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

Implementation and performance evaluation of multi-completion procedures for term rewriting systems with recursive path orderings with status

佐藤 晴彦 † 栗原 正仁 †

Haruhiko SATO Masahito KURIHARA

† 北海道大学情報科学研究科

Graduate School of Information Science and Technology, Hokkaido University

{haru,kurihara}@main.ist.hokudai.ac.jp

本論文では，多重完備化手続き MKB_{po} において取り扱う簡約順序を辞書式経路順序 (LPO) とした MKB_{lpo} からステータスを用いた再帰的経路順序 (RPOS) に拡張することで，完備化が可能となる範囲を大きく広げた完備化手続き MKB_{rpos} の実装とその性能評価について報告する．MKB_{lpo} と同様，RPOS の集合を論理関数で表現し，BDD で処理することにより高速化を図っている．実験により，考慮すべき順序の数が多い場合は，考え得る全ての RPOS を MKB で並列に取り扱った場合よりも高速であることを示す．また，MKB_{lpo} で解ける問題についても，考慮すべきステータスの組合せが数倍から数百倍多いにもかかわらず，平均的にわずかな計算時間のオーバヘッドで解くことができることを実証する．

1 まえがき

項書換え系 [1, 2] は書換え型の計算モデルの一つであり，項の対 $l \to r$ の形で与えられた書換え規則の集合によって項を順次書き換えることで計算を表現するものである．項書換え系は関数型プログラミング言語、等式に基づく言語、定理自動証明、記号処理など、計算機科学の様々な分野で広く用いられている。項書換え系に関する重要な性質として，停止性と合流性がある．停止性は項の書換えが無限に続かないという性質であり，合流性は書換えの道筋がどのようなものであっても，最終的に得られる正規形が（もしあれば）一意となるための十分条件である．これらの性質を共に満たす時，項書換え系は完備であるという．停止性は簡約順序と呼ばれる項の集合上の半順序を適切に定めることで保証され，停止性を満たす項書換え系は，さらに弱合流性と呼ばれる性質が保証されることによって完備となる．

与えられた等式仕様に対して，適切な簡約順序を用いて停止性を保証しながら，合流性を満たすよう書換え規則の追加や変更を行うことで完備な書換え系を構成する手続きを完備化手続き (KB)[3] といい，最も基本的な手続きは Knuth と Bendix により考案された．KB は等式仕様と 1 つの簡約順序を引数とする手続きであり，与える簡約順序によって結果が成功，失敗，発散（手続きが終了しない）の 3 つのいずれかとなる．よって，完備化が成功するような適切な簡約順序を利用者が見つけ出すことが必要となるが，手続きが終了しないことがあるため，考え得る簡約順序を順に一つずつ試していくということは不可能である．この問題に対する解決法として，複数の完備化手続きを並列に実行する事が考えられる．しかし，考え得る簡約順序は一般的に膨大になるため，単純に並列に実行する事は事実上困難である．

この問題に対し，栗原ら [4] は複数の KB を並列に実行するプロセス群の動作を単一のプロセスで効率良くシミュレートする MKB と呼ばれる完備化手続きを開発した．本論文では，MKB のような複数の簡約順序を同時に取り扱う完備化手続きを，以前の完備化手続きと区別するため多重完備化手続きと呼ぶ．更に，横山ら [5] は MKB において取り扱う簡約順序を経路順序 (PO) に限定し，PO を論理関数として表現し，更に論理関数を BDD（二分決定グラフ）[6][7][8] を用いて処理することで MKB の効率を向上させた多重完備化手続き MKB_{po} を開発した．PO とは，優先順位と呼ばれる関数記号の集合上の半順序を項の集合上に構文的に拡張して得られる順序のクラスであり，関数記号が有限個ならば，考え得る全ての経路順序は有限個である．よって，簡約順序を PO

に限定することで，利用者が簡約順序をあらかじめ用意することなく多重完備化を行うことが可能となった．しかし，横山らが示した実装および実験結果は PO のうちで最も単純な辞書式経路順序 (LPO) のみであるため，それがより広いクラスの PO の時にも現実的な時間で解が求まるかは自明ではない．以下，本論文では LPO を用いた実装を MKB_{lpo} と記し，基本的なアイディアとしての MKB_{po} と区別する．

本論文では， MKB_{po} において取り扱う簡約順序をステータス付き再帰的経路順序 (RPOS)[9] と呼ばれる標準的によく用いられる実用的で広いクラスにすることで，完備化が可能となる範囲を大きく広げた多重完備化手続き MKB_{rpos} の実装と性能評価結果について報告する．MKB_{po} と同様， MKB_{rpos} においても RPOS を論理関数で表現し，BDD で処理するため，ステータスの論理関数表現の抽象化を行い，さらに具体的なステータスの論理表現を 3 つ示す．

RPOS におけるステータスとは，関数記号の等しい 2 つの項の比較に含まれる部分項の列同士の比較方法を定めるものである．アリティが n の各関数記号に対して，逐次的な比較の順序を指定する $(1\ 2 \ldots n)$ の $n!$通りの置換あるいは多重集合順序による一括的な比較を指定する mul の合計 $n!{+}1$ 通りのうちの 1 つをステータスとして与えることが可能である．それに対して，LPO はステータスを $(1\ 2 \ldots n)$ の 1 つに固定している．したがって，関数記号 $f_1, \ldots, f_\nu$ のアリティがそれぞれ $n_1, \ldots, n_\nu$ であるとき，ステータスの組合せの数は LPO が 1 であるのに対し，RPOS では

$$\prod_{i=1}^{\nu}[u(n_i - 2)n_i! + 1]$$

となり，多くの場合，これは膨大なものとなる．ただし，$u(x)$ は単位ステップ関数

$$u(x) = \text{ if } x \geq 0 \text{ then } 1 \text{ else } 0$$

である。MKB_{rpos} で考慮すべき経路順序の総数はこの数に優先順位の総数（全順序に限定しても $\nu!$ある）を乗じたものとなる．

このように，MKB_{rpos} では考慮すべき簡約順序の数は一般に膨大となるが，完備化の標準的な問題集である [10] に掲載されている殆ど全ての問題を高々3 分以内，多くの場合 1 秒以内で解くことができ，問題の規模が増大するにつれ，考え得る全ての RPOS を MKB で並列に取り扱うよりも効率よく完備化が行えることを，実験結果から示す．また，MKB_{lpo} で解ける問題についても，考慮すべきステータスの組合せが数倍から数百倍多いにもかかわらず，平均的にわずかな計算時間のオーバヘッドで解くことができることを実証する．

2 再帰的経路順序を考慮した多重完備化手続き

MKB_{lpo} は，MKB_{po} における経路順序を LPO に限定したが，これを RPOS に拡張することで，完備化が成功となる範囲が広がり，かつ直接全ての RPOS を入力して MKB を実行するよりも高速な多重完備化が実現できるものと考えられる．本節では， MKB_{po} の基本的な考え方をたどりながら，そのような実装 MKB_{rpos} の論理的な枠組を述べる．なお，以下では，次のような表記法を用いる．

関数記号の集合を Σ，関数記号 f のアリティを $ar(f)$，アリティが 2 以上の関数記号の集合を Σ' で表す．変数の集合を $\mathcal{V}$，項 t に現れる変数の集合を $\mathcal{V}ar(t)$ で表す．Σ 上の厳格半順序を優先順位と呼び，$\succ$ で表す．

2.1 ステータス付き再帰的経路順序

一般に，経路順序では関数記号の等しい 2 つの項 $f(t_1, \ldots, t_n), f(u_1, \ldots, u_n)$ の順序を定めるため，それぞれの項の部分項同士の順序を用いて，部分項の組 $\langle t_1, \ldots, t_n\rangle, \langle u_1, \ldots, u_n\rangle$ の順序を再帰的に定義する．LPO では，部分項は関数記号とは無関係に左から右へと比較されるが，RPOS では，その比較方法を関数記号ごとに定める．項の集合上の順序を項の組の集合上の順序に拡張する方法として，RPOS では辞書式拡張と多重集合拡張の 2 つを用いる．

定義 1 （辞書式拡張）$\succ$ を集合 S 上の厳格半順序とする．$t_1, \ldots, t_n, u_1, \ldots, u_n \in S$ とすると，$\succ$ の辞書式拡張によって得られる S^n 上の厳格半順序 $\succ_{lex}$ は，次のように定義される．

$$\langle t_1, \ldots, t_n\rangle \succ_{lex} \langle u_1, \ldots, u_n\rangle \iff$$
$$\exists k \leq n.(\forall i < k.t_i = u_i) \wedge t_k > u_k$$

定義 2 （多重集合拡張）$\succ$ を集合 S 上の厳格半順序とする．また，$\mathcal{M}(S)$ を S の要素から構成される有限な多重集合の集合とする．$M, N \in \mathcal{M}(S)$ とすると，$\succ$ の多重集合拡張によって得られる $\mathcal{M}(S)$ 上の

厳格半順序 $\succ_{mul}$ は，次のように定義される．

$$M \succ_{mul} N \iff M \neq N \ and$$
$$\forall n \in N - M. \exists m \in M - N. m \succ n$$

$\{1, \ldots, n\}$ の要素からなる全ての置換の集合を $Per(n) = \{\pi_1^n, \ldots, \pi_{n!}^n\}$ で表す．以降，混乱が生じないときには π_i^n を単に π_i と書く．置換 $\pi \in Per(n)$ を用いて辞書式の比較を行うことを表すステータスの集合を $Lex(n) = \{lex_\pi | \pi \in Per(n)\}$ とする．多重集合を用いて比較を行うことを表すステータスを mul で表す．ただし，アリティが 0 または 1 のときはステータスの区別がないので，形式的に mul を用いることにする．これらを用いると，アリティが n である関数記号における全てのステータスの集合 $Status(n)$ は

$$Status(n) = \begin{cases} \{mul\} \text{ if } n \leq 1 \\ Lex(n) \cup \{mul\} \text{ if } n \geq 2 \end{cases}$$

となる．

ステータスと関数記号との対応関係を表すため，ステータス関数と呼ばれる関数記号からステータスへの写像を導入する．

定義 3（ステータス関数）ステータス関数 τ は関数記号 $f \in \Sigma$ をステータス $\tau(f) \in Status(n)$ に対応づける写像である．ただし，$n = ar(f)$ とする．

項の集合上の任意の半順序 $\succ$ に対し，ステータスに基づく項の S^n 上の半順序 $\succ^{\tau(f)}$ を以下のように定義する．

$$\langle t_1, \ldots, t_n \rangle \succ^{mul} \langle u_1, \ldots, u_n \rangle \iff$$
$$\{t_1, \ldots, t_n\} \succ_{mul} \{u_1, \ldots, u_n\}$$
$$\langle t_1, \ldots, t_n \rangle \succ^{lex_\pi} \langle u_1, \ldots, u_n \rangle \iff$$
$$\langle t_{\pi(1)}, \ldots, t_{\pi(n)} \rangle \succ_{lex} \langle u_{\pi(1)}, \ldots, u_{\pi(n)} \rangle$$

以上の定義より，ステータス付き再帰的経路順序 (RPOS) は以下のように定義できる．

定義 4（*RPOS*）$\succ$ を優先順位，τ をステータス関数とする．t, u を項とし，t および u が変数でない場合は $t = f(t_1, \ldots, t_m), u = g(u_1, \ldots, u_n)$ であるものとすると，$t \succ_{rpos} u$ を，以下の条件のうちいずれかが成り立つことと定義する．

- $u \in \mathcal{V}ar(t)$
- $t_i \succeq_{rpos} u \ for \ 1 \leq \exists i \leq m$
- $f \succ g \ and \ t \succ_{rpos} u_j \ for \ 1 \leq \forall j \leq n$
- $f = g \ and \ t \succ_{rpos} u_j \ for \ 1 \leq \forall j \leq n,$
  $and \ \langle t_1, \ldots, t_n \rangle \succ_{rpos}^{\tau(f)} \langle u_1, \ldots, u_m \rangle$

ただし，$\succeq_{rpos}$ は $\succ_{rpos}$ または $=$ であることを表し，$\succ_{rpos}^{\tau(f)}$ は $(\succ_{rpos})^{\tau(f)}$ を表す．

特に，ステータス $lex_\pi \in Status(n)$ のうち，恒等置換 $\pi = (1\ 2 \ldots n)$ に対する lex_π を $left$，また $\pi = (n\ n-1 \ldots 1)$ に対する lex_π を $right$ と名付ける．$left$ は引数を左から右へ，$right$ は右から左へ比較するステータスを表現している．すべての f が $left$ ステータスを持つとき，RPOS は辞書式経路順序 (LPO) に一致する．また，すべての f が mul ステータスを持つとき，RPOS は多重集合経路順序 (MPO) に一致する．(MPO は旧来から再帰的経路順序 (RPO) と呼ばれてきたものである．最近は RPOS のことを RPO と呼ぶことがある．)

2.2 論理関数による RPOS の集合の表現

本節では，優先順位やステータスといった，RPOS を特徴付ける属性に対応する論理変数を導入し，論理変数への真偽の割り当てと RPOS の対応関係を示す．これにより，RPOS の集合を論理関数で表現することが可能となる．

優先順位 $\succ$ を表現する論理変数の集合 X_{prec}，及び変数の真偽に対応する意味を示す．

$$X_{prec} = \{x_{fg} | f, g \in \Sigma, f \neq g\}$$

$$x_{fg} = \begin{cases} 1, \text{ if } f \succ g \\ 0, \text{ if } f \not\succ g \end{cases}$$

関数記号の優先順位が推移性，半対称性を満たすという制約に対応する論理関数 $T(X_{prec}), A(X_{prec})$ は以下のように表せる．

$$T(X_{prec}) = \prod_{f \neq g \neq h \neq f} (\overline{x}_{fg} + \overline{x}_{gh} + x_{fh})$$

$$A(X_{prec}) = \prod_{f \neq g} (\overline{x}_{fg} + \overline{x}_{gf})$$

次に，ステータス関数の論理的表現を示す．ステータス関数 τ の論理的表現とは，アリティが 2 以上のすべての関数記号 $f \in \Sigma'$ に対して次の 2 つを定めたものである．

- 論理変数の集合 X^f

- 論理関数 $STAT^f_s(X^f), s \in Status(ar(f))$

ただし，$f \neq g$ ならば $X^f \cap X^g = \emptyset$ であるものとする．また，$STAT^f_s$ は次の条件（良定義性と呼ぶ）を満たすものとする．

[良定義性] 任意の f, s に対して

$$Only^f_s(X^f) = STAT^f_s(X^f) \prod_{s' \neq s} \overline{STAT}^f_{s'}(X^f)$$

で定義される論理式は充足可能であること．

τ が与えられたとき，この条件は任意の f に対して，$STAT^f_{\tau(f)} = 1$ でかつ $\tau(f)$ 以外のステータス s' について $STAT^f_{s'} = 0$ となるような代入（各変数への論理値の割当て）が存在することを保証するので，その代入を $\tau(f)$ の論理的表現とすることができる．そのような代入が一意に定まるとは限らないが，値の任意性（don't care）を認めることで $STAT^f_s$ を簡潔に定義できる可能性をもたせる．逆に各 $s \in Status(ar(f))$ に対して求まるそのような代入のみが意味をもつ．そのような代入は次の条件で表すことができる．

[ステータス条件] 任意の f に対して $\Sigma_s Only^f_s = 1$.

この条件を満たす代入が与えられたとき，ただ 1 つの s に対して $STAT^f_s = 1$ となるので，$\tau(f) = s$ として τ を一意に決定することができる．

以下では

$$X_{stat} = \cup_{f \in \Sigma'} X^f$$

$$S(X_{stat}) = \prod_{f \in \Sigma'} \sum_{s \in Status(ar(f))} Only^f_s$$

とおく．

X_{stat} の具体的な定義は様々なものが考えられ，それに応じて $STAT^f_s$ の具体的な定義も変化する．具体的な定義は，後に示す 4 種類の X_{stat} と共に示す．

以上の全ての論理変数を含む集合を

$$X = X_{prec} \cup X_{stat}$$

とする．

代入が $T(X_{prec})A(X_{prec})S(X_{stat}) = 1$ を満たすことを前提とし，さらに，項 t, u に対し，$t \succ_{rpos} u$ となるような RPOS を表現する論理関数 $RPOS_{t,u}(X)$ を，再帰的に以下のように定義する．ここで，t, u が変数でない場合は $t = g(t_1, \ldots, t_m), u = g(u_1, \ldots, u_n)$ とする．

$$RPOS_{t,u}(X) = \begin{cases} 0 \text{ if } t = u \text{ or } t \in \mathcal{V} \text{ or } \mathcal{V} \ni u \notin \mathcal{V}ar(t) \\ 1 \text{ if } t \notin \mathcal{V}, u \in \mathcal{V}ar(t) \text{ or } \exists i.t_i = u \\ \sum_{i=1}^m RPOS_{t_i,u}(X) \\ \quad + x_{fg} \cdot \prod_{j=1}^n RPOS_{t,u_j}(X) \\ \quad \text{if } f \neq g, \forall i.t_i \neq u \\ RPOS_{t_1,u_1}(X) \text{ if } f = g, ar(f) = 1, t \neq u, t_1 \neq u \\ \sum_{i=1}^m RPOS_{t_i,u}(X) \\ \quad + \sum_{lex_\pi \in Lex(n)} STAT^f_{lex_\pi}(X) \cdot LEX_{t,u,\pi}(X) \\ \quad + STAT^f_{mul}(X) \cdot MEX_{t,u}(X) \\ \quad \text{if } f = g, ar(f) \geq 2, t \neq u, \forall i.t_i \neq u \end{cases}$$

ただし，$LEX_{t,u,\pi}(X), MEX_{t,u}(X)$ は辞書式拡張と多重集合拡張を表す論理関数であり，以下のように定義される．

$$LEX_{t,u,\pi}(X) = RPOS_{t_{\pi(k)},u_{\pi(k)}}(X) \cdot \prod_{j=k+1}^{n} RPOS_{t,u_{\pi(j)}}(X)$$

ただし，k は $t_{\pi(k)} \neq u_{\pi(k)}$ となるような最小の整数 $(1 \leq k \leq n)$ である．

$$MEX_{t,u}(X) = \prod_{y \in N-M} \sum_{x \in M-N} RPOS_{x,y}(X)$$

ただし，M, N は多重集合であり，$M = \{t_1, \ldots, t_n\}, N = \{u_1, \ldots, u_n\}$ とする．

文献 [5] と同様の考え方により，明らかに次のことが成り立つ．

定理 1 X へのある代入のもとで

$$T(X_{prec})A(X_{prec})S(X_{stat})RPOS_{t,u}(X) = 1$$

ならば，その代入が表現する優先順位 $\succ$ とステータス関数 τ のもとで

$$t \succ_{rpos} u.$$

2.3 MKB_{rpos} 推論規則

MKB_{po} と同様，ノードと呼ばれる 4 つ組 $\langle t : u, F_1, F_2, F_3 \rangle$ を用いることで複数の簡約順序を同時に取り扱う．$t : u$ は項の順序対であり，F_1, F_2, F_3 は論理関数である．各 F は，$F = 1$ を満たす代入によって表現される RPOS の集合を表現している．F_1 は $t : u$ を $t \to u$ の向きに向き付け，F_2 は $t : u$ を $u \to t$ の向きに向き付け，F_3 は $t : u$ が向き付けされていない RPOS の集合である．また，このノードはノード $\langle u : t, F_2, F_1, F_3 \rangle$ と同一視される．MKB_{rpos}

はノードの集合 N に対する推論規則として表される．具体的な推論規則や完備化手続きの実装例および完備化の終了判定については文献 [5] を参照されたい．

3 論理関数によるステータス関数の表現

本研究においては，文献 [5] と同様に，論理関数を BDD で実装する．BDD の積及び和の計算時間は，BDD のサイズの積に比例するため，完備化手続きの性能を向上させるためには，平均的な BDD のサイズを小さくすることが重要である．そのサイズは論理関数の具体的な定義に依存する．本節では，前節で抽象的に定義されていた X^f と $STAT^f_s$ の具体的な定義を 4 種類示す．

なお，いずれにおいても $n = ar(f) \geq 2$ とする．

3.1 インデックスによる置換表現 (P1)

アリティが n の関数記号が取り得る $n!$ 通りの置換と，論理変数を 1 対 1 に対応させるため，$n!$ 個の論理変数を導入し，対応する置換を用いる場合に論理変数が 1 となるようにする．2.2 で抽象的に表現した X^f と $STAT^f_s$ を以下のように具体的に表現する．

$$X^f = \{x^f_{mul}\} \cup \{x^f_i | 1 \leq i \leq n!\}$$

$$STAT^f_s(X^f) = \begin{cases} x^f_{mul} \text{ if } s = mul \\ x^f_i \text{ if } s = lex_{\pi_i} (i = 1, 2, \ldots, n!) \end{cases}$$

例えば，$\pi = \pi^3_5 = (3\ 1\ 2)$ とすると，$STAT^f_{lex_\pi}$ は x^f_5 となる．

良定義性は自明である．s に対応する 1 つの変数を 1，他の全ての変数を 0 とすれば，$Only^f_s = 1$ となる．

3.2 対応関係による置換表現 (P2)

$(1\ 2 \ldots n)$ の置換を表現するために，$1 \leq i, j \leq n$ を満たす n^2 通りの i, j の組について，$\pi(j) = i$ が成り立つときに限り $x_{ij} = 1$ とするように X^f と $STAT^f_s$ を定める．

$$X^f = \{x^f_{mul}\} \cup \{x^f_{ij} | 1 \leq i, j \leq n\}$$

$$STAT^f_s(X^f) = \begin{cases} x^f_{mul} \text{ if } s = mul \\ \prod_{j=1}^n x^f_{\pi(j),j} \text{ if } s = lex_\pi \end{cases}$$

例えば，$\pi = (3\ 1\ 2)$ とすると，$STAT^f_{lex_\pi}$ は $x^f_{31} x^f_{12} x^f_{23}$ となる．

s に対応する単項式を構成するすべての変数を 1，他のすべての変数を 0 とすれば，$Only^f_s = 1$ となり，良定義性が確かめられる．

3.3 優先関係による置換表現 (P3)

置換 π を用いて辞書式の比較を行う場合，i 番目の部分項は $\pi^{-1}(i)$ 番目に比較されることになる．つまり，$\pi^{-1}(i)$ の値が小さいほど，i 番目の部分項が優先的に用いられることを意味している．この優先関係を用いて，$1 \leq i < j \leq n$ を満たす $n(n-1)/2$ 通りの i, j について，i 番目の部分項は j 番目の部分項より先に用いられるかどうかという情報を論理変数に対応させる．これは後の良定義性の証明からわかるように，各 k について $\pi(k)$ と $\pi(k+1)$ の大小だけを論理変数でコーディングすることで表現できる．

$$X^f = \{x^f_{mul}\} \cup \{x^f_{ij} | 1 \leq i < j \leq n\}$$

$$STAT^f_s(X^f) = \begin{cases} x^f_{mul} \text{ if } s = mul \\ \bar{x}^f_{mul} P^f_\pi \text{ if } s = lex_\pi \end{cases}$$

ただし，

$$P^f_\pi = \prod_{k=1}^{n-1} L^f_{\pi(k),\pi(k+1)}$$

$$L^f_{ij} = \begin{cases} x^f_{ij} \text{ if } i < j \\ \bar{x}^f_{ji} \text{ if } j < i \end{cases}$$

である．例えば，$\pi = (3\ 1\ 2)$ とすると，$STAT^f_{lex_\pi}$ は $\bar{x}^f_{mul} \bar{x}^f_{13} x^f_{12}$ となる．

良定義性の証明を以下に示す．$s = mul$ のときは $x_{mul} = 1$，他のすべての変数の値を任意に定めると $Only^f_s = 1$．$s = lex_\pi$ のときは $x_{mul} = 0$ とするが，その他の変数の値を定めるために，$\{1, 2, \ldots, n\}$ 上の全順序 $<_\pi$ を次式で定義する．

$$i <_\pi j \iff \pi^{-1}(i) < \pi^{-1}(j)$$

すなわち，$i <_\pi j$ は $\pi = (\ldots i \ldots j \ldots)$ の形であることを表している．この全順序を次式で定義される代入で論理的に表現する（$1 \leq i < j \leq n$）．

$$x^f_{ij} = \begin{cases} 1 \text{ if } i <_\pi j \\ 0 \text{ if } j <_\pi i \end{cases}$$

この代入のもとで $P^f_\pi = 1$，その他の置換 $\pi' (\neq \pi)$ に対して $P^f_{\pi'} = 0$ となることを以下のように確かめることができる．

- $P^f_\pi = 1$：任意の $k(1 \le k < n)$ について，$i = \pi(k)$，$j = \pi(k+1)$ とおくと，$i <_\pi j$．したがって，$i < j$ ならば $x^f_{ij} = 1$，$j < i$ ならば $x^f_{ji} = 0$ となるので，いずれにせよ $L^f_{ij} = 1$．ゆえに $P^f_\pi = 1$.
- $P^f_{\pi'} = 0$：任意の k について，$i = \pi'(k)$，$j = \pi'(k+1)$ とおくと，$i <_{\pi'} j$．もし $\pi' \neq \pi$ ならば，ある k が存在して $i \not<_\pi j$ となっているはずである．このとき，$j <_\pi i$ なので，$j < i$ ならば $x^f_{ji} = 1$，$i < j$ ならば $x^f_{ij} = 0$ となり，いずれにせよ $L^f_{ij} = 0$．ゆえに $P^f_{\pi'} = 0$．（証明終）

3.4 変数の数を最小にする置換表現 (P4)

異なる n 通りの状態を区別するためには，少なくとも $b = \lceil log_2 n \rceil$ ビットの情報が必要である．よって，関数記号 f が取り得る $ar(f)! + 1$ 個のステータスそれぞれに 0 から $ar(f)!$ のインデックスを割り当て，その 2 進表現における各桁の値と論理変数を対応させることで，論理変数の数を最も少なくすることができる．以下，$b_f = \lceil log_2(ar(f)! + 1) \rceil$ とする．

$$X^f = \{x^f_i | 1 \le i \le b_f\}$$

$$STAT^f_s(X^f) = \begin{cases} \prod_{j=1}^{b_f} P^{i-1}_j \text{ if } s = lex_{\pi_i} \\ \prod_{j=1}^{b_f} P^{ar(f)!}_j \text{ if } s = mul \end{cases}$$

ただし，

$$P^i_j = \begin{cases} x^f_j \text{ if } c^i_j = 1 \\ \bar{x}^f_j \text{ if } c^i_j = 0 \end{cases}$$

とする．ここで、c^i_j は i の 2 進表現 $c_{b_f} \ldots c_1$ における c_j を表す．例えば，$\pi = \pi^3_5 = (3\ 1\ 2)$ とすると，$STAT^f_{lex_\pi}$ は $x^f_2 \bar{x}^f_1 \bar{x}^f_0$ となる．

$STAT^f_s$ は，s に対応するインデックスの 2 進表現における各桁の値を対応する論理変数の値としたとき，またそのときに限り 1 となることから，良定義性を満たすことがわかる．

4 実験結果

本章では，MKB$_{rpos}$ を実装し，いくつかの条件のもとで多重完備化を行った結果について述べる．なお，BDD のサイズ（ノード数）はその変数順序（根から終端ノードに至る経路上で出現する変数の順序）に大きく依存するが，この実装では使用した BDD パッケージ（BuDDy）の動的な順序付けを行う機能に任せている．以下の実験では，文献 [10] に示された例題のうち，RPOS で完備化が可能であり，アリティが 2 以上の関数記号を含む 22 の項書換え系すべてを用いている．また，実験結果の表中における No は，完備化に用いた例題の番号 (EXAMPLE 3.XX の XX) を表す．

表 1: MKB と MKB$_{rpos}$ の計算時間の比較

No.	順序数	MKB (sec)	rpos (sec)	$\frac{MKB}{rpos}$
2	6	0.42	0.42	1.01
32	6	0.40	0.40	1.00
33	6	0.40	0.40	1.01
17	18	0.42	0.41	1.00
5	18	6.00	4.98	1.20
11	72	0.42	0.41	1.03
4	72	50.59	32.59	1.55
8	162	0.79	0.52	1.52
1	216	4.43	1.48	2.99
18	216	0.56	0.44	1.27
3	216	126.16	41.07	3.07
16	360	0.45	0.41	1.11
31	360	0.43	0.41	1.06
10	648	0.60	0.42	1.44
13	1080	1.29	0.47	2.73
14	1080	1.02	0.44	2.30
30	1080	0.59	0.41	1.43
19	3240	3.26	0.47	6.91
21	3240	3.76	0.47	7.97
6	3240	606.59	14.39	42.15
7	7560	>1800	147.33	-
20	88179840	>1800	8.69	-

4.1 MKB と MKB$_{rpos}$ の比較

考え得る全ての RPOS を考慮した MKB と MKB$_{rpos}$ それぞれを用いて完備化を行い，計算時間を調べた．結果を表 1 に示す．考慮すべき簡約順序の数が少ないいくつかの例では，わずかに MKB の方が高速となっているが，ほとんどの例では MKB$_{rpos}$ の方が遥かに高速であり，考慮すべき簡約順序の数が増加するにつれ，速度差も大きくなっていることが分かる．

表 2: MKB_{lpo} と MKB_{rpos} の計算時間の比較

No	lpo の順序数	lpo (sec)	rpos の順序数	rpos (sec)	$\frac{rpos}{lpo}$
2	2	0.42	6	0.42	1.00
32	2	0.40	6	0.40	1.00
33	6	0.40	6	0.40	1.00
17	2	0.42	18	0.42	1.00
5	6	5.06	18	5.04	1.00
11	24	0.41	72	0.41	1.00
4	24	32.14	72	32.09	1.00
8	6	0.52	162	0.52	1.00
1	24	1.46	216	1.45	0.99
18	24	0.44	216	0.45	1.02
3	24	16.52	216	41.50	2.51
16	120	0.41	360	0.41	1.00
31	120	0.40	360	0.41	1.02
10	24	0.42	648	0.42	1.00
13	120	0.47	1080	0.47	1.00
14	120	0.44	1080	0.45	1.02
30	120	0.41	1080	0.41	1.00
19	120	0.47	3240	0.48	1.02
21	120	0.47	3240	0.47	1.00
7	120	148.11	7560	148.02	1.00
20	362880	8.30	88179840	8.99	1.08

表 3: 置換表現と計算時間の関係

No.	P1 (sec)	P2 (sec)	P3 (sec)	P4 (sec)
2	0.42	0.42	0.42	0.42
32	0.40	0.40	0.40	0.40
33	0.40	0.40	0.40	0.40
17	0.42	0.42	0.42	0.41
5	5.03	5.23	4.98	5.25
11	0.41	0.42	0.41	0.41
4	32.05	32.17	31.77	32.26
8	0.53	0.52	0.52	0.52
1	1.46	1.45	1.47	1.48
18	0.44	0.45	0.44	0.44
3	41.63	41.19	42.52	41.65
16	0.41	0.41	0.41	0.41
31	0.41	0.41	0.41	0.41
10	0.42	0.42	0.42	0.42
13	0.47	0.47	0.47	0.47
14	0.45	0.44	0.44	0.44
30	0.41	0.41	0.41	0.41
19	0.47	0.47	0.47	0.47
21	0.47	0.47	0.47	0.48
6	14.06	14.14	14.17	14.34
7	149.15	150.37	147.79	152.39
20	9.92	8.95	8.49	8.72

4.2 MKB_{lpo} と MKB_{rpos} の比較

MKB_{lpo} でも解ける問題について，MKB_{lpo} と MKB_{rpos} の両者を用いて完備化を行い，計算時間を調べた結果を表 2 に示す．MKB_{rpos} では考慮すべき簡約順序の数が数倍から数百倍となっているが，最も時間のかかる例でも MKB_{lpo} の 2.5 倍，その他の例では MKB_{lpo} とほぼ変わらない時間で完備化が完了しており，考慮すべき簡約順序数の増加に伴うオーバヘッドは非常に小さいと言える．

4.3 置換表現と性能の関係

MKB_{rpos} において，第 3 節で導入した 4 種類の論理関数 P1, P2, P3, P4 による置換表現それぞれを用いて多重完備化を行い，計算時間及び手続き中で生成された BDD ノード数を調べた．計算時間を表 3, ノード数の平均値を表 4 に示す．計算時間はどの置換表現を用いた場合も殆ど変わらないが，ノード数は P3 を用いた場合がほぼ全ての例において最も小さくなっている．これは P3 および P4 は論理変数の数が少なく BDD の深さも小さくなり，さらに P3 は P4 よりも RPOS の特徴をよく表現しており，論理関数の値に影響を及ぼさない変数がより多く生じるためと考えられる．

5 むすび

簡約順序をステータス付き再帰的経路順序 (RPOS) に特化した多重完備化手続き MKB_{rpos} の実装とその性能評価を行った．ステータスの論理表現を良定義性を満たす関数 $STAT_s^f$ により抽象化し，$t \succ_{rpos} u$ を満たす RPOS の集合を表現する論理関数 $RPOS_{t,u}(X)$ を統一的に定義した．また $STAT_s^f$ の具体的な例として $P1, P2, P3, P4$ の 4 種類を導入して実装した．

表 4: 置換表現と BDD ノード数の関係

No.	P1	P2	P3	P4
2	4.00	6.00	3.00	3.00
32	5.00	8.00	0.00	2.00
33	5.00	5.00	5.00	5.00
17	16.00	25.00	1.00	5.00
5	7.00	10.00	2.00	4.00
11	25.00	28.00	20.00	22.00
4	15.00	17.00	14.00	14.00
8	42.21	63.21	7.21	13.21
1	20.33	27.33	12.33	14.33
18	22.00	31.00	7.00	11.00
3	113.61	127.61	96.11	98.86
16	88.40	91.40	83.40	85.40
31	66.00	69.00	61.00	63.00
10	65.50	86.50	30.50	36.50
13	70.00	74.00	68.00	68.00
14	35.00	44.00	20.00	24.00
30	73.33	82.33	58.33	62.33
19	34.00	43.00	24.00	26.00
21	81.00	102.00	46.00	52.00
6	153.48	178.48	131.48	135.48
7	147.83	244.83	116.83	114.83
20	5020.00	5081.00	4929.00	4937.00

標準的な問題集 [10] の例題を用いた実験により，簡約順序の数が大幅に増加しても，MKB_{rpos} は現実的な時間で解が求められることを示した．また，ステータスの論理関数表現として P3 を用いた場合に最も BDD のサイズが小さくなることを実験から示した．

参考文献

[1] Terese, “Term rewriting systems”, Cambridge University Press, 2003.

[2] F. Baader, T. Nipkow “Term Rewriting and All That”, Cambridge University Press, 1998.

[3] 外山芳人, “完備化による等式証明”, 人工知能学会誌, Vol.16, No.5, pp.668-674, 2001.

[4] M. Kurihara and H. Kondo, “Completion for multiple reduction orderings”, Journal of Automated Reasoning, Vol.23, No.1, pp.25-42, 1999.

[5] 横山壱星 栗原正仁, “優先順位と二分決定グラフに基づく複数経路順序下の項書換え系完備化手続き”, 人工知能学会論文誌, Vol.19, No.6, pp.472-482, 2004.

[6] R. Bryant, “Graph-based algorithms for boolean function manipulation”, IEEE Transactions on Computers, Vol.C-35, No.8, pp.677-691, 1986.

[7] 石浦菜岐佐, “BDD とは”, 情報処理, Vol.34, No.5, pp.585-592, 1993.

[8] 湊真一, “計算機上での BDD の処理技法”, 情報処理, Vol.34, No.5, pp.593-599, 1993.

[9] J. Steinbach, “Extensions and comparison of simplification orderings”, Proceedings of the 3rd Conference on Rewriting Techniques an Applications, N. Dershowitz, Ed., Vol.355 of Lecture Notes in Computer Science, Springer, pp.434-448, 1989.

[10] J. Steinbach and U. Kühler, “Check your ordering - termination proofs and open problems”, SEKI report SR-90-25(SFB), Fachbereich Informatik, Universität Kaiserslautern, Germany, 1990.