プログラミング 16－1
（1997．11．21）

二分決定グラフを用いた書換え型プログラムの停止性検証器

近藤　久
茨城大学工学部システム工学科
〒316 日立市中成沢町 4-12-1
TEL：0294-38-5221
e-mail: *kondo@lily.dse.ibaraki.ac.jp*

栗原　正仁
北海道工業大学電気工学科
〒006 札幌市手稲区前田 7-15
TEL：011-681-2161（内線 455）
e-mail: *kurihara@hit.ac.jp*

概要：書換え型プログラムの理論的基礎として項書換え系がある．項書換え系に要求されることが多い重要な性質の一つとして停止性がある．本稿では，項書換え系の停止性の証明可能性を論理関数として表現し，その充足可能性を判定するために二分決定グラフを用いた停止性検証器を提案し，停止性検証のコーディング，基本アルゴリズムを示す．本手法は，停止性検証に経路順序と呼ばれる項の集合上の半順序を用いる．この順序は項を構成する演算子の集合上の半順序（優先順位）を項の集合上に拡張することによって得られる．ある優先順位$\succ$のもとで関数記号 f，gが $f \succ g$であることを論理変数 x_{fg}に対応させ，停止性の証明可能性を表す論理関数を効率良く表現するために二分決定グラフを用いて停止性を検証している．

Termination Verifier for Rewrite Rule Programs Using Binary Decision Diagram

Hisashi Kondo
Dept. of Systems Engineering, Faculty of Engineering, Ibaraki University
4-12-1 Naka-Narusawa, Hitachi 316, Japan
TEL：+81-294-38-5221
e-mail: *kondo@lily.dse.ibaraki.ac.jp*

Masahito Kurihara
Dept. of Electrical Engineering, Hokkaido Institute of Technology
7-15 Maeda Teine Ward, Sapporo 006, Japan
TEL：+81-11-681-2161 (ext. 455)
e-mail: *kurihara@hit.ac.jp*

Abstruct: We present a method of proving the termination of rewrite rule programs written as a set of rewrite rules (such programs are called term rewriting systems) by using binary decision diagrams (BDD's) for efficient representation of provability.

First, we give a recursive definition of the boolean function that computes the provability based on a partial ordering $\succ$ on the set of function symbols. Then the construction of the BDD's for this function, in which a primitive expression $f \succ g$ consisting of two function symbols f and g is associated with the logical variable x_{fg}, is incorporated into an interactive, incremental termination verification verifier.

1 はじめに

書換え型プログラムの理論的基礎として項書換え系（*term rewriting system*：TRS）[1, 2, 3]がある．TRSは書換え規則の集合により定義される計算システムである．その実行は書換え規則を与えられた項に繰り返し適用することによって行なわれる．人工知能の分野では定理自動証明や，プログラムの検証，合成などの対象としてTRSを用いた研究が活発に行なわれている．

TRSに要求されることが多い重要な性質の1つとして停止性がある．停止性の検証に最もよく用いられる方法は，経路順序（*path ordering*）と呼ばれるクラスに属する半順序$\succ$を適当に定め，すべての書換え規則について（左辺）$\succ$（右辺）の成立を確認することである．経路順序は関数記号の集合上の半順序（優先順位（*precedence*））を項の集合上に拡張して得られる．この優先順位を決定する問題は，経路順序の定義と（左辺）$\succ$（右辺）という条件を制約とする制約充足問題であり，一般にNP完全な問題となる．

制約充足問題の解法の一手法として，二分決定グラフ（*Binary Decision Diagram*：BDD）[5, 6]が注目を浴びている．BDDは論理関数をコンパクトに表現することができ，論理変数順序を固定し，重複節点の共有化，冗長節点の削除を行なうことによって論理関数に対する表現が一意に定まり標準形となるという特徴を持つ．さらに，論理関数が充足可能かどうかをただちに判定することができ，充足可能な場合には，論理関数を1とする入力割当を入力変数の数に比例する時間で求められる．

本稿では，対話的項書換え系作成支援システムに組み込むことを目的とした，経路順序とBDDを用いた停止性検証器を提案する．以下，**2**ではTRSと停止性及びBDDについて，**3**ではBDDを用いるための停止性検証のコーディング，基本手続き，変数順序，制約順序について述べる．**4**は結論である．

2 準備

2.1 項書換え系と停止性

演算子の集合を$\mathcal{F}$，変数の集合を$\mathcal{V}$とする．$\mathcal{F},\mathcal{V}$から構成される項の集合を$\mathcal{T}(\mathcal{F},\mathcal{V})$，$\mathcal{F}$のみから構成される項の集合を$\mathcal{T}(\mathcal{F})$とする．書換え規則（以後，ルールと記述）は$l \to r$の形をした$\mathcal{T}(\mathcal{F},\mathcal{V})$中の項の順序対である．但し，$l$は変数ではなく，$r$に含まれる変数は$l$にも含まれる．TRS：$\mathcal{R}$はルールの有限集合である．項$s$が$\mathcal{R}$中のある規則の一度の適用によって項$t$に書き換えられるとき，$s \to_{\mathcal{R}} t$と表す．$\mathcal{R}$は，もし$t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \cdots$のような無限の書換え列が存在しなければ停止する（*terminating*）といわれる．停止性は一般に決定不能な性質であるが経路順序と呼ばれるクラスの半順序に基づく十分条件が知られている．

本稿では，BDDを用いる方法の考え方を明確に述べるため，定義の簡潔さと効率の面から辞書式経路順序（*lexicographic path odering:lpo*）を用いるが，本手法の考え方は他の経路順序にも容易に拡張できる．

定義 1 (辞書式経路順序 [4]) $\succ$を固定個の引数をもつ関数記号の集合$\mathcal{F}$上の（厳格）半順序とする．項の集合$\mathcal{T}(\mathcal{F})$上の辞書式経路順序$\succ_{lpo}$は以下のように定義される．

項を$s \equiv f(s_1,\ldots,s_m)$，$t \equiv g(t_1,\ldots,t_n)$とする．このとき$s \succ_{lpo} t$であるのは以下のいずれかをみたすときに限る．

(1) $s_i \succeq_{lpo} t$ for *some* $i, 1 \le i \le m$
(2) $f \succ g$ and $s \succ_{lpo} t_j$ for *all* $j, 1 \le j \le n$
(3) $f = g$ and $\exists k, 1 \le k \le n$,
$s_i \equiv t_i$ for *all* $i, 1 \le i < k$,
$s_k \succ_{lpo} t_k, s \succ_{lpo} t_j$ for *all* $j, k < j \le n$

半順序$\succ$を優先順位（*precedence*）と呼ぶ．

$\succ_{lpo}$はさらに変数を含み得る項の集合$\mathcal{T}(\mathcal{F},\mathcal{V})$上に，以下のように拡張される．すなわち，$t \in \mathcal{T}(\mathcal{F},\mathcal{V})$を項，$x \in \mathcal{V}$を変数とするとき，

- xが項tの真部分項のとき，またそのときに限り，$t \succ_{lpo} x$.
- $x \not\succ_{lpo} t$.

定理 1 ある辞書式経路順序$\succ_{lpo}$に対しTRS:$\mathcal{R}$のすべてのルール$l \to r$が$l \succ_{lpo} r$ならば$\mathcal{R}$は停止する．

2.2 二分決定グラフ

BDDはグラフによる論理関数表現の一方法である．はじめAkersが表現法を提案し[5]，Bryantによって効率のよい演算法が提案された[6]．一般に，論理関数は変数の取る値によって木構造で表現することができる．この木表現に変数順位（変数の集合上の全順序）を導入し，終端節点，重複節点の共有化，冗長節点の削除を行なうことによって規約化した木構造を既約な順序付きBDD（*reduced ordered* BDD:ROBDD）と呼ぶ．以下，ROBDDを単にBDDと呼ぶ．

BDDの主な特徴は，

1. 変数順位を固定すれば，BDDは論理関数の標準形となる，
2. 多くの実用的に重要な論理関数が現実的な節点数で表現できる，
3. 論理関数に対する演算が表現のサイズに比例する時間で行なえる，

とまとめることができる．

BDDを用いる場合，特に重要なのは変数順序と制約順序である．変数順序の選択はBDDのサイズに影響を与えるが，与えられた論理関数のBDDのサイズを最小にする変数順序の決定はNP完全である．問題分野によっては，最適あるいは準最適な変数順序を決定する発見的な手法がいくつか提案されているが，これらの方法は一般的に共通したものではなく，問題に適した変数順序の計算法を考察しなければならない．

制約順序はBDD間の逐次的な合成の順序であり，同じ変数順序であっても制約順序が異なると，(最終的に得られるBDDは同じだが) 計算途中でのBDDのサイズに影響を与える．制約順序の計算法も，問題に応じて発見的な手法を考察しなければならない．

3 二分決定グラフを用いた停止性検証

3.1 停止性検証のコーディング

BDDは論理関数を表す手段であるため，$\succ_{lpo}$による停止性の証明可能性を論理関数として表現しなければならない．

$\mathcal{F}(\mathcal{R})$を$\mathcal{R}$中に現れる関数記号の集合とする．$X = \{x(f,g) | f,g \in \mathcal{F}(\mathcal{R})\}$をそれらの関数記号から作られる順序対$f \succ g$に対応した論理変数の集合とする．

$s \succ_{lpo} t$に対応する論理関数$lpo(s,t)$は定義1およびその直後に述べた拡張に基づき，以下のように再帰的に定義できる．

定義 2 (論理関数$lpo(s,t)$) $\mathcal{T}(\mathcal{F},\mathcal{V})$に属する2つの項を$s,t$とする．また，$s,t \notin \mathcal{V}$のときには，それぞれ，$s \equiv f(s_1,\ldots,s_m)$，$t \equiv g(t_1,\ldots,t_n)$とおく．

$lpo(s,t) =$

$$\begin{cases} 0, \text{ if } s \equiv t \text{ or } s \in \mathcal{V} \text{ or } \mathcal{V} \ni t \notin Var(s) \\ 1, \text{ if } t \in Var(s) \text{ or } s_i \equiv t \text{ for some } i \\ \sum_{i=1}^{m} lpo(s_i,t) + x(f,g) \cdot \\ \quad \prod_{j=1}^{n} lpo(s,t_j), \text{ if } f \neq g, \forall i. s_i \neq t \\ \sum_{i=1}^{m} lpo(s_i,t) + lpo(s_k,t_k) \cdot \\ \quad \prod_{j=k+1}^{n} lpo(s,t_j), \text{ if } f = g, \forall i. s_i \neq t \end{cases}$$

ただし，kは$s_i \neq t_i$をみたす最小のi $(1 \leq i \leq n)$である．$Var(s)$は項s中に出現している変数の集合である．

また，TRS:$\mathcal{R} = \{l_i \to r_i\}_{i=1}^{m}$が与えられたとき，すべてのルールについて$l_i \succ_{lpo} r_i$であることを示す論理関数は以下のように表すことができる．

$$lpo(\mathcal{R}) = \prod_{i=1}^{m} lpo(l_i,r_i) \qquad (1)$$

(1)式は，(厳格) 半順序$\succ$の性質，すなわち，**推移性** ($f \succ g$かつ$g \succ h$ならば$f \succ h$)，**非反射性** ($f \not\succ f$) を考慮していない．これらの性質を考慮に入れ，推移性と非反射性を論理関数としてコーディングしたものをそれぞれ，$t(X)$，$i(X)$とすると辞書式経路順序による$\mathcal{R}$の停止性の証明可能性は

$$T(\mathcal{R}) = lpo(\mathcal{R}) \cdot t(X) \cdot i(X) \qquad (2)$$

となる．ただし，

$$t(X) = \prod_{f,g,h \in \mathcal{F}} [\overline{x}(f,g) + \overline{x}(g,h) + x(f,h)$$

$$i(X) = \prod_{f \in \mathcal{F}} \overline{x}(f,f).$$

$\mathcal{R}$に現れる関数記号の数をnとすると，Xに含まれる変数はn^2個である．したがって，推移性条件の数はn^3となり，上記の$T(\mathcal{R})$において，$t(X)$が最も支配的な項になる傾向を持つ．しかし，実際に$lpo(l_i,r_i)(1 \leq i \leq m)$中に出現する変数は$X$の一部にすぎない．実際，$l_i$中に$f$，$r_i$中に$g$が出現しているときに限り，$lpo(l_i,r_i)$中に変数$x(f,g)$が出現する．以下では，このように実際に(1)式中に出現する変数の集合$X(\mathcal{R})$上で推移性条件と非反射性条件を定義する．特に，任意のfに対し，$x(f,f) \notin X(\mathcal{R})$であることに注意する．

TRS:$\mathcal{R}$に現れる関数記号の集合$\mathcal{F}(\mathcal{R})$を頂点の集合とし，論理式$lpo(\mathcal{R})$に現れる各論理変数$x(f,g) \in X(\mathcal{R})$を始点$f$から終点$g$への有向辺とするグラフ$G = (V,E)$，$V = \mathcal{F}(\mathcal{R})$，$E = X(\mathcal{R})$を考える．

定義 3 (推移性条件) グラフGの有向辺$x(f,g) \in E$の始点fから終点g ($f \neq g$) へ至る長さ2以上の経路

$$\begin{array}{l} f = v_0 v_1 \ldots v_n = g \\ \text{ただし，} \ n \geq 2, \\ (\forall i : 0 \leq i \leq n) v_i \in V, \\ (\forall j : 1 \leq j \leq n) x(v_{j-1}, v_j) \in E \end{array}$$

が存在するとき，含意

$$x(f,v_1) \wedge \ldots \wedge x(v_{n-1},g) \Rightarrow x(f,g)$$

あるいは，それを表す論理関数

$$\overline{x}(f,v_1) + \ldots + \overline{x}(v_{n-1},g) + x(f,g)$$

を，この経路に関する**推移性条件**と呼ぶ．また，このようなすべての有向辺および経路に関する推移性条件の論理積を**$X(\mathcal{R})$の推移性条件**と呼び，$t(X(\mathcal{R}))$と表記する．

実際には，すべての経路を考える必要はなく，少なくとも，同一頂点を2度以上通らない，経路の一部を置き換えるような有向辺を持たない，という2つの条件をみたす (極小の) ものを数えあげればよい (この2つの条件の詳細については紙面の関係上省略する)．

特に，グラフGが強完全グラフ，すなわち$E = \{x(f,g) \mid f \neq g\}$のときには，$X(\mathcal{R})$の推移性条件は互いに異なる3つの関数記号$f,g,h$のすべての組合せに対する条件

$$x(f,g) \wedge x(g,h) \Rightarrow x(f,h)$$

の論理積となる．

定義 4 (非反射性条件) グラフGにおける閉路

$$\begin{array}{l} v_0 v_1 \ldots v_n = v_0 \\ \text{ただし，} \ n \geq 2, \\ (\forall i : 0 \leq i \leq n) v_i \in V, \\ (\forall j : 1 \leq j \leq n) x(v_{j-1}, v_j) \in E \end{array}$$

の上の有向辺 $x(v_{i-1}, v_i)$ $(1 \leq i \leq n)$ の論理積の否定

$$\neg[x(v_0, v_1) \wedge \ldots \wedge x(v_{n-1}, v_0)]$$

すなわち，

$$\neg x(v_0, v_1) \vee \ldots \vee \neg x(v_{n-1}, v_0)$$

あるいは，それを表す論理関数

$$\overline{x}(v_0, v_1) + \ldots + \overline{x}(v_{n-1}, v_0)$$

をこの閉路に関する**非反射性条件**と呼ぶ．また，このようなすべての閉路に関する非反射性条件の論理積を**$X(\mathcal{R})$の非反射性条件**と呼び，$i(X(\mathcal{R}))$ と表記する．

推移性条件のときと同様に，実際には，すべての閉路を考える必要はなく，始点＝終点であることを除き，同一の頂点を 2 度以上通らない，閉路の一部を置き換えるような有向辺 (バイパス) を持たない，という 2 つの条件をみたす（極小の）ものだけを数えあげればよい（推移性条件と同様に条件の詳細に関しては省略する）．

特に，グラフ G が強完全グラフのときには，$X(\mathcal{R})$ の非反射条件は異なる 2 つの関数記号 f, g のすべての組合せに対する条件

$$\neg x(f, g) \vee \neg x(g, f)$$

の論理積となる．

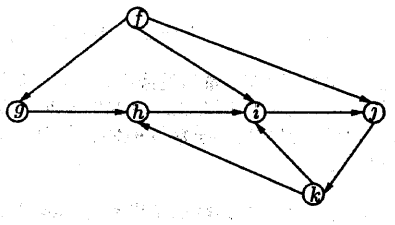

図 1: グラフ G

例 1 グラフ G が図 1のように与えられるとき，推移性条件は，

$$\begin{aligned}
\overline{x}(f,i) + \overline{x}(i,j) + x(f,j) \\
\overline{x}(k,h) + \overline{x}(h,i) + x(k,i) \\
\overline{x}(f,g) + \overline{x}(g,h) + \overline{x}(h,i) + x(f,i) \\
\overline{x}(f,j) + \overline{x}(j,k) + \overline{x}(k,i) + x(f,i)
\end{aligned}$$

非反射性条件は，

$$\overline{x}(i,j) + \overline{x}(j,k) + \overline{x}(k,i)$$

となる．ここで，推移性条件から $\overline{x}(f,j) + \overline{x}(j,k) + \overline{x}(k,h) + \overline{x}(h,i) + x(f,i)$，非反射性条件から $\overline{x}(h,i) + \overline{x}(i,j) + \overline{x}(j,k) + \overline{x}(k,h)$ が（推移性条件，非反射性条件について触れた，それぞれ 2 つの条件により）それぞれ削除されていることに注意する．

定義 3および 4を用いると，(2) 式と同様に次式を得る．

$$T(\mathcal{R}) = lpo(\mathcal{R}) \cdot t(X(\mathcal{R})) \cdot i(X(\mathcal{R})) \qquad (3)$$

グラフ G は実際には疎であることが多く，$X(\mathcal{R})$ 上の推移性条件と非反射性条件の数え上げと BDD の生成は実用的な時間と領域で実行できると期待される．

3.2 基本手続き

TRS : $\mathcal{R}$ の停止性を辞書式経路順序で証明可能かどうか，すなわち，$T(\mathcal{R})$ の充足可能性を判定する手続きを示す．ただし，対話的システムの場合には，利用者からの対話的入力として $\mathcal{R}$ が暗黙に与えられているとみなすこととし，また，充足不能と判定された場合は実際に手続きを終了するのではなく，警告を発してルールを再入力させるなど，対話的処理を続行するものとする．

この手続きでは以下の大域変数を用いている．

- R: これまで入力されたルールの集合．初期値は空．(全入力を表す $\mathcal{R}$ とは字体を変えていることに注意．)
- $X(R)$: $lpo(R)$ 中に出現する論理変数の全順序集合．初期値は空．
- $bdd(R)$: 論理関数 $T(R)$ の BDD 表現．初期値は論理値 1 を表す BDD．

手続きは以下のステップを終了まで反復する．

1. $\mathcal{R} = \emptyset$ なら終了．$T(\mathcal{R})$ は充足可能であり，$\mathcal{R}$ は停止性をもつ．
2. $\mathcal{R}$ から任意の 1 本のルール $l \to r$ を取り除き，$R' = R \cup \{l \to r\}$ とする．$lpo(l, r)$ 中に生起している論理変数のうち，$X(R)$ に含まれていないものの集合をΔXとする．また，$X(R') = X(R) \cup \Delta X$に変数順序を導入し全順序集合とする．
3. $lpo(l, r)$ の BDD を Δlpo とする．また，ΔX 中の変数を少なくとも 1 つ含むような極小な推移性条件および非反射性条件をすべて求め，その論理積の BDD をそれぞれΔt, Δi とする．
4. 4 つの BDD: $bdd(R)$, Δlpo, Δt, Δi の論理積を表す BDD を $bdd(R')$ とする．もし，それが論理値 0 に対応するものなら終了．$T(\mathcal{R})$ は充足不能である．
5. R', $X(R')$, $bdd(R')$ を，それぞれ新たに R, $X(R)$, $bdd(R)$ とする．

この手続きは，$T(R)$ が充足可能であることを不変に保ちながら，ルールを 1 本処理する毎に論理式 $lpo(R)$, $t(X(R))$, $i(X(R))$ の変化分 Δlpo, Δt, Δi のみをインクリメンタルに求め，これまでの $T(R)$ に論理積として付け加えている．

この手続きは以下の 3 つの点で非決定的である．

1. ステップ 2 でのルールの選択が任意.
2. ステップ 2 で与える論理変数の全順序が任意.
3. ステップ 4 での 4 つの BDD の論理積を計算する順序が任意.

実際には，この任意性をどうインプリメントするかが効率に影響を与え得る．BDD 技術の一般的な言葉で述べると，(2) は**変数順序**，(1) および (3) は**制約順序**をどう決めるかという問題となる．本稿では，それぞれ以下のように対処することとする．

1. ルールの選択の順序は利用者が入力した順序に固定する．
2. 変数順序については，次節で述べるようないくつかの簡単なヒューリスティクスにより，これまでの順序をインクリメンタルに拡張するものとする．その中でも,「生起逆順」を推奨する．
3. 4 つの BDD の積の求め方の場合の数を，積が交換可能であることを考慮して求めると，$((a\cdot b)\cdot c)\cdot d$ のように線形順に積をとるのが 12 通り，$(a\cdot b)\cdot(c\cdot d)$ のように分割して積をとるのが 6 通りで，合計 18 通りあるが，$((bdd(R)\cdot \Delta lpo)\cdot \Delta i)\cdot \Delta t$ の順を推奨する．

例 2 2 本の書換え規則からなる TRS:$\mathcal{R}=\{f(h(x))\to g(x), g(i(x))\to f(x)\}$ を考える．図 2に得られた BDD を示す．変数 x_1,x_2,x_3,x_4に対して根から論理値 1 へ至るパスを辿り，それぞれの枝にラベル付けされた論理値を割り当てることによって解が得られる．

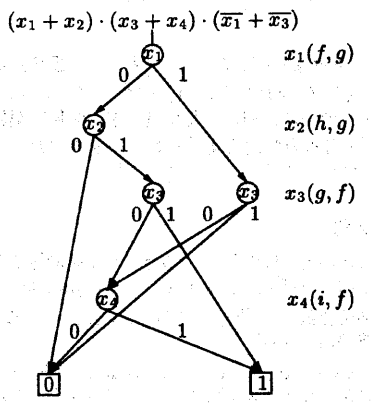

図 2: 得られた BDD

3.3 変数順序

最適な変数順序を求めることは NP 完全であるため，ここではヒューリスティックな順序の決め方のみに限定して議論する．特に，実用の観点から，インクリメンタルに拡張できる順序だけを考える．すなわち，新たな変数が出現したときには，現時点での順序関係を崩さないように変数順序を拡張する．例えば，ある時点で $x(f,g)<x(i,j)$ であったのが，その後，ルールを付加するなどした別の時点で $x(f,g)>x(i,j)$ と変更されることは許さない．

このような条件をみたす変数順序として，以下のものを検討した．

- **ランダム順**：新たな変数が生起するごとに，その変数に $[0,1]$ 内の乱数を割り当て，変数順序をこの乱数の昇順とする．乱数値が等しい場合は，さらに乱数を生成してタイブレークする．この順序が有効なものとは思われないが，他の順序の有効性を評価する際の比較基準として用いる．
- **生起順**：変数が生起した順序をそのまま変数順序として用いる．すなわち，定義 2の論理式を再帰的に左から深さ優先で展開して，各変数が生起した順序を変数順序とする．この順序はインプリメンテーションの点で最も自然なものである．
- **左優先順**：関数記号の集合 $\mathcal{F}(\mathcal{R})$ 上に 2 つの全順序$<_L$と$<_R$を以下のように定義する．

  $f<_L g \Leftrightarrow$
  fは gよりもルールの左辺中に先に出現
  $f<_R g \Leftrightarrow$
  fは gよりもルールの右辺中に先に出現

  これらの全順序に基づき，変数順序を以下のように定める．

  $$x(f,g) < x(i,j) \Leftrightarrow$$
  $$f<_L i \quad \text{or} \quad (f=i \text{ and } g<_R j)$$

- **右優先順**：左優先順と同様の考え方であるが，関数記号が右辺に生起した順序を優先し，変数順序を以下のように定める．

  $$x(f,g) < x(i,j) \Leftrightarrow$$
  $$g<_R j \quad \text{or} \quad (g=j \text{ and } f<_L i)$$

ランダム順を除く上記の変数順序を全く逆にした順序も考察することとし，それぞれ，**生起逆順**，**左優先逆順**，**右優先逆順**と呼ぶ．

例 3 TRS:$\mathcal{R}=\{f(g(x))\to h(x), h(f(x))\to g(x)\}$ がこの順に与えられたとする．このとき，$X(\mathcal{R})=\{x(f,h),x(g,h),x(h,g),x(f,g)\}$ なので，変数の数は 4，可能な変数順序は 24 通りある．

ランダム順を除く上記の 6 通りの変数順序を図 3に示す．図では，縦方向には$<_L$，横方向には$<_R$の順に関数記号を並べてあり，一般に，変数順序 $x(f_1,g_1)<x(f_2,g_2)<\ldots<x(f_n,g_n)$ を表現するために，f_i行 g_i列に整数値 i を記入してある $(1\le i\le n)$．たとえば，生起順では $x(g,h)<x(f,h)<x(f,g)<x(h,g)$ となる．

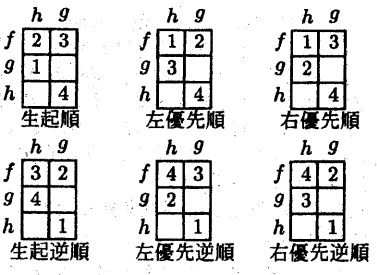

図 3: 例題の変数順序

3.4 制約順序

制約順序に関わる問題としては，すでに述べたように，4つの BDD の積を求める順序があるが，もっとグローバルな視点から眺めると，インクリメンタルな停止性検証法を目的とする **3.2**の基本手続きの枠組自体が制約順序をすでに規定していることに注意する．なぜなら，この枠組では，ルールを 1 本読み込む毎に，そのルールからインクリメンタルに得られる 3 種類の制約を現在の BDD に加えていっているからである．すなわち，第 j番目のルール入力から得られる制約を $(\Delta lpo)_j, (\Delta t)_j, (\Delta i)_j (1 \leq j \leq m)$ とすると，この手続きは，

$$\begin{aligned} T(\mathcal{R}) \quad = \quad & (\Delta lpo)_1 \cdot (\Delta i)_1 \cdot (\Delta t)_1 \cdot \\ & \cdots \cdot (\Delta lpo)_m \cdot (\Delta i)_m \cdot (\Delta t)_m \end{aligned}$$

の積を左から計算している．明らかに，これは (3) 式による定義

$$T(\mathcal{R}) = lpo(\mathcal{R}) \cdot t(X(\mathcal{R})) \cdot i(X(\mathcal{R}))$$

を素直に計算した場合と制約順序が異なる．前者を「漸増方式」，後者を「一括方式」と呼び，この両者の効果については研究会当日に述べる．

3.5 推移性条件の削除

推移性条件は $X(\mathcal{R})$ のサイズが大きい場合，非常に大きな論理式となることがあるので，効率上の問題が生じる．以下では，推移性条件を考慮せず，非反射性条件のみを考慮する方法を示す．この場合，求まる解は $X(\mathcal{R})$ 上の推移性条件を満たさなくなるが，その推移的閉包は（推移的閉包の定義から）X 上の推移性条件をみたす．ただし，非反射性条件を定義する際に考察する閉路としては極小のものばかりではなく，バイパスを持つものも列挙する必要がある．なぜなら，すでに述べたように，バイパスを持つ閉路を考えなくてもよい根拠は推移性条件によるものだからである．このように，非反射性条件の数は増えるが，それ以上に推移性条件の数の減少が効果的である場合が多い．

バイパスを持つ閉路を含めて構成して得られる非反射性条件を $I(X(\mathcal{R}))$ とする．このとき，$\mathcal{R}$ の lpo による停止性の判定は

$$T(\mathcal{R}) = lpo(\mathcal{R}) \cdot I(X(\mathcal{R}))$$

の充足可能性に帰着される．

3.2と同様に，これをインクリメンタルに計算する手続きを構成できる．

4 おわりに

本稿では，BDD を用いた TRS の停止性検証器について述べた．本手法は BDD により停止性を保証する優先順位を求める新たな方法であり，停止性のコーディング，基本手続き，変数順序，制約順序について述べた．紙面の関係上，実験結果を省略したが，基本手続きに基づく停止性検証器と **3.5** の変更をおこなった検証器を実現し，多くの計算機実験を行なっている．実験結果から，変数順序として生起逆順，制約順序として $((bdd(R) \cdot \Delta lpo) \cdot \Delta i) \cdot \Delta t$ が効率良く停止性を検証できることを確認している．さらに，**3.5**の変更を行なうことにより，より効率が改善できることを実験結果より得ている．詳細な実験結果については研究会当日に述べる．対話的 TRS 作成支援システムへ停止性検証器を組み込む場合，本手法を用いることによって対話性を犠牲にすることなく停止性の検証を行なうことができると考えられる．

今後の課題として少なくとも以下の 3 点が挙げられる．

- ルールの詳細な構造解析などに基づく、変数順序の新しいヒューリスティクスの考案と評価．
- ルールの入力順に依存する制約順序の効率への影響に関する考察．
- 他の経路順序への拡張とその実用性の考察．

謝辞

第 1 著者は本研究の一部に対し，文部省科学研究費補助金（#09780231）による支援を受けている．第 2 著者は本研究の一部に対し，北海道工業大学特別奨励研究助成金及び文部省科学研究費補助金（#09650444）による支援を受けている．

参考文献

[1] 二木厚吉，外山芳人：項書き換え型計算モデルとその応用，情報処理，Vol. 24, No. 2, pp.133–146(1983).

[2] 井田哲雄：計算モデルの基礎理論，pp.223–296，岩波書店 (1991).

[3] Dershowitz, N. and Jouannaud, J.-P. : 書換え系，コンピュータ基礎理論ハンドブック II：形式的モデルと意味論，pp.243–320，丸善 (1994).

[4] Dershowitz, N. : Termination of Rewriting, *J. Symbolic Computation*, Vol. 3, pp.69–116(1987).

[5] Akers, S. B. : Binary Decision Diagrams, *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp.509–516(1978).

[6] Bryant, R. E. : Graph-based Algorithm for Boolean Function Manipulation, *IEEE Trans. Comput.*, Vol. C-35, No. 5, pp.677–691(1986).