項書き換えシステム停止性検証における SAT ソルバ利用のための新しい符号化法

Encodings for Proving Termination of Term Rewriting Systems using SAT Solvers

五十嵐圭一　青戸等人　外山芳人

Keiichi Igarashi　Takahito Aoto　Yoshihito Toyama

東北大学 電気通信研究所

RIEC, Tohoku University

{igarashi,aoto,toyama}@nue.riec.tohoku.ac.jp

項書き換えシステムの停止性については，さまざまな検証法が研究されるとともに自動検証システムの開発が行われている．近年，項書き換えシステムの停止性自動検証システムの研究において，SAT ソルバを用いた効率的な実装が注目されている．本研究では，経路順序に基づく停止性条件の判定に SAT ソルバを利用するための新しい符号化法を考案するとともに，従来手法との比較実験を行なったので報告する．

1 はじめに

項書き換えシステムの停止性の検証は，項書き換えシステムに基づく定理自動証明やプログラムの自動検証において極めて重要である．このため，様々な停止性検証法が提案されており，AProVE[4] や TTT[6] などの停止性自動検証システムも開発されている．一方，近年，大規模なハードウェアやソフトウェアの自動検証技術として SAT ソルバを利用した検証技術が注目されている．項書き換えシステムの停止性についても，SAT ソルバを利用した検証技術が提案されている [2, 9].

項書き換えシステムの停止性検証法のうち，最も基本的な手法として辞書式経路順序に基づく停止性検証法がある．辞書式経路順序は，関数記号上の優先順位に基づき生成される項上の順序であり，適当な優先順位の元で全ての書き換え規則において左辺が右辺より大きくなる時，項書き換えシステムの停止性を保証することが出来る．Codish ら [2] は，辞書式経路順序による停止性検証において，SAT ソルバを利用することにより，それまで開発されていた停止性自動検証システムより良好なパフォーマンスが得られることを報告した．

SAT ソルバとは，命題論理式の充足可能性を判定するシステムである．SAT ソルバを利用するためには，問題を論理式に符号化する必要がある．辞書式経路順序による停止性判定問題を命題論理式に符号化する手法として，近藤&栗原 [7] による原子符号化 (atom-based encoding) がある．文献 [2] において，Codish らは，記号符号化 (symbol-based encoding) を提案し，原子符号化に対する優位性を論じている．

辞書式経路順序の拡張として，置換付き辞書式経路順序 (lexicographic path ordering with permutation) がある．置換付き辞書式経路順序では，関数記号上の優先順位に加えて，それぞれの関数記号に対する引数の置換をパラメータとして，項上の順序を生成する．置換の符号化として佐藤&栗原 [8] による方法が知られている．Schneider-Kamp らは，最近，Codish らの結果を置換付き辞書式経路順序へ拡張している [9] が，ここでも，同様の符号化法が採用されている．

本論文では，まず，優先順位の原子符号化および記号符号化に基づく辞書式経路順序の実験を行い，実際にどの程度のパフォーマンスの違いがあるかを確認する．次に，Codish らが提案した記号符号化が，置換の符号化にも利用可能であることを指摘し，記号符号化に基づく置換の符号化を提案する．そして，置換を SK 式符号化および記号符号化に基づいて符号化した置換付き辞書式経路順序について，その比較実験を行い，その結果を報告する．

2 項書き換えシステムの停止性と辞書式経路順序

書き換え規則の集合を**項書き換えシステム**とよぶ．以下は，2 つの書き換え規則からなる項書き換えシステムである．

$$\mathcal{R}\left\{\begin{array}{lcl} +(0,y) & \to & 0 \\ +(\mathsf{s}(x),y) & \to & \mathsf{s}(+(x,y)) \end{array}\right.$$

整数 $0,1,2,\ldots$ が $0,\mathsf{s}(0),\mathsf{s}(\mathsf{s}(0)),\ldots$ により表現されるとき，$\mathcal{R}$ は整数の加算を計算する．書き換えシステムにおける計算とは，書き換え規則の適用が出来なくなるまで簡約を繰り返すことをいう．例えば，$2+1=3$ の計算は次の簡約列により実行される：$+(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)) \to \mathsf{s}(+(\mathsf{s}(0)),\mathsf{s}(0)) \to \mathsf{s}(\mathsf{s}(+(0,\mathsf{s}(0)))) \to \mathsf{s}(\mathsf{s}(\mathsf{s}(0)))$．

上記の項書き換えシステム $\mathcal{R}$ では，計算ステップが必ず停止し解が求まる．しかしながら，一般には，任意の項書き換えシステムでそうなるわけではない．項書き換えシステムが**停止性**を持つとは，無限に続く簡約が存在しないこと，つまり，どんな項に対しても有限ステップで計算結果が求まることをいう．

項書き換えシステムの停止性を示すためのさまざまな手法が知られている．なかでも，基本的な方法の 1 つは**辞書式経路順序**を利用する方法である [1]．この方法は，関数記号上の**優先順位**を定め，優先順位により生成された辞書式経路順序で，全ての書き換え規則において左辺が右辺よりも大きいことを示すことにより，項書き換えシステムの停止性を保証する．

例えば，上に示した項書き換えシステム $\mathcal{R}$ では，$+ > \mathsf{s} > 0$ と優先順位を定める．項のルート記号 $+,0$ の比較により，$+(0,y) > 0$ となり，また，項のルート記号 $+,\mathsf{s}$ の比較により，$+(\mathsf{s}(x),y)$ と $\mathsf{s}(+(x,y))$ の大小は，$+(\mathsf{s}(x),y)$ と $+(x,y)$ の大小に帰着される．今度は，項のルート記号が両方とも $+$ であるため，引数同士の比較に帰着され，$\mathsf{s}(x)$ と x，y と y の比較になるが，$\mathsf{s}(x) > x$ が成立するため，結局，左辺が右辺より大きいことが導かれる．関数記号が同じ場合の引数同士の比較は通常は左から順に行うが，この順番は関数記号に応じて定めることも可能である (置換付き辞書式経路順序)．

以下に，優先順位 $>$ に基づく辞書式経路順序 $>_{lpo}$ の定義を示す [1]．$s >_{lpo} t \Leftrightarrow$

1. $t \in V$, $s \notin V$ かつ $t \in V(s)$,
2. $s = f(s_1,\ldots,s_n)$ かつ $\exists i.s_i \geq_{lpo} t$,
3. $s = f(s_1,\ldots,s_n)$, $t = g(t_1,\ldots,t_m)$, $f > g$ かつ $\forall i.s > t_i$, または
4. $s = f(s_1,\ldots,s_n)$, $t = f(t_1,\ldots,t_n)$, $\forall i.s >_{lpo} t_i$ かつ $\langle s_1,\ldots,s_n\rangle \gg_{lex} \langle t_1,\ldots,t_n\rangle$.

ここで，辞書式比較 $\langle s_1,\ldots,s_n\rangle \gg_{lex} \langle t_1,\ldots,t_n\rangle$ は以下により再帰的に定義される．

1. $s_1 >_{lpo} t_1$, または
2. $s_1 = t_1$ かつ $\langle s_2,\ldots,s_n\rangle \gg_{lex} \langle t_2,\ldots,t_n\rangle$.

上記の辞書式比較は，引数比較を左から右にした場合であるが，関数記号に応じて引数比較の順番を変えることも出来る．その場合は，関数記号 f それぞれに対する置換 $\pi_f : \{1,\ldots,n\} \to \{1,\ldots,n\}$ を与え，定義の第 4 項の $\gg_{lex}$ を以下により定義される $\gg_{lex(\pi_f)}$ と置き換える：$\langle s_1,\ldots,s_n\rangle \gg_{lex(\pi_f)} \langle t_1,\ldots,t_n\rangle \Leftrightarrow \langle s_{\pi_f(1)},\ldots,s_{\pi_f(n)}\rangle \gg_{lex} \langle t_{\pi_f(1)},\ldots,t_{\pi_f(n)}\rangle$．このようにして拡張された辞書式経路順序を**置換付き辞書式経路順序**とよぶ．以下では，関数記号 f それぞれに対する置換 $(\pi_f)_{f\in\mathcal{F}}$ に基づく置換付き辞書式経路順序を $>^{\pi}_{lpo}$ と書く．

命題 1 ([1]) ある優先順位 $>$ と関数記号それぞれに対し定めた置換 $(\pi_f)_{f\in\mathcal{F}}$ が存在して，任意の書き換え規則 $l \to r \in \mathcal{R}$ について，$l >^{\pi}_{lpo} r$ が成立するとき，項書き換えシステム $\mathcal{R}$ は停止性を持つ．

3 辞書式経路順序の符号化と充足可能性問題への帰着

辞書式経路順序に基づく項書き換えシステム $\mathcal{R}$ の停止性検証のためには，適切な優先順位および適切な置換に対して，書き換え規則が順序付けられることをいえばよい．従って，書き換え規則が順序つけられるような優先順位や置換の存在性の検証をすることになる．この存在問題は，命題論理式の充足可能性問題に還元することが可能であり，従って，SAT ソルバを利用して解くことが可能である．

命題論理式への符号化には，素となる命題である優先順位や置換の符号化法が欠かせない．この部分の符号化については次節以降で説明する．優先順位の符号化法 e に基づく (置換付きでない) 辞書式経路順序の符号化 $[\![\_ >_{lpo} \_]\!]_e$ は以下により与えられる．

1. $s \in V$ のとき．$[\![s >_{lpo} t]\!]_e = \bot$

2. $t \in V$ のとき．$[\![s >_{lpo} t]\!]_e =$

$$\begin{cases} \top & (s \neq t \text{ かつ } t \in V(s) \text{ のとき}) \\ \bot & (\text{それ以外}) \end{cases}$$

3. $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$ $(f \neq g)$ のとき．$[\![s >_{lpo} t]\!]_e =$

$$\bigvee_{i=1}^{n} [\![s_i \geq_{lpo} t]\!]_e \vee (e\,(f > g) \wedge \bigwedge_{i=1}^{m} [\![s >_{lpo} t_i]\!]_e)$$

4. $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ のとき．$[\![s >_{lpo} t]\!]_e =$

$$\bigvee_{i=1}^{n} [\![s_i \geq_{lpo} t]\!]_e \vee (\bigwedge_{i=1}^{n} [\![s >_{lpo} t_i]\!]_e \wedge [\![\langle s_1, \ldots, s_n\rangle \gg_{lex} \langle t_1, \ldots, t_n\rangle]\!]_e)$$

また，$[\![\langle s_1, \ldots, s_n\rangle \gg_{lex} \langle t_1, \ldots, t_n\rangle]\!]_e =$

$$\begin{cases} \bot & (n = m = 0) \\ [\![s_1 >_{lpo} t_1]\!]_e & (s_1 \neq t_1) \\ [\![\langle s_2, \ldots, s_n\rangle \gg_{lex} \langle t_2, \ldots, t_n\rangle]\!]_e & (s_1 = t_1) \end{cases}$$

置換付き辞書式経路順序の場合も同様にして命題論理式に符号化することが出来る [8]．優先順位および置換の符号化法 e に基づく (置換付き/置換なし) 辞書式経路順序による停止性判定問題は

$$COND_e \wedge \bigwedge_{l \to r \in \mathcal{R}} [\![l >_{lpo}^{\pi} r]\!]_e$$

という形の命題論理式の充足可能性問題に帰着される．ここで，論理式 $COND_e$ はそれぞれの優先順位や置換の符号化法 e により必要となる追加条件を符号化した論理式で，必要ない場合は $COND_e = \top$ とおく．

4 優先順位の符号化法とその比較実験

前節で優先順位の符号化法 e に基づく辞書式経路順序による停止性検証問題の命題論理式への符号化について説明した．優先順位の符号化法として，近藤&栗原 [7] による原子符号化 (atom-based encoding) と Codish ら [2] による記号符号化 (symbol-based encoding) が知られている．本節では，この 2 つの符号化法について説明するとともに，比較実験を行う．また，参考として，比較的単純な符号化法 (マス式符号化) を 1 つ与え，それとの比較実験も行う．なお，以下では，関数記号集合を $\mathcal{F} = \{f_1, \ldots, f_{|\mathcal{F}|}\}$ とし，優先順位として全順序のみを考える．

4.1 原子符号化

近藤&栗原 [7] において使用されている原子符号化について説明する．なお，原子符号化の名前は文献 [2] による．原子符号化では 2 つの関数記号の大小に対して論理変数を割り当てる．論理変数集合として $\{x_{ij} \mid 0 < i < j \leq |\mathcal{F}|\}$ を用いる．論理変数 x_{ij} $(i < j)$ は $f_i > f_j$ を表わすときに 1，そうでないとき (すなわち $f_j > f_i$ のとき) は 0 となると考える．

$f_i > f_j$	f_1	f_2	$\cdots$	$f_{\lvert\mathcal{F}\rvert-1}$	$f_{\lvert\mathcal{F}\rvert}$
f_1	$-$	x_{12}	$\cdots$	$x_{1(\lvert\mathcal{F}\rvert-1)}$	$x_{1\lvert\mathcal{F}\rvert}$
f_2	$-$	$-$	$\cdots$	$x_{2(\lvert\mathcal{F}\rvert-1)}$	$x_{2\lvert\mathcal{F}\rvert}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$f_{\lvert\mathcal{F}\rvert-1}$	$-$	$-$	$\cdots$	$-$	$x_{(\lvert\mathcal{F}\rvert-1)\lvert\mathcal{F}\rvert}$

すなわち，優先順位の原子符号化 e は次のようになる．

$$e(f_i > f_j) = \begin{cases} x_{ij} & (i > j) \\ \neg x_{ij} & (i < j) \end{cases}$$

$>$ が全順序であることを保証するため，原子符号化では，推移律を表わす以下の命題が $COND_e$ として必要となる．

$$\bigwedge_{\substack{1 \leq i,j,k \leq |\mathcal{F}| \\ i \neq j \neq k \neq i}} \neg e(f_i > f_j) \vee \neg e(f_j > f_k) \vee e(f_i > f_k)$$

4.2 記号符号化

Codish ら [2] により提案された記号符号化について説明する．記号符号化では，$f_1, \ldots, f_{|\mathcal{F}|}$ のそれぞれに $0, 1, \ldots, |\mathcal{F}| - 1$ までの重み $w(f_i)$ を付け，$f_i > f_j \Leftrightarrow w(f_i) > w(f_j)$ と約束する．一方で，$0, 1, \ldots, |\mathcal{F}| - 1$ までの重みを長さ $m = \lceil \log |\mathcal{F}| \rceil$ の 2 進数で表わし，この 2 進数のそれぞれの桁のビットに対応する論理変数を用意する．つまり，論理変数集合は $\{x_{ij} \mid 1 \leq i \leq |\mathcal{F}|, 1 \leq j \leq m\}$ となる．

	$w(f_i)$ の 2 進数表現
f_1	$x_{11}\ x_{12}\ \cdots x_{1m}$
f_2	$x_{21}\ x_{22}\ \cdots x_{2m}$
$\vdots$	$\cdots\cdots\cdots\cdots\cdots$
$f_{\lvert\mathcal{F}\rvert}$	$x_{\lvert\mathcal{F}\rvert 1}\ x_{\lvert\mathcal{F}\rvert 2}\ \cdots x_{\lvert\mathcal{F}\rvert m}$

このとき，ビット列 $x_{11}\ x_{12}\ \cdots x_{1m}$ は自然数の重みを 2 進数化したものであるから，k ビット目で

$x_{ik}=1, x_{jk}=0$，それより高次のビット $l\,(<k)$ では $x_{il}=x_{jl}$ となるような k が存在すれば，$f_i > f_j$ となる．従って，優先順位の 2 進数符号化は $e(f_i > f_j) =$

$$\bigvee_{1\le k\le|\mathcal{F}|}(x_{ik}\wedge\neg x_{jk}\wedge\bigwedge_{0<l<k}(x_{il}\leftrightarrow x_{jl}))$$

により与えられる．自然数で重み付けしているため，$>$ は自動的に全順序となる．従って，$COND_e=\top$ とおけばよい．

4.3 マス式符号化

マス式では $\{x_{ij}\mid 1\le i,j\le|\mathcal{F}|\}$ の変数を用意し，$x_{ij}=1 \Leftrightarrow$ 関数記号 f_i は j 番目に大きい関数記号であるとき，と約束する．従って，マス式符号化は $e(f_i > f_j) =$

$$\bigvee_{1\le k\le|\mathcal{F}|}(x_{ik}\wedge\bigwedge_{0<l<k}\neg x_{jl})$$

により与えられる．j 番目に小さい関数記号は唯 1 つに，また，関数記号の重みは唯 1 つになる必要があるため，マス式符号化ではこれらの一意性を示す次の命題を $COND_e$ として追加する必要がある．

$$(\bigwedge_{1\le j\le|\mathcal{F}|}\mathrm{ones}(\{x_{ij}\mid 1\le i\le|\mathcal{F}|\}))\wedge(\bigwedge_{1\le i\le|\mathcal{F}|}\mathrm{ones}(\{x_{ij}\mid 1\le j\le|\mathcal{F}|\}))$$

ここで，ones(P) は，P 中の命題のうち丁度 1 つが真となることを符号化した命題とする．

4.4 実装および比較実験

優先順位の符号化に 3 つの符号化法を用いた辞書式経路順序に基づく停止性検証プログラムを関数型プログラム言語 SML/NJ を用いて実装した．SAT ソルバは minisat を利用し，論理式の論理積標準形への変換には文献 [5] における変換アルゴリズム ct^{**} を用いた．

表 1 に，Termination Problem Database[1] (TPDB) からの例について，停止性検証実験を行った結果を示す．Codish/から始まる行は，文献 [2] で報告されている検証に時間を要する例についての実行時間，Codish/*はその合計時間を示している．また，SK/* には，経路順序による停止性のベンチマークとして使用されることを多い文献 [10] の例題集について検証を試みた合計時間を示す．また，aprove/*には，文献 [9] と同じ実験例についての合計時間を示した．全て時間は実行に要した CPU 時間であり，計測には 1.2GHz Pentium プロセッサおよび 1G バイトのメモリを搭載した PC を用いた．

実験の結果，原子符号化と記号符号化で，ほぼ同様のパフォーマンスが得られた．ただし，大規模な例 (Codish/Cime_mucrl1) については，記号符号化がはるかに良いパフォーマンスを示しており，大規模な例に対する記号符号化の利点が確認された．また，原子符号化と記号符号化も，単純な符号化法であるマス式と比較するとはるかに良いパフォーマンスを示すことが確認できた．

5 置換の符号化法とその比較実験

置換付き辞書式経路順序を符号化するには，関数記号の優先順位に加えて，関数記号それぞれに対する置換を符号化する必要がある．各関数記号 f に対する置換は，f のアリティを $ar(f)$ と表わすとき，全単射 $\pi_f : \{1,\ldots,ar(f)\}\to\{1,\ldots,ar(f)\}$ である．本節では，まず，文献 [8, 9] で用いられている置換の符号化について説明し，次に，置換の記号符号化について提案する．更に，比較のために考案した方法である隣接式について説明する．

5.1 佐藤&栗原による符号化

佐藤&栗原および Codish ら [8, 9] で用いられている符号化 (以下，SK 式符号化とよぶ) について説明する．SK 式符号化では，$\bigcup_{f\in\mathcal{F},ar(f)\ge 2}\{x^f_{ij}\mid 1\le i,j\le ar(f)\}$ の論理変数を用いる．$x^f_{ij}=1\Leftrightarrow$ (関数記号 f の) 第 i 引数は j 番目に比較される，と約束する．このとき，j 番目に比較される引数は唯 1 つに定まり，各引数が比較されるのは 1 度だけ，という条件を $COND_e$ に追加する．

$$\bigwedge_{\substack{f\in\mathcal{F}\\ ar(f)\ge 2}}(\bigwedge_{1\le j\le|\mathcal{F}|}\mathrm{ones}(\{x^f_{ij}\mid 1\le i\le|\mathcal{F}|\})\wedge\bigwedge_{1\le i\le|\mathcal{F}|}\mathrm{ones}(\{x^f_{ij}\mid 1\le j\le|\mathcal{F}|\}))$$

[1] http://www.lri.fr/~marche/tpdb/

優先順位の符号化	原子	記号	マス式	記号	記号	記号
辞書置換の符号化	-	-	-	SK 式	記号	隣接式
Codish/Zantema_z30	0	2	2	161	180	163
Codish/Cime_mucrl1	750	86	-	1057	1009	1117
Codish/currying_AG01_No_3.13	9	7	77	399	427	400
Codish/higher-order_Bird_Hamming	8	7	78	146	157	146
Codish/HM_t005	18	35	359	432	425	421
Codish/HM_t009	43	107	1907	112	191	126
Codish/TRCSR_Ex1_2_AEL03_C	37	64	1323	71	76	81
Codish/TRCSR_Ex1_2_AEL03_GM	47	37	1317	40	39	48
Codish/TRCSR_Ex26_Luc03b_C	17	30	467	35	35	37
Codish/TRCSR_Ex2_Luc02a_C	17	32	457	34	36	35
Codish/TRCSR_Ex3_3_25_Bor03_C	12	24	225	29	29	31
Codish/TRCSR_Ex4_7_37_Bor03_C	15	23	292	29	29	31
Codish/TRCSR_Ex5_7_Luc97_C	32	65	1189	70	72	72
Codish/TRCSR_Ex6_15_AEL02_C	55	100	2747	108	121	121
Codish/TRCSR_Ex6_15_AEL02_FR	66	37	1961	44	43	47
Codish/TRCSR_Ex6_15_AEL02_GM	97	48	3040	58	55	63
Codish/TRCSR_Ex6_15_AEL02_Z	68	34	1898	42	38	46
Codish/TRCSR_Ex7_BLR02_C	15	23	285	24	25	26
Codish/TRCSR_Ex8_BLR02_C	11	22	220	28	27	28
Codish/TRCSR_Ex9_BLR02_C	12	35	229	33	38	33
Codish/TRCSR_ExAppendixB_AEL03_C	44	72	1624	80	80	87
Codish/TRCSR_ExIntrod_GM99_C	27	53	909	58	59	63
Codish/TRCSR_ExIntrod_Zan97_C	16	25	410	30	31	31
Codish/TRCSR_ExSec11_1_Luc02a_C	22	37	634	41	41	42
Codish/AProVE_AAECC-ring	85	43	2471	98	44	168
Codish/* の合計	1528	1042	-	3262	3220	3472
SK90/* の合計	247	293	1214	407	380	493
Aprove/* の合計	5307	4640	93145	10087	9548	11257

(単位 msec.)

表 1: SAT ソルバを利用した辞書式経路順序による停止性検証

5.2 記号符号化

$i > j \Leftrightarrow$ 第 i 引数は第 j 引数より先に比較される，という順序を考えると，置換の符号化に対しても，優先順位の符号化で用いた記号符号化がそのまま利用できることがわかる．f の引数 i に $0, \ldots, ar(f)-1$ までの重み $w(i)$ をつける．f の引数の比較は，重みの大きい順に行う．つまり，$i > j \Leftrightarrow w(i) > w(j)$ とする．また，重みを 2 進数で表現し，そのそれぞれのビットを論理変数に対応させる．つまり，用意する論理変数は，$m = \lceil \log ar(f) \rceil$ とするとき，$\bigcup_{f \in \mathcal{F}, ar(f) \geq 2} \{x_{ij}^f \mid 1 \leq i \leq ar(f), 1 \leq j \leq m\}$ となる．

5.3 隣接式符号化

隣接式符号化では，それぞれの関数記号 f について，$x_{ij}^f \Leftrightarrow$ 引数 i の次に引数 j が比較される，に対応する論理変数を導入する．このとき，比較の最初と最後を表わすため，x_{0i}^f,x_{in}^f $(n = ar(f)+1, 1 \leq i \leq ar(f))$ なる論理変数も入れる．つまり，論理変数集合としては $\bigcup_{f \in \mathcal{F}, ar(f) \geq 2} \{x_{ij}^f \mid 0 \leq i \leq ar(f), 1 \leq j \leq ar(f)+1\}$ を用意する．第 i 引数の後に比較される引数も，第 j 引数の前に比較される引数も唯 1 つに決まる必要がある．従って，これらの一意性条件を符号化した以下の命題を $COND_e$ に追加する．

$$\bigwedge_{\substack{f \in \mathcal{F} \\ ar(f) \geq 2}} \Big(\bigwedge_{0 \leq i \leq |\mathcal{F}|} (\mathrm{ones}(\{x_{ij}^f \mid i \neq j, 1 \leq j \leq ar(f)+1\})) \wedge \bigwedge_{1 \leq j \leq |\mathcal{F}|+1} (\mathrm{ones}(\{x_{ij}^f \mid i \neq j, 0 \leq i \leq ar(f)\})) \Big)$$

5.4 実装および比較実験

表 1 に，優先順位の符号化に記号符号化を，置換の符号化に 3 つの符号化を用いた場合の置換付き辞書式経路順序による停止性検証実験の結果を示す．実験結果を見ると，記号符号化を使用した場合でも，SK

式符号化を使用した場合と遜色ないパフォーマンスを示している．このことから，特に置換の場合にだけ，記号符号化を避ける理由はないように見受けられる．

また，アリティの大きな関数記号を使用した場合には，関数記号の優先順位の符号化と同様に，記号符号化の方が利点があるのではないかと考えられる．通常の関数プログラムをモデル化した項書き換えシステムでは，あまり大きなアリティをもつ関数記号を使用することは考えにくいが，プロトコルのモデル化 [3] などでは，大きなアリティをもつ関数記号を用いる場合も考えられる．そのような場合には，置換についても記号符号化を用いることにより，より効率的な停止性検証を行うことが出来るのではないかと予測される．

また，比較のために考案した隣接式符号化と比べても，SK 式符号化，記号符号化の順に若干づつパフォーマンスがよくなっているものの，あまり優劣に差が見られないことから，実験対象となっている項書き換えシステムには，大きなアリティをもつ関数記号を用いた例が少ないために，実験結果には置換の符号化法の優劣があまり明らかには表面化していないと考えられる．

6 おわりに

本論文では，置換付き辞書式経路順序における置換の符号化法として，関数記号の優先順位の符号化法で標準的となっている記号符号化に基づく方法を提案するとともに，比較実験を行い，文献 [8, 9] で用いられている従来手法と比較して遜色ないパフォーマンスを得られたことを報告した．また，アリティの大きな関数記号を使用する場合には，関数記号の優先順位の符号化と同様に，記号符号化の方が利点があるのではないかと予想される．このような観点からの実験例を用いて，提案手法の有効性について，より精密な実証実験を進めることは今後の課題である．

参考文献

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] M. Codish, V. Lagoon, and P. J. Stuckey. Solving partial order constraints for LPO termination. In *Proc. of RTA 2006*, Vol. 4098 of *LNCS*, pp. 4–18. Springer-Verlag, 2006.

[3] T. Genet and F. Klay. Rewriting for cryptographic protocol verification. In *Proc. of CADE-17*, Vol. 1831 of *LNAI*, pp. 271–290. Springer-Verlag, 2000.

[4] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. of IJCAR 2006*, Vol. 4130 of *LNAI*, pp. 281–286. Springer-Verlag, 2006.

[5] E. Giunchiglia and R. Sebastiani. Applying the Davis-Putnam procedure to non-clausal formulas. In *Proc. of the 6th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, Vol. 1792 of *LNCS*, pp. 84–94. Springer-Verlag, 1999.

[6] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.

[7] M. Kurihara and H. Kondo. Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In *Proc. of 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Vol. 3029 of *LNCS*, pp. 827–837. Springer-Verlag, 2004.

[8] 佐藤晴彦, 栗原正仁. ステータス付き再帰的経路順序による項書換え系多重完備化手続きの実装と性能評価. *IEICE Trans. on Inf. & Sys.*, J89-D(4):624–631, 2006.

[9] P. Schneider-Kamp, R. Thiemann, E. Annov, M. Codish, and J. Giesl. Proving termination using recursive path orders and SAT solving. In *Proc. of FroCoS '07*, Vol. 4720 of *LNAI*, pp. 267–282. Springer-Verlag, 2007.

[10] U. K. Steinbach. Check your ordering–termination proofs and open problems. Technical report, Universität Kaiserslautern, 1990.