プログラミング 10－9
（1996．11．1）

関数論理型言語の停止性検証

田中あずさ　中西正和
慶應義塾大学 理工学研究科 計算機科学専攻

近年，関数論理型言語の研究が盛んになってきているが，関数論理型言語の停止性検証の研究はまだあまりなされていない．そこで，本研究ではナローイングを操作的意味とする関数論理型言語 ALF を題材に，停止性の検証を試みた．

K. Verschaetse らは述語のインスタンス間の関係を用い，プログラムについて連立不等式を立て，それが解けないことを示すことで論理型言語の停止性検証を行なった．本研究では項書換え系の停止性検証に用いられる辞書式順序を利用することで，K. Verschaetse らの方法を関数論理型言語に適用し，停止性検証を行なった．

Automatic Termination Analysis of Functional Logic Programs

Azusa TANAKA　Masakazu NAKANISHI
Department of Computer Science, Keio University

We present an implementation method for the automatic termination analysis of ALF. ALF is a functional logic programming language. The operational semantics of the language consists of resolution to solve predicates, and narrowing and rewriting to evaluate functional expressions.

We introduced K. Verschaetse's method of termination analysis on logic programs as the basic idea. His framework based on the notion of a level mapping. They formulated inequalities for the program, and shown inequalities were not solvable. We use the lexicographic path order to extend K. Verschaetse's method for functional logic programs.

1 はじめに

計算の停止性は重要な性質である．項書換え系や論理プログラムの停止性検証に関しての研究が数多くなされている [1, 7].

近年，関数論理型言語についての研究が盛んに行なわれている [2, 3] が関数論理型言語における停止性検証の研究はまだあまりなされていない．そこで，本研究では関数論理型言語 ALF における停止性の検証を目的とし，その具体的方法を提案し，実装，考察を行なう．

停止性とは計算の過程が無限に続かないことをいう．計算の停止性を示すことができると計算の正当性を検証することが容易になる．計算の正当性とは「計算が停止し，かつ答が正しい」ということである．プログラムの停止性の検証はプログラムを実行することなしに停止性を示し，プログラムの正当性の検証を容易にする．

一般には停止性検証は決定不能な問題であるが，項書換え系などで，実用上重要な多くのプログラムの停止性の機械的検証が可能であることが示されている [5].

2 関数論理型言語 ALF

関数論理型言語は関数型言語と論理型言語の両方のパラダイムを持ちあわせた言語であり，関数表現をゴールリテラルに用いたり，等式の条件に任意の論理式を用いたりすることができる．その停止性検証には関数論理型言語独自の特徴に起因する問題があると考えることができる．

関数論理型言語の停止性検証をする上で重要なのはその操作的意味である．

関数論理型言語の一つに ALF [3] というプログラミング言語がある．本研究では，ALF のサブセットについて，その停止性の検証を目的とする．ALF ではプログラミングの簡便さのためにモジュールや型宣言などが用意されているが，本研究ではより単純な形にしたものについて考える．

ALF のプログラムは次の 3 種類の節からなる．

1. = を用いないで定義される関係節
2. ナローイングに使われる条件付等式
3. 書き換えに使われる条件付等式

特に条件部のないものを公理という．

2.1 操作的意味

ALF の操作的意味は述語の融合 (resolution) と関数表現の最左最内基底ナローイング (leftmost-innermost basic narrowing) と書換え (rewriting) に基づく．

以下に ALF の操作的意味を表す．

ゴールは骨格とその環境で書くことができる [4]. 骨格は元のプログラムの項とリテラルの集合で，環境は実際のゴールを得るための骨格に適用する代入である．

初期ゴールは $< G; id >$ で表される．項 t の位置 π に対して t/π は t の位置 π にある部分項であることを示し，$t[\pi \leftarrow s]$ は t の部分項 t/π を s で置き換えることによって得られる項であるとし，次の導出規則を定義する．

$< L_1, \cdots, L_n; \sigma >$ を与えられたゴールとする．$L_1, \cdots, L_n$ を骨格リテラルという．σ は環境であり，ある代入 σ' と σ に対して，$\sigma' \circ \sigma$ はそれらの合成である．

1. L_1 が $s = t$ の等式で，$\sigma(s)$ と $\sigma(t)$ の最汎単一化代入 σ' があるとき，ゴール

   $$< L_2, \cdots L_n; \sigma' \circ \sigma >$$

   が反射 (reflection) によって導出される．
2. L_1 が等式でなく，プログラム節 $L \leftarrow C$ があり，σ' が $\sigma(L_1)$ と L の最汎単一化代入であるとき，ゴール

   $$< C, L_2, \cdots, L_n; \sigma' \circ \sigma >$$

   が融合 (resolution) によって導出される．

3. π が最初の骨格リテラル L_1 の最左最内 (leftmost-innermost) の位置とする．すなわち，部分項 L_1/π はその引数の全てが変数と構成子からなる項である定義された関数記号である．

   (a) プログラム節 $l = r \leftarrow C$ があり，$\sigma(L_1/\pi)$ と l が最汎単一化代入 σ'によって単一化可能である時，ゴール

   $$< C, L_1[\pi \leftarrow r], L_2, \cdots, L_n; \sigma' \circ \sigma >$$

   が最内基底ナローイング (innermost basic narrowing) によって導出される．

   (b) x が新しい変数で，σ' が代入 $x \leftarrow \sigma(L_1/\pi)$ である時，ゴール

   $$< L_1[\pi \leftarrow x], L_2, \cdots L_n; \sigma' \circ \sigma >$$

   が，最内反射 (innermost reflection) によって導出される．

4. π が L_1 の非変数 (non-variable) な位置にあり，$l = r \leftarrow C$ がプログラム節であり，σ' が$\sigma(L_1/\pi) = \sigma'(l)$ なる代入であり，ゴール $< C; \sigma' >$ が $\sigma(L_1)$ のどんな変数も具体化できないような空のゴールを導出することができる時，ゴール

   $$< L_1[\pi \leftarrow \sigma'(r)], L_2, \cdots, L_n; \sigma >$$

   が書き換え (rewriting) によって導出できる．
5. L_1 が等式であり，両辺の同じ最外の位置の構成子が違うものである時，全てのゴールが拒否される (rejected)．

2.2 停止性

本研究では，ゴールに対して 2.1 節で定義したどの導出規則も適用できなくなったときに，プログラムは停止するとする．

3 停止性検証

ALF のプログラムは関係節と条件付等式からなる．項書換え系の停止性検証は書き換えられる項の間の関係が用いられる．論理プログラムの停止性検証は節の頭部と体部に現れるリテラルの間の関係が用いられる．条件付等式の場合，等式と条件の間の関係を調べなければならない．

停止性の機械的検証の方針としては，Verschaetse [6] らの論理型言語の自動停止性検証の方法を関数論理型言語に用いることができるように拡張する．その際に辞書式順序 (lexicographic path order) [5] を用いた項書換え系の停止性検証の方法を利用する．

3.1 論理型言語の停止性の定義

論理型言語では，質問の集合 S に対して，S のどの要素に対してもその線形導出木が有限であるとき，プログラムが S に対して停止するという．

定義 3.1 (level mapping)
質問の集合 S に関する level mapping とは，述語のインスタンスから自然数への写像である．

定義 3.2 (recurrency)
プログラム P が質問の集合 S に関して，再帰的である (P is recurrent with respect to S) とは，以下のような level mapping $|.|$ が存在することをいう．

$$|A'| > |B_i|$$

ここで，A' の導出節 $A'\theta \leftarrow B_1, \ldots, B_m$ について，B_i は A' と同じ述語記号を持つ．

再帰的であることと停止性があることは同値である．

3.2 論理型言語の停止の条件

定義 3.3 最小循環集合 (minimal cyclic collection)
P の最小循環集合とは，以下の条件を満たす P 中の節の有限集合 $\{cl_1, \ldots, cl_m\}$ である．

- どの対 $\{cl_i, cl_j\}$ についても，節の頭部は異なる述語記号を持ち，
- その集合の要素の順序 $cl_{i(1)}, \ldots, cl_{i(m)}$ が存在し，アトムの m 個組 $A'_1, A'_2, \ldots, A'_m$ が存在して，各 $k(1 \leq k \leq m)$ について，
  - アトム A'_k は節 $cl_{i(k)}$ の体部にある
  - $cl_{i(k)}$ の頭部を A_{k-1} と表すと，A'_k が $A_{k \bmod m}$ と同じ述語記号を持つ．

定義 3.4 semi-linear-norm

$$
\begin{aligned}
&||V|| = 0 \\
&\qquad V \text{ が変数のとき} \\
&||f(t_1, \ldots, t_n)|| = c + ||t_{i_1}|| + \cdots + ||t_{i_m}|| \\
&\qquad c \in N, \{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\} \text{ で} \\
&\qquad c, i_1, \ldots, i_m \text{ は } f/n \text{ に依存する}
\end{aligned}
$$

停止性の証明にはよくリストの長さ (list-length) や項の大きさ (term-size) が用いられる．

定義 3.5 rigid-norm

項 t が $||.||$ に関して rigid であるとは，任意の代入 σ に対して，

$$||t\sigma|| = ||t||$$

となるときをいう．

定義 3.6 natural level mapping

任意の $p(t_1, t_2, \ldots, t_n)$ に対し，

$$|p(t_1, t_2, \ldots, t_n)|_{nat} = \begin{cases} \Sigma_{i \in I^p_n} ||t_i|| & if\ I^p_n \neq \{\} \\ 0 & \text{それ以外} \end{cases}$$

ここで，$||.||$ は semi-linear norm であり，I^p_n は p/n にのみ依存する引数位置の集合であり，任意の $p(u_1, \ldots, u_n)$ と任意の $i \in I^p_n$ に対し，u_i は $||.||$ に関して rigid である．

定義 3.7 size expression

$||.||$ を semi-linear norm とする．項 t を size expression に写す写像 $abs_{||.||}(t)$ を以下のように定義する．

$$
\begin{aligned}
&abs_{||.||}(V) = V \quad V \text{ が変数のとき} \\
&abs_{||.||}(f(t_1, \ldots, t_n)) = \\
&\qquad c + abs(t_{i_1}) + \cdots + abs(t_{i_m}) \\
&\qquad \text{それ以外}
\end{aligned}
$$

ここで，c は自然数であり，$\{i_1, \ldots, i_m\} \in \{1, \ldots, n\}$ であり，c と $i_1, \ldots, i_m$ は f/n に依存する．

定義 3.8 $|.|$ を質問の集合 S から得られる natural level mapping とする．このとき，

$$
\begin{aligned}
&abs_{|.|_{nat}}(p(t_1, \ldots, t_n)) = \\
&\qquad abs_{||.||}(t_{i_1}) + \cdots + abs_{||.||}(t_{i_k})
\end{aligned}
$$

ここで，$\{i_1, \ldots, i_k\} = I^p_n$ は引数位置の集合で，$|.|_{nat}$ によって p/n と関連づけられていて，$||.||$ は $|.|_{nat}$ の semi-linear norm である．

命題 3.1 $|.|_{nat}$ を質問の集合 S から作られる natural level mapping であるとする．ここで P の任意の最小循環集合の代表元を一つとり，以下のように書く．

$$
\begin{aligned}
A_0 &\leftarrow B^1_1, \ldots, A'_1, \ldots, B^1_{n_1} \\
&\vdots \\
A_{m-1} &\leftarrow B^m_1, \ldots, A'_m, \ldots, B^m_{n_m}
\end{aligned}
$$

ここで，各節は標準化され，わかれている．以下の系

$$
\begin{cases}
abs_{|.|_{nat}}(A'_1) \geq abs_{|.|_{nat}}(A_1) \\
\qquad\qquad \vdots \\
abs_{|.|_{nat}}(A'_{m-1}) \geq abs_{|.|_{nat}}(A_{m-1}) \\
abs_{|.|_{nat}}(A'_m) \geq abs_{|.|_{nat}}(A_0)
\end{cases}
$$

が自然数上で解を持たないとき，P は S と $|.|_{nat}$ に関して再帰的である．

3.3 辞書式順序

この順序を用いると，与えられた書換え規則だけに着目して，項書換え系の停止性を証明できる．

定義 3.9 項の集合を $\mathcal{T}(\mathcal{F},\mathcal{V})$ とする．また関数記号の集合 $\mathcal{F}$ 上の厳格半順序を $\succ$ とし，これを優先順位と呼ぶ．この優先順位をもとに，辞書式順序 $\succ_{lpo}$ を次のように帰納的に定義する．項 $s=f(s_1,\cdots,s_n)$ と t が条件 *1*〜*3* のいずれかを満たすとき，$s\succ_{lpo}t$ とする．

1. $t=f(t_1,\cdots,t_n)$ かつ 条件 *(a)*〜*(c)* を満たす $i\in\{1,\cdots,n\}$ が存在する．
(a) 任意の $j\in\{1,\cdots,(i-1)\}$ について，$s_j=t_j$，
(b) $s_i\succ_{lpo}t_i$，
(c) 任意の $j\in\{(i+1),\cdots,n\}$ について，$s\succ_{lpo}t_j$．
2. $t=g(t_1,\cdots,t_m), f\succ g$，
かつ $\forall i\in\{1,\cdots,m\}, s\succ_{lpo}t_i$．
3. $s_i\succeq_{lpo}t$ を満たす $i\in\{1,\cdots,n\}$ が存在する．
ただし，$\succeq_{lpo}$ は $\succ_{lpo}$ の反射閉包を表す．

4 ALF の停止性検証

操作的意味からわかるように，ALF は基本的には論理型言語と同様に実行される．したがってその停止性検証も同様に行なうことができる．違うのは等式で結ばれている項の書き換えの部分である．これがあることで，節の導出の際に，リテラルの引数項が書き換えられていることがある．natural level mapping は引数項の興味ある引数位置から定めるので，等式の左辺，すなわち '=' という述語の第 1 引数項を興味ある引数項として natural level mapping を定めるとすると，等式の左辺の項が右辺の項へと書き換えられたときに，その norm が大きくならなければ，問題はない．そこで，等式の左辺と右辺の順序関係を辞書式順序を用いて調べる．辞書式順序によって，等式の左辺の項が右辺の項より大きいという順序が得られなかった場合には，natural level mapping に右辺の項を興味ある項として加える．

以下の手順で，停止性検証を行なった．

1. 操作的意味のナローイング・書換えが適用される全ての等式について，辞書式順序により順序関係を調べる．
2. 最小循環集合を作る．
3. 1．2．の結果を用いて，述語の natural level mapping を定める．
4. 各最小循環集合について，不等式を立てる．
5. 不等式が解けないことを示す．

5 アルゴリズム

停止性検証アルゴリズムについて述べる．

5.1 書き換え規則の順序

書き換え規則の集合 R が与えられたとき，R 中の全ての規則について左辺の項が右辺の項より大きいという順序を辞書式順序によってつけることができるかどうかを調べる．R 中のある書き換え規則 $R1$ についてその左辺の項 s と右辺の項 t が，$s\succ_{lpo}t$ となるかを調べ，R 中の全ての書き換え規則について再帰的に調べる．

ある項 s,t が与えられ $s\succ_{lpo}t$ となるかどうかを調べる．演算子の集合上の厳格半順序を $\succ$ とする．

- s が変数の場合は失敗を告げる．
- s が変数でなく，t が変数の場合，s 中に t が生起していれば $s\succ_{lpo}t$ である．

s も t も変数でない場合，$s=f(s_1,\cdots,s_m)$，$t=g(t_1,\cdots,t_n)$ とする．

1. $f=g$ の場合，
(a) s,t について，先頭からある位置までは $s_j=t_j$ となり，はじめて，$s_j\neq t_j$ となった項について，$s_j\succ_{lpo}t_j$ である．
(b) (a) の残りの項について，$s\succ_{lpo}t_j$ である．
これらの条件を満たすとき，$s\succ_{lpo}t$ である．
2. $f\neq g$ の場合，$f\succ g$ かつ t 中の全ての項について $s\succ_{lpo}t_j$ のとき，$s\succ_{lpo}t$ である．

5.2 最小循環集合

節の集合 C のある節 $C1$ を含む最小循環集合を全て求め，同様に残りの C 中の節についてもそれを含む最小循環集合を全て求める．

ある節 $C1$ と節の集合 $C-C1$ を与えられ，最小循環集合を全て求める．

1. 節 $C1$ の頭部の述語と同じ述語を体部に含む節 $C2$ を最小循環集合の候補とする．
2. 1. 3. で作った候補が最小循環集合の場合は，節 $C1$ とそれを最小循環集合の一つとして返す．
3. そうでない場合は，$C2$ の頭部の述語と同じ述語を体部に含むような節と $C1$ と $C2$ を候補の集合として，2. へ戻る．
4. 候補を作れなくなったら終了する．

5.3 不等式の解法

定理 5.1

$$\begin{cases} T_1 \geq T_2 \\ T_3 \geq T_4 \end{cases}$$

であることは、$T_1+T_3 \geq T_2+T_4$ であることと同値。

この定理を用い，連立不等式の全ての不等式の左辺の和と右辺の和を比較し，左辺の和に現れる変数が全て右辺の和に現れ，右辺に自然数がある時に，その連立不等式は自然数の範囲で解を持たないことを示した．

6 結果

挿入ソートのプログラムに対する停止性検証を例とする．

M が最小循環集合，I が連立不等式，A は停止するとき yes，しないとき no になる．

```
| ?- main('ex/isort',M,I,A).

A = yes,
I = [[[_A + _B,+(1,_A,1,_B)]],[[_C,+(1,1,_C)]],
[[1+ _E,+(1,1,_D,1,_E)]]],
M = [[[:-,s(_A) < s(_B),_A < _B]],
[[:-,even(s(s(_C))),even(_C)]],
[[:-,member(_F,[_D | _E]),member(_F,_E)]]] ?
```

図 1: 結果

7 結論

ALF のプログラムに対して，この方法で停止性検証をすることができることが分かった．今後，どの程度この方法が有効か知るために，どのようなプログラムに対しては停止性検証ができないかを調べる必要がある．

参考文献

[1] Danny de Schreye and Stefaan Decorte. Termination of logic programs :the never-ending story. *THE JOURNAL OF LOGIC PROGRAMMING*, pp. 199–260, 1994.

[2] 浜名誠, 西岡知之, 中原鉱一, アート・ミデルドープ, 井田哲雄. 作用型項書換え系に基づく関数論理型言語の設計と実装. 情報処理学会論文誌, pp. 1897–1905, 1995.

[3] M. Hanus. Efficient implementation of narrowing and rewriting. *Workshop on Processing Declarative Knowledge*, Vol. 567, pp. 344–365, 1991.

[4] S. Hölldobler. From paramoduration to narrowing. In *In Proc. 5th Conference on Logic Programming & 5th Symposium on Logic Programming*, pp. 327–342, 1988.

[5] 能登正人, 栗原正仁, 大内東. 拡張ステータスによる項書換え系の停止性検証. 情報処理学会論文誌, pp. 2867–2871, 1995.

[6] Kristof Verschaetse and Stefaan Decorte. Automatic termination analysis. *Logic Program Synthesis and Transformation Proceedings of LOPSTR 92, International Workshop on Logic Program Synthesis and Transformation*, pp. 168–183, 1992.

[7] 大崎人士, アート・ミデルドープ, 井田哲雄. 意味ラベリングによる分配消去法. コンピュータソフトウェア, Vol. 13, No. 2, pp. 58–73, Mar. 1996.