}} = \mathsf{c?}\ (2\ 1)\ ((\mathsf{c?}\ (2\ 1)\ \Box\ (3\ 1\ 2)\ \mathsf{c!})^*.(\mathsf{n?}\ \mathsf{n!}))$$

5　CFE から文法プログラムへの変換

本ツールは先行研究に基づき，CFE を文法プログラムに変換し，結果を出力して終了する．$\mathcal{E}'_{\mathsf{snoc}^{-1}}$ は $\mathcal{G}'_{\mathsf{snoc}^{-1}}$ に変換され，以下のように出力される．

```
(SIGNATURE snocInv/1/2 n/0/1 c/2/1)
(RULES
  snocInv -> c? (2 1) F1;
       F1 -> c? (2 1) F1 (3 1 2) c!;
       F1 -> n? n!
)
```

F1 は新しく生成された補助的な非終端記号である．

6　まとめ

本研究では先行研究の決定化の自動化の指針を示し，それに基づき自動的な決定化を実装した．本ツールは先行研究 [3] で決定化の成功例として使用されている 8 個のベンチマーク（snoc，reverse，snocrev，inc，unbin，treelist，unbin2，reverse3）すべてを決定化することに成功した．今後の課題は，等価関係に基づいた変換の最適な適用順序や適用回数についての検証である．

参考文献

[1] R. Glück and M. Kawabe. A method for automatic program inversion based on LR(0) parsing. *Fundam. Informaticae*, 66(4):367–395, 2005.
[2] H. Leiß. Towards Kleene algebra with recursion. In *Proc. CSL 1991*, Vol. 626 of *LNCS*, pp. 242–256. Springer, 1992.
[3] N. Nishida and M. Niwa. Determinization of inverted grammar programs via context-free expressions. *J. Log. Algebraic Methods Program.*, 122:1–26, 2021.
[4] N. Nishida and G. Vidal. Computing more specific versions of conditional rewriting systems. In *Revised Selected Papers of LOPSTR 2021*, Vol. 7844 of *LNCS*, pp. 137–154, 2013.
[5] 西田, 酒井, 坂部. 構成子項書換え系の逆計算プログラムの生成. 電子情報通信学会論文誌 D-I, J88-D-I(8):1171–1183, 2005.

Copyright ©2022 Information Processing Society of Japan.
All Rights Reserved.