HOKKAIDO UNIVERSITY

Title	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
Author(s)	横山, 壱星; Yokoyama, Issei; 栗原, 正仁 他
Citation	人工知能学会論文誌, 19(6), 472-482 https://doi.org/10.1527/tjsai.19.472
Issue Date	2004
Doc URL	https://hdl.handle.net/2115/14561
Type	journal article
File Information	yokoyama2004jsai-final.pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

論 文

Technical Papers

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

Completion for Multiple Path Orderings based on Precedence and Binary Decision Diagrams

横山 壱星
Issei Yokoyama
北海道大学大学院 工学研究科
Graduate School of Engineering, Hokkaido University
issei@main.eng.hokudai.ac.jp

栗原 正仁
Masahito Kurihara
北海道大学大学院 情報科学研究科
Graduate School of Information Science and Technology, Hokkaido University
kurihara@main.ist.hokudai.ac.jp

keywords: term rewriting system, completion, precedence, path ordering, binary decision diagram

Summary

In this paper, we propose a completion procedure (called MKB_{po}) for term rewriting systems. Based on the existing procedure MKB which works with multiple reduction orderings and the ATMS nodes, the MKB_{po} improves its performance by restricting the class of reduction orderings to precedence-based path orderings, representing them by logical functions in which a logical variable x_{fg} represents the precedence $f \succ g$. By using BDD (binary decision diagrams) as a representation of logical functions, the procedure can be implemented efficiently. This makes it possible to save the number of quasi-parallel processes effectively and suppress the rapid increase in the amount of computation time asymptotically.

1. はじめに

関数型プログラミング，代数的仕様記述，定理自動証明，記号処理など，等式によって表現されたシステムが数多く研究されている．項書換え系 [二木 83, Dershowitz 90, 外山 01, Terese 03] は与えられた等式仕様から導かれる項の書換え規則の集合として定義され，項を書換え規則に従って順次書換えることで計算を行う．項書換え系に要求される重要な性質として，停止性（書換えが停止すること）および合流性（書換えの結果が一意であること）があり，これらを共に満たすとき，書換え系は完備であるいう．停止性は簡約順序とよばれる項の集合上の順序を適切に定めることによって保証され，停止性を満たす項書換え系は，さらに弱合流性（あるいは局所合流性）とよばれる性質が保証されることによって完備となることが知られている．

与えられた等式仕様に対して，書換えの停止性を満たすように簡約順序を適切に決定し，さらにその上で合流性を満足させることで完備な書換え系を得る操作を完備化手続き（KB）という．完備化手続きは１つの簡約順序に対して行われ，結果は完備化成功，失敗，発散して手続きが終了しない，の３つのいずれかをとり，これは簡約順序の取り方に大きく左右される．このため，簡約順序の取り方は非常に重要であり，たくさんの順序のなかから完備化が成功する順序を見つけることが目的となる．

しかし単一の順序に対する KB を順次実行するのでは，発散した場合に次の順序へ処理を移すことができないため望ましい方法とはいえない．また，計算機のオペレーティングシステムが備えるマルチタスク機能によって複数の手続きを並行処理する場合（PKB）においても，共通の処理が多く無駄が多い．これらの問題点に対し，ATMS ノードとよばれるデータ構造に基づく複数の簡約順序を同時に処理する完備化手続き（MKB）が栗原らによって提案され，大幅な効率の改善がみられた．しかし，この手続きの実行にはあらかじめ停止性を満たす簡約順序を複数用意して与える必要がある．従って利用者には停止性や簡約順序に関する知識がある程度要求されるため，誰もが容易に完備化を扱えるとはいえない．

本論文では，簡約順序を経路順序に限定し，論理関数で表現する．経路順序とは，優先順位とよばれる関数記号の集合上の半順序を構文的に拡張して得られる順序である．任意の経路順序は優先順位の組み合わせで決まるので，関数記号同士の順序関係 $f \succ g$ の有無を論理変数 x_{fg} の真偽に対応させることで，項書換え系の停止条件

を論理関数で表現できる．これにより，MKB におけるプロセスの数を大幅に節約し，かつ利用者が簡約順序をあらかじめ用意することなく完備化を行うことを可能にした複数簡約順序完備化手続き MKB_{po} を提案する．さらに論理関数を二分決定グラフによって処理することによって問題の規模の増大に対する効率の維持をはかる．また，実際の問題に提案手法を適用した実験を行い，従来手法に対する優位性を検証する．

2. 項 書 換 え 系

今後の議論に現れる基本的な定義や記号は参考文献 [Dershowitz 90][井田 91][Terese 03] に詳しい．以下では本論文で必要な事項の概略のみを述べる．

2·1 等式系と項書換え系

関数記号の集合を $\mathcal{F}$，関数記号を $f,g,h,\cdots$，変数の集合を $\mathcal{V}$，変数記号を $x,y,z,\cdots$ で表す．$\mathcal{F},\mathcal{V}$ から構成される項の集合を $\mathcal{T}(\mathcal{F},\mathcal{V})$ で表す．また，項 t に現れる変数の集合を $Var(t)$ で表す．

特別な定数記号□を 1 つ含む項を文脈といい，$c[]$ で表す．$c[]$ の□を項 s で置き換えて得られる項を $c[s]$ と表す．項 s を $c[s]$ の部分項という．特に $c \neq \Box$ のとき項 s は $c[s]$ の真部分項である．($\equiv$ は構文的に等しいことを表す)

代入 θ は変数の集合 $\mathcal{V}$ から項の集合 $\mathcal{T}(\mathcal{F},\mathcal{V})$ への写像 $\theta:\mathcal{V}\mapsto\mathcal{T}$ である．代入は以下のようにして，$\mathcal{T}(\mathcal{F},\mathcal{V})$ から $\mathcal{T}(\mathcal{F},\mathcal{V})$ への写像に拡張される．

$$t = f(t_1,\ldots,t_n) \text{ のとき,} \theta(t) = f(\theta(t_1),\ldots,\theta(t_n))$$

慣例に倣い $\theta(t)$ を $t\theta$ と表す．ある代入 θ に対し，$t\theta = u$ が成り立つとき，項 u は項 t のインスタンスであるという．

等式 $s \leftrightarrow t$ は $\mathcal{T}(\mathcal{F},\mathcal{V})$ 中の項の非順序対である．等式系 $\mathcal{E}$ は等式の有限集合である．$l \leftrightarrow r$ が $\mathcal{E}$ の等式であり，$s \equiv c[l\theta], t \equiv c[r\theta]$ となるような文脈 $c[]$ と代入 θ が存在するとき $s \leftrightarrow_{\mathcal{E}} t$ または $t \leftrightarrow_{\mathcal{E}} s$ と表す．

書換え規則（以後，ルール）$l \to r$ は $\mathcal{T}(\mathcal{F},\mathcal{V})$ 中の項の順序対である．項書換え系 $\mathcal{R}$ はルールの有限集合である．$s \equiv c[l\theta], t \equiv c[r\theta]$ となるような文脈 $c[]$ と代入 θ が存在するとき $s \to_{\mathcal{R}} t$ と表し，項 s は項 t に書換えられる（リダクションされる）という．

項の中のルールが適用できる部分をリデックスとよぶ．項 s が 0 回以上の書換えで項 t に到達できるときにも，同様に s は t に書換えられるといい，この場合 $s \overset{*}{\to}_{\mathcal{R}} t$ とかき，これは $\to_{\mathcal{R}}$ の反射推移閉包を表す．特に 1 回の書換えを強調するときは 1 ステップの書換えという．$s \leftarrow_{\mathcal{R}} u \to_{\mathcal{R}} t$ の形をした 2 ステップでの $s \leftrightarrow_{\mathcal{R}} t$ の証明をピークという．

項 s がリデックスをもたないとき，既約であるという．$s \overset{*}{\to} t$ で t が既約であるとき，s は正規形 t をもつという．項書換え系 $\mathcal{R}$ での項 s の任意の正規形を $s\downarrow_{\mathcal{R}}$ で表す．正規形は項書換え系における計算結果とみなせる．

$l \to r$，$s \to t$ を項書換え系 $\mathcal{R}$ 中のルールとする．(ただし，2 つのルール間で同名の変数が出現しないように，適切な名前変えがされていると仮定する.) また，ある文脈 $c[]$ と非変数 u に対して $s \equiv c[u]$ であるとする．$u\theta \equiv l\theta$ となる代入 θ（但し θ は u と l の最汎単一化子）が存在するとき $s\theta \equiv c\theta[u\theta]$ は $t\theta$ にも $c\theta[r\theta]$ にも書換えられる．この 2 項から作られる等式 $t\theta \leftrightarrow c\theta[r\theta]$ を危険対という．$\mathcal{R}$ 中のルールから得られる危険対の集合を $cp(\mathcal{R})$ と表す．危険対をそれぞれ書換えて正規形を求めたとき，それらが一致するならば危険対は収束するといい，一致しないならば発散するという．

2·2 停止性と簡約順序

項書換え系 $\mathcal{R}$ において，全ての項に対する書換えが必ず止まるとき $\mathcal{R}$ は停止性を満たすという．すなわち，項書換え系 $\mathcal{R}$ において，$s_0 \to_{\mathcal{R}} s_1 \to_{\mathcal{R}} \cdots$ なる無限の書換え列が存在しないならば，$\mathcal{R}$ は停止性を満たす．

簡約順序 $>$ は項の集合 $\mathcal{T}(\mathcal{F},\mathcal{V})$ 上で置き換え，代入に対して閉じた整礎な半順序である．$>$ が整礎であるとは，$\mathcal{T}(\mathcal{F},\mathcal{V})$ 中の項の無限の下降列が存在しないことである．このような簡約順序のもとで，以下の定理によって停止性が保証される．

定理（停止定理） 項書換え系 $\mathcal{R} = \{l_i \to r_i | 1 \le i \le n\}$ は，

$$l_i > r_i, \quad \text{for all} \quad i \in \{1,\ldots,n\}$$

となる簡約順序 $>$ が $\mathcal{T}(\mathcal{F},\mathcal{V})$ 上に存在するならば停止する．

簡約順序として経路順序がよく用いられる．経路順序は関数記号の集合上の半順序を項の集合の上に構文的に拡張して定義される半順序である．経路順序として辞書式経路順序，再帰的経路順序，再帰的分割順序など [Dershowitz 87] が知られているが，本論文では辞書式経路順序を用いる．

定義（辞書式経路順序） $\succ$ を固定個の引数を取る関数記号の集合 $\mathcal{F}$ 上の半順序とする．このとき項の集合 $\mathcal{T}(\mathcal{F},\mathcal{V})$ 上の辞書式経路順序 $\succ_{lpo}$ は以下のように再帰的に定義される．ただし，項 s,t が変数ではないときには，$s \equiv f(s_1,\ldots,s_m), t \equiv g(t_1,\ldots,t_n)$ とおく．

$s \succ_{lpo} t$ であることは，$s \notin \mathcal{V}$ で，かつ，次のいずれかを満たすことと定義する．

(1) $t \in Var(s)$

(2) $s_i \succeq_{lpo} t$ for $1 \le \exists i \le m$

(3) $f \succ g$ and $s \succ_{lpo} t_j$ for $1 \le \forall j \le n$

(4) $f = g$ and $1 \le \exists k \le n$,

$$\begin{cases} s_i \equiv t_i \text{ for } 1 \le \forall i < k, \\ s_k \succ_{lpo} t_k,\ s \succ_{lpo} t_j \text{ for } k < \forall j \le n \end{cases}$$

上記の定義の $\mathcal{F}$ 上の半順序 $\succ$ を優先順位とよび，正確にはこれは厳格半順序である (厳格半順序:反射性のかわりに非反射性を満たす半順序)．なお，2つ目の条件に現れる $s \succeq_{lpo} t$ とは，$s \succ_{lpo} t$ または $s \equiv t$ を表す．

重要なのは，関数記号の集合上の優先順位が，項の集合上の順序を決定する点である．任意の経路順序は優先順位の取り方によって表現することができる．また，本論文では，辞書式経路順序を扱うが，再帰的経路順序等の他の経路順序は単に 2 項の大小関係の取り方の違いから生じる順序であるため，本論文における議論は辞書式経路順序に限らず他の経路順序においても成立する．

2·3 合流性と完備性

項書換え系では，ある項 s に対して適用可能なルールが複数存在する場合がある．項の中にリデックスが複数存在する場合には何通りもの書換えが生じ，ルールの適用の順番によって結果が異なることが普通である．任意の項について，適用するルールの順番に依らず書換えの結果が一意に定まるとき，書換え系は合流性を満たすという．すなわち，任意の項 s,t,u に対し，$s \overset{*}{\leftarrow}_{\mathcal{R}} u \overset{*}{\rightarrow}_{\mathcal{R}} t$ であるとき，$s \overset{*}{\rightarrow}_{\mathcal{R}} v \overset{*}{\leftarrow}_{\mathcal{R}} t$ となる適当な項 v が存在するとき $\mathcal{R}$ は合流性を満たす．また，任意の項 s,t,u に対し，$s \leftarrow_{\mathcal{R}} u \rightarrow_{\mathcal{R}} t$ であるとき，$s \overset{*}{\rightarrow}_{\mathcal{R}} v \overset{*}{\leftarrow}_{\mathcal{R}} t$ となる適当な項 v が存在するとき項書換え系 $\mathcal{R}$ は弱合流性を満たすという．

補題（Newman の補題） 停止性を満たす項書換え系 $\mathcal{R}$ が弱合流性を満たすならば，合流性も満たす．

補題（危険対補題） 任意のピーク $s \leftarrow_{\mathcal{R}} u \rightarrow_{\mathcal{R}} t$ に対して，書換え証明 $s \overset{*}{\rightarrow}_{\mathcal{R}} v \overset{*}{\leftarrow}_{\mathcal{R}} t$ または危険対証明 $s \equiv c[p\theta] \leftrightarrow c[q\theta] \equiv t$ が存在する（但し $p \leftrightarrow q \in cp(\mathcal{R})$ は危険対）．

Newman の補題，危険対補題および弱合流性の定義から，停止する項書換え系 $\mathcal{R}$ は，全ての危険対 $p \leftrightarrow q$ が収束するとき，すなわち適当な項 r が存在し $p \overset{*}{\rightarrow}_{\mathcal{R}} r \overset{*}{\leftarrow}_{\mathcal{R}} q$ を満たせば合流性を満たす．

停止性と合流性の両方を満たす項書換え系を完備であるという．等式系 $\mathcal{E}$ から得られた項書換え系 $\mathcal{R}$ が完備であるならば，以下の性質が成立する．

(1) 項 s の正規形 $s\downarrow_{\mathcal{R}}$ は唯一に定まる．
(2) $\mathcal{E}$ のもとで等式 $s = t$ が成立する必要十分条件は，$s\downarrow_{\mathcal{R}}$ と $t\downarrow_{\mathcal{R}}$ が一致することである．

このとき，任意の項 s,t について $s \overset{*}{\leftrightarrow}_{\mathcal{E}} t$ であることと $s\downarrow_{\mathcal{R}} \equiv t\downarrow_{\mathcal{R}}$ であることは同値となる．すなわち，完備な書換え系において，項 s と t が $\mathcal{E}$ のもとで等価であるか否かという $s = t$ の証明問題は $s\downarrow_{\mathcal{R}}$ と $t\downarrow_{\mathcal{R}}$ を比較することで決定可能となる．

与えられた等式仕様に対し，停止性と合流性の両方を満たすよう向き付けを行い完備な書換え系を得る操作を完備化手続きという．

3. 単一簡約順序下の完備化手続き

3·1 Knuth-Bendix の完備化手続き

最も基本的な完備化手続き (KB) は Knuth と Bendix によって与えられた [Knuth 70]．前節の補題に基づき，停止性が保証された項書換え系に対し，ルールによって生じる全ての危険対を収束させることで完備な系を得る．

KB に等式仕様を与え実行すると，繰り返し中の手続きによって危険対を収束させつつ与えられた等式仕様と矛盾しないよう論理的に等価な変換が行われる．手続きが停止すれば，与えられた等式仕様と等価で完備な書換え系が得られるか，もしくは得られず失敗する．なお，KB は危険対を永久に生成し続けて繰り返しが停止しない場合 **(完備化の発散)** がある半アルゴリズムである．

Bachmair らは KB を 6 本の抽象的推論規則として定式化した [Bachmair 86](図 1)．$>$ は簡約順序，$\mathcal{E}$ は任意の等式系，$\mathcal{R}$ は全てのルール $l \rightarrow r$ に対し $l > r$ が成り立つ任意の項書換え系，$\triangleright$ は項の集合上の整礎な順序であり，本論文ではこの $\triangleright$ を包含順序 (*encompassment ordering*) として用いる．$s \triangleright t$ であるとは s の部分項が t の代入例であり，その逆が成り立たないとき，かつそのときに限る．

DELETE: 自明な等式を削除
$(\mathcal{E} \cup \{s \leftrightarrow s\};\ \mathcal{R}) \vdash (\mathcal{E};\ \mathcal{R})$
ORIENT: 等式を $>$ に従って向き付け
$(\mathcal{E} \cup \{s \leftrightarrow t\};\ \mathcal{R}) \vdash (\mathcal{E};\ \mathcal{R} \cup \{s \rightarrow t\})$ if $s > t$
SIMPLIFY: 等式の左辺あるいは右辺を書換え
$(\mathcal{E} \cup \{s \leftrightarrow t\};\ \mathcal{R}) \vdash (\mathcal{E} \cup \{s \leftrightarrow u\};\ \mathcal{R})$ if $t \rightarrow_{\mathcal{R}} u$
COMPOSE: ルールの右辺を書換え
$(\mathcal{E};\ \mathcal{R} \cup \{s \rightarrow t\}) \vdash (\mathcal{E};\ \mathcal{R} \cup \{s \rightarrow u\})$ if $t \rightarrow_{\mathcal{R}} u$
COLLAPSE: ルールの左辺を書換え
$(\mathcal{E};\ \mathcal{R} \cup \{t \rightarrow s\}) \vdash (\mathcal{E} \cup \{u \leftrightarrow s\};\ \mathcal{R})$
if $l \rightarrow r \in \mathcal{R},\ t \rightarrow_{\{l \rightarrow r\}} u,$ and $t \triangleright l$
DEDUCE: 危険対の生成
$(\mathcal{E};\ \mathcal{R}) \vdash (\mathcal{E} \cup \{s \leftrightarrow t\};\ \mathcal{R})$ if $s \leftarrow_{\mathcal{R}} u \rightarrow_{\mathcal{R}} t$

図 1 KB 推論規則

等式系 $\mathcal{E}$ と項書換え系 $\mathcal{R}$ に対し推論規則を 1 つ適用して $\mathcal{E}'$ と $\mathcal{R}'$ が得られる場合，$(\mathcal{E};\ \mathcal{R}) \vdash_{KB} (\mathcal{E}';\ \mathcal{R}')$ とかく．以下 $\vdash_{KB}$ を $\vdash$ と略記する．手続きは与えられた等式仕様 $\mathcal{E}_0$ および空の項書換え系 $\mathcal{R}_0$ を出発点とし，完備化手続きの変更列 $(\mathcal{E}_0;\ \mathcal{R}_0) \vdash (\mathcal{E}_1;\ \mathcal{R}_1) \vdash \cdots \vdash (\mathcal{E}_\infty;\ \mathcal{R}_\infty)$ を得る．このとき $\mathcal{E}_\infty$ が空になりかつ $\mathcal{R}_\infty$ が収束するならば完備化は成功する．$\mathcal{E}_\infty$ が空でなければ，完備化は失敗する．なお，推論規則 DEDUCE を適用する際に生成される危険対の親となるルールの選び方についての条件があり，いつまでも選ばれないようなルールがないようにしなければならない．すなわち全てのルールが有限ステップの間に必ず選ばれる必要がある．これを公平であるという．また，上記の完備化手続きは適用する推論規則の順番等は問わないが，推論規則の選択についても公平性を期す必要がある．すなわち，いつまでも選ばれな

いような推論規則がないようにしなければならない．以上に留意して，6 本の推論規則を順次適用するならば，得られる $\mathcal{R}_\infty$ は収束する．

3·2 KB の問題点

前節でみた通り KB の手続きは極めて簡明な形で完備化を行うが，項書換え系の停止性を保証するために必要な簡約順序を利用者に要求するという問題点がある．このため利用者には停止性に関する知識が要求され，また知識を有していたとしても，適切な簡約順序を与えることは難しい．

また，与える簡約順序に依存して手続きの結果が成功，失敗，発散のいずれかとなるが，発散の場合は手続きが停止しないことも大きな問題である．これを避けるための明らかな解決法は与える簡約順序を変更することであるが，これは第 1 の問題点同様，どの簡約順序に変更するかを与えることが困難である．またそれが可能だとしても，考えうる簡約順序の数は複数存在するため，適切な順序を与えるということが難しい．そこで，適切な簡約順序の候補を複数個用意してそれぞれ別個に完備化を行うことが現実的な解決法として考えられる．

しかし，KB は簡約順序の取り方によっては手続きが発散し処理が停止しない．つまり，複数の簡約順序に対する完備化手続きを逐次行っていくのでは，途中で完備化が発散するようなタスクが生じた場合，その順序を諦めて次の簡約順序へ処理を移すことができなくなる．従って，複数の完備化手続きのプロセスを並行にマルチタスク処理で動作させることで解決することになるが，この方法では各プロセス間に重複した推論が生じるなどして効率がよくない．

4. 複数簡約順序下の完備化手続き

前節で見た通り，単一の簡約順序に対する完備化手続きでは複数の簡約順序から効率的に解を得ることは難しい．栗原らは，ATMS(*Assumption-based truth maintenance system*) とよばれる推論システムの推論を効率良く行わせる補助機構を用い，問題解決器 (完備化手続き) の重複した推論や既に得られた推論結果の再利用を行うことで，複数の簡約順序を同時に取り扱う完備化手続き (MKB) を提案した [Kurihara 99].

ATMS は問題解決器の推論結果を ATMS ノードとして記録する真偽維持システムである．ATMS ノードは $\langle datum, label, justification\rangle$ の 3 つのスロットを持ち，それぞれ，$datum$ は問題解決器が推論して得た事実，$label$ はその事実を成立させる環境の集合 (環境は問題解決器が推論を行うために用いた仮定の集合)，$justification$ は他の ATMS ノードからどのようにこのノードが導かれたかを示し，推論間の依存関係を表す．以上のノードを処理することにより，ATMS はラベル中に複数の環境を保持し，多重文脈を扱うことを可能にする．本論文の提案手法の基盤となる複数簡約順序完備化手続きを，次節にて詳説する．

4·1 複数簡約順序完備化

MKB において，任意の簡約順序の有限集合を $O = \{\succ_1,\ldots,\succ_n\}$，$I = \{1,\ldots,n\}$ はそのインデックスの集合とする．簡約順序の集合 O および等式仕様 $\mathcal{E}$ を与えられ，MKB は平行プロセス $\{P_1,\ldots,P_n\}$ をシミュレートする．ここでプロセス P_i は簡約順序 $\succ_i$ のもとでの $\mathcal{E}$ に対する 単一の KB を実行していることと同等となるため，集合 I を並行して動作するプロセスのインデックスとして見なすことができる．

単純に KB を簡約順序の数だけ並行して実行するのでは，各プロセス間で非常に関連の深い推論が存在するため (全く同じ推論が現れる場合さえある)，所要時間の面でも，必要となる記憶容量の面でも非常に効率が悪い．そのため関連する KB の推論をまとめて 1 本の推論規則で行うような設計を行うことで効率化が可能となる．KB が $\mathcal{E}$ と $\mathcal{R}$ に対する推論を行ったのに対し，MKB では ATMS ノードに基づくデータ構造の集合 N に対する推論規則として形式化される．

定義（ノード） 4 つ組 $\langle s:t, L_1, L_2, L_3\rangle$ をノードという．ただし，$s:t$ は項 s,t の順序対，L_1, L_2, L_3 はラベルと呼ばれる I の部分集合で，下記の条件（ラベル条件と呼ぶ) を満たすものとする．

- L_1, L_2, L_3 は互いに素 (disjoint) である．
- $i \in L_1$ ならば $s \succ_i t$，かつ，$i \in L_2$ ならば $t \succ_i s$

第 1 スロットの s,t は ATMS ノードにおける $datum$ に，第 2,3,4 スロットは $label$ に対応し，$label$ 中の要素は環境 (プロセス) に対応し，$s:t$ に対する複数の環境の処理をひとまとめに行うことになる．直観的には，ラベル条件が保たれているならば L_1 が保持する環境において，項書換え系 $\mathcal{R}$ はルール $s \to t$ をもち，逆に L_2 が保持する環境においては $\mathcal{R}$ はルール $t \to s$ をもつことと考えることができる．L_3 が保持する環境では，項 s と t は $s \to t$ にも $t \to s$ にも向き付けられておらず，等号系 $\mathcal{E}$ が等式 $s \leftrightarrow t$ をもっていることととらえてよい．つまり，MKB では 3 つの $label$ をノードにもたせることにより，複数のプロセスにおける $s:t$ の状態 ($s \to t$, $t \to s$, $s \leftrightarrow t$) を同時に 1 つのノードで管理する．項の順序と L_1 と L_2 を入れ替えた $\langle s:t, L_1, L_2, L_3\rangle$ と $\langle t:s, L_2, L_1, L_3\rangle$ は同じものと見なす．

図 2 に MKB の推論規則を示す．推論規則 Rewrite-1 に現れる $t \doteq l$ とは，t が l のインスタンスであり，またその逆も成り立つことを意味する．すなわち，t と l は変数の付け替えを行うことで構文的に等しくなる．

推論規則 Delete は $datum$ が同じ項の対になり，また第 1 ラベルと第 2 ラベルが空で第 3 ラベルが空でないようなノードをノード集合から削除する．

DELETE:
$N \cup \{\langle s : s, \emptyset, \emptyset, L\rangle\} \vdash N \quad \text{if } L \neq \emptyset$

ORIENT:
$N \cup \{\langle s : t, L_1, L_2, L_3 \cup L\rangle\} \vdash N \cup \{\langle s : t, L_1 \cup L, L_2, L_3\rangle\}$
if $L \neq \emptyset, L_3 \cap L = \emptyset$, and $s \succ_i t$ for all $i \in L$

REWRITE-1:
$N \cup \{\langle s : t, L_1, L_2, L_3\rangle\} \vdash N \cup \left\{ \begin{array}{l} \langle s : t, L_1 \setminus L, L_2, L_3 \setminus L\rangle, \\ \langle s : u, L_1 \cap L, \emptyset, L_3 \cap L\rangle \end{array} \right\}$
if $\langle l : r, L, \ldots, \ldots\rangle \in N, t \rightarrow_{\{l \rightarrow r\}} u, t \doteq l$,
and $(L_1 \cup L_3) \cap L \neq \emptyset$

REWRITE-2:
$N \cup \{\langle s : t, L_1, L_2, L_3\rangle\}$
$\vdash N \cup \left\{ \begin{array}{l} \langle s : t, L_1 \setminus L, L_2 \setminus L, L_3 \setminus L\rangle, \\ \langle s : u, L_1 \cap L, \emptyset, (L_2 \cup L_3) \cap L\rangle \end{array} \right\}$
if $\langle l : r, L, \ldots, \ldots\rangle \in N, t \rightarrow_{\{l \rightarrow r\}} u, t \triangleright l$,
and $(L_1 \cup L_2 \cup L_3) \cap L \neq \emptyset$

DEDUCE:
$N \vdash N \cup \{\langle s : t, \emptyset, \emptyset, L \cap L'\rangle\}$
if $\langle l : r, L, \ldots, \ldots\rangle \in N, \langle l' : r', L', \ldots, \ldots\rangle \in N, L \cap L' \neq \emptyset$,
and $s \leftarrow_{\{l \rightarrow r\}} u \rightarrow_{\{l' \rightarrow r'\}} t$

図 2 MKB 推論規則

GC:
$N \cup \{\langle s : t, \emptyset, \emptyset, \emptyset\} \vdash N$

SUBSUME:
$N \cup \left\{ \begin{array}{l} \langle s : t, L_1, L_2, L_3\rangle \\ \langle s' : t', L'_1, L'_2, L'_3\rangle \end{array} \right\}$
$\vdash N \cup \{\langle s : t, L_1 \cup L'_1, L_2 \cup L'_2, L\rangle\}$
if $s : t$ and $s' : t'$ are the same (up to renaming of variables),
where $L = (L_3 \cup L'_3) \setminus (L_1 \cup L_2 \cup L'_1 \cup L'_2)$.

図 3 MKB 補助推論規則

ORIENT は項 s,t を $s \succ_i t$ の形で向き付けできる少なくとも 1 つのプロセス $P_i (i \in L_1)$ に対して KB-ORIENT(今後 KB における各規則を KB-ORIENT のように表記する) を適用することに相当する.(ただし，実際の実装においては,「少なくとも 1 つ」を「すべての」と解釈して実行することを想定する.)

REWRITE-1,REWRITE-2 はルール $l \rightarrow r$ で項 t を u に書換え，結果としてノード $\langle s : t,\ \ldots\rangle$ の $label$ を変更し，新たなノード $\langle s : u,\ \ldots\rangle$ を生成しノード集合に追加する．REWRITE-1 はプロセス $P_i (i \in L_1 \cap L)$ において KB-COMPOSE を，またプロセス $P_i (i \in L_3 \cap L)$ について KB-SIMPLIFY を適用することに対応し，$P_i (i \in (L_1 \cup L_3) \cap L)$ に対するこれらの処理はまとめてこの 1 つの推論規則で行われる．

REWRITE-2 はプロセス $P_i (i \in L_2 \cap L)$ において KB-COLLAPSE を行うことを REWRITE-1 に付加した推論規則である．結果はノード $\langle s : t,\ \ldots\rangle$ に対する $label$ の変更と，新たなノード $\langle s : u,\ \ldots\rangle$ の生成である．

DEDUCE は 2 つのノード間で生じる危険対を新たなノードとして生成しノード集合に追加する．危険対の生成に寄与するルールを保持する環境は新しいノードの第 3 ラベルに格納される．

以上 5 本の推論規則に加えて，補助的に図 3 に示す推論規則を追加する．これらは，KB のシミュレート自体については寄与しないが，MKB のノードの効率的な処理に貢献する．

GC (*garbage-collect*) は全ての $label$ が空であるようなノードを削除することにより，ノード集合のサイズを小さくする．

SUBSUME は変数の付け替えによって同一の $datum$ となるような 2 つのノードを 1 つにまとめる．

MKB は，等式仕様となる有限の等式の集合 (等式系) および簡約順序の集合 O を入力とし，初期のノード集合

$$N_0 = \{\langle s : t, \emptyset, \emptyset, I\rangle | s \leftrightarrow t \in \mathcal{E}\}$$

を出発点として推論する手続きである ($I = \{1, \ldots, |O|\}$)．明らかに N_0 はラベル条件を満たしている．また，上記の 7 本の推論規則を用いて推論を行う限り，N から得られる N' もまたラベル条件を満たす.

4·2 辞書式経路順序 $\succ_{lpo}$ への限定

第 2 章で述べた通り，経路順序は優先順位とよばれる関数記号の集合上の半順序を構文的に拡張してできる順序である．優先順位の取り方に依存することを考慮すると，経路順序は関数記号 f,g 同士の関係がある ($f \succ g$) のか，ない ($f \not\succ g$) のか，という 2 値の優先順位の複数の組み合わせで決定する．すなわち，簡約順序としてノードの $label$ に格納する対象は考えうる経路順序の数だけあるインデックスではなく，その順序を決定付ける優先順位の取り方に関する情報の方がより適しているといえる．

本論文では，完備化に用いる簡約順序を経路順序に限定し (具体的には辞書式経路順序について議論する)，優先順位に基づく構造から帰納的に求めた論理関数によって表現する．論理関数で表現された辞書式経路順序は，ただ 1 つのプロセスで扱うことが可能となる．簡約順序を論理関数で扱うことによって，手続きを始めるにあたって MKB に与えられる環境は，考えうる全ての辞書式経路順序を保持することができる．

また，前節のインデックスによって考えうる全ての辞書式経路順序を扱おうとすると，関数記号の集合 $\mathcal{F}$ についてのその総数 ($|\succ_{lpo}^{\mathcal{F}}|$ と表記するものとする) は関数記号が 2 つの場合で 3，3 つの場合で 19，4 つの場合で 219，5 つの場合で 4231，6 つの場合で 130023, $\cdots$ となり指数的に考慮すべき順序 (プロセス) の数が増加する．本手法では $label$ が保持する辞書式経路順序の総数は 0 から最悪の場合で $|\succ_{lpo}^{\mathcal{F}}|$ となるが，実際には，考慮すべきプロセスが増大する前に完備化が成功することにより，大幅に処理の対象となるプロセスを節約できることが期待される．

この考え方に基づき，本節では簡約順序をより効率的に表現できるようノードのデータ構造および推論規則に多少の変更を加え，ノード内の $label$ に保持されていたインデックスを複数の辞書式経路順序を表現する論理関

数に置き換えてより効率化を図った複数簡約順序下の完備化手続き (MKB_{po}) を設計する．提案手法は簡約順序を経路順序に特化することにより簡約順序全体に対する汎用性を失うが，一方で並行して処理されるプロセスが考えうる全ての経路順序を網羅している点で優れている．また，経路順序に対しては辞書式経路順序のみならず他の経路順序についても同様に用いることができる．

4·3 論理関数 $lpo_{s,t}(X)$ による $\succ_{lpo}$ の表現

本論文では，優先順位の取り方を論理変数で表し，その組み合わせを論理和・論理積で表すことで，対応する辞書式経路順序を論理関数の形で表現する．そのため関数記号同士の順序を表す論理変数 x_{fg} を導入する．$\mathcal{F}$ を項 s,t 中に現れる関数記号の集合とする．$X=\{x_{fg}|f,g\in\mathcal{F},f\neq g\}$ 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．$f\neq g$ の条件は，優先順位が厳格半順序であることから，同一の関数記号同士の順序を表す論理変数は不要なために付け加える．この論理変数 x_{fg} を

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

とすることで，集合 X の全ての論理変数への真偽の割当てによって任意の優先順位 $\succ$ を表現でき，すなわち任意の辞書式経路順序をも表現することが可能である．また，このようにして論理関数によって表現された複数の順序の論理和を取ることで，複数の順序を同時に取り扱うことが可能となる．

完備化手続きにおいて，項の対 $s:t$ が与えられた際に推論規則 Orient で向き付けを行う機会が生じるため，$s\to t$，または $t\to s$ として向き付けする経路順序がどのようなものであるのかを知らなくてはいけない．s，t を固定すると，$\succ$ に依存して $s\succ_{lpo}t$ が成り立つか否かが一意に決まる．そこで，$\succ$ を表現する X への真偽の割当て $\alpha_i(1\leq i\leq 2^{|X|})$ が与えられたとき，$s\succ_{lpo}t$ が成り立てば 1，さもなくば 0 を値とする論理関数 $lpo_{s,t}(X)$ を考える（図 4）[近藤 98].

なお，優先順位は半順序であることから前述の非反射性に加え，さらに 2 つの制約

- 推移性（$f\succ g$ かつ $g\succ h$ ならば $f\succ h$）
- 反対称性（$f\succ g$ ならば $g\not\succ f$）

を満たす必要がある．これらの制約を満足するときに限り 1 を値とする論理関数はそれぞれ

$$tr(X)=\prod_{f,g,h\in\mathcal{F}}[\bar{x}_{fg}+\bar{x}_{gh}+x_{fh}]$$

$$as(X)=\prod_{f,g\in\mathcal{F}}\bar{x}_{fg}+\bar{x}_{gf}$$

で与えられる．

以上を考慮すると，ある項の対 $s:t$ が与えられたとき，等式を $s\to t$ に向き付け，かつ半順序であるという制約を満たすような辞書式経路順序はこれらの論理積

$$lpo_{s,t}(X)\cdot tr(X)\cdot as(X)$$

の値を 1 とするような変数の割当て α_i で表される．

4·4 MKB_{po} のノードと推論規則

MKB_{po} で扱うノードは 5 つ組 $\langle s:t,F_1,F_2,F_3,G\rangle$ とする．$s:t$ は項 s,t の順序対である．ラベル F_1,F_2,F_3 は辞書式経路順序を表す論理関数であり，G はその論理和である．

基本的な構造は MKB におけるノードと同様であり，$datum$ および $label$ の意味は変わらない．ただし $label$ にはインデックスではなく論理関数が保持されるという点でのみ異なる．また，第 5 スロットとして G が追加されているが，これは演算の効率を上げるため単に L_1,L_2,L_3 の論理和を保持するものであり，MKB の手続き自体には大きく変更が加わるわけではない．

MKB_{po} の推論規則を図 5 に示す．基本的な考え方は MKB と同様であり，ラベルの表現が論理演算に変更されている以外は前節に示した通りである．推論規則 Delete のラベルに現れる論理関数 0 は，MKB における空のインデックス集合，すなわち該当するプロセスはないことを示す．

MKB と MKB_{po} で大きく違うのは推論規則 Orient である．（ここで用いられている $\succ_i$ は，X への真偽の割当て α_i が表す優先順位により導かれる辞書式経路順序である．）MKB において Orient は全てのプロセスを走査して項 s と t の向き付けを行うため，ノードが保持するプロセスの数が大きくなるに従って計算時間も増大する．一方で MKB_{po} においては $s\to t$ に向き付けされる経路順序を表す論理関数 F を求めてノード内の論理関数との論理演算を数回行うのみであるので，保持する経路順序の数が大きくなることに対する影響は極めて少ない．

また，Orient の条件

$$\text{if } F\neq 0 \text{ and } s\succ_i t \text{ for all } \alpha_i \text{ with } F(\alpha_i)=1$$

は項の対 $s:t$ を $s\to t$ の向きに向き付けするような経路順序のみを F が保持することを意味する．F はそのような順序を少なくとも 1 つ保持すべきである．（ただし，MKB と同様に，実際の実装においては，「少なくとも 1 つ」を「すべての」と解釈して，図 4 の論理関数を利用して F を求めることを想定する．）

しかし，実際のノードへの変更にはこの F が表す全ての順序が関わるわけではない．Orient は等式を向き付ける操作であり，これは向き付けの対象となる等式を保持する環境 (ノード内の F_3 にあたる) に対してのみ行われるべきであるので，変更後のノードにおける F_1 は F との単純な論理和ではなく $F_1+F\cdot F_3$ となっている．これ以外の推論規則についても，論理演算を用いて適切な環境に対する操作となるよう設計されている．

$\mathcal{T}(\mathcal{F},\mathcal{V})$ 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．

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

ただし，k は $s_i \neq t_i$ を満たす最小の i $(1 \leq i \leq n)$ である．

図 4 論理関数 $lpo_{s,t}(X)$

DELETE: $N \cup \{\langle s:s,0,0,F,G\rangle\} \vdash N$ if $F \neq 0$

ORIENT:

$$N \cup \{\langle s:t,F_1,F_2,F_3,G\rangle\} \vdash N \cup \{\langle s:t,F_1+F\cdot F_3,F_2,\bar{F}\cdot F_3,G\rangle\}$$

if $F \neq 0$ and $s \succ_i t$ for all α_i with $F(\alpha_i)=1$

REWRITE-1:

$$N \cup \{\langle s:t,F_1,F_2,F_3,G\rangle\} \vdash N \cup \left\{ \begin{array}{l} \langle s:t,F_1\cdot\bar{F},F_2,F_3\cdot\bar{F},(F_1+F_3)\cdot\bar{F}+F_2\rangle, \\ \langle s:u,F_1\cdot F,0,F_3\cdot F,(F_1+F_3)\cdot F\rangle \end{array} \right\}$$

if $\langle l:r,F,\ldots,\ldots,\ldots\rangle \in N, t \to_{\{l\to r\}} u, t \doteq l$, and $(F_1+F_3)\cdot F \neq 0$

REWRITE-2:

$$N \cup \{\langle s:t,F_1,F_2,F_3,G\rangle\} \vdash N \cup \left\{ \begin{array}{l} \langle s:t,F_1\cdot\bar{F},F_2\cdot\bar{F},F_3\cdot\bar{F},(F_1+F_2+F_3)\cdot\bar{F}\rangle, \\ \langle s:u,F_1\cdot F,0,(F_2+F_3)\cdot F,(F_1+F_2+F_3)\cdot F\rangle \end{array} \right\}$$

if $\langle l:r,F,\ldots,\ldots,\ldots\rangle \in N, t \to_{\{l\to r\}} u, t \rhd l$, and $(F_1+F_2+F_3)\cdot F \neq 0$

DEDUCE:

$$N \vdash N \cup \{\langle s:t,0,0,F\cdot F',F\cdot F'\rangle\}$$

if $\langle l:r,F,\ldots,\ldots,\ldots\rangle \in N, \langle l':r',F',\ldots,\ldots,\ldots\rangle \in N$, $F\cdot F' \neq 0$, and $s \leftarrow_{\{l\to r\}} u \to_{\{l'\to r'\}} t$

GC:

$$N \cup \{\langle s:t,0,0,0\} \vdash N$$

SUBSUME:

$$N \cup \left\{ \begin{array}{l} \langle s:t,F_1,F_2,F_3,G\rangle \\ \langle s':t',F_1',F_2',F_3',G'\rangle \end{array} \right\} \vdash N \cup \{\langle s:t,F_1+F_1',F_2+F_2',F,G+G'\rangle\}$$

if $s:t$ and $s':t'$ are the same (up to renaming of variables), where $F = (F_3+F_3')\cdot\overline{(F_1+F_1')}\cdot\overline{(F_2+F_2')}$.

図 5 MKB_{po} 推論規則

procedure $mkb_{po}(\mathcal{E},\mathcal{F})$;
begin
  $N_o := \{\langle s:t,\ 0,\ 0,\ wild,\ wild\ \rangle \mid s \leftrightarrow t \in \mathcal{E}\}$
  $N_c := \emptyset$
  while $success(N_o,N_c) =$ **false do**
    if $N_o = \emptyset$ **then return(fail)**
    else
      $n := choose(N_o)$;
      $N_o := union(N_o - \{n\}, delete(rewrite(\{n\},N_c)))$;
      if $n \neq \langle\ldots,0,0,0\rangle$ **then**
        $n := orient(n)$;
        if $n \neq \langle\ldots,0,0,\ldots\rangle$ **then**
          $N_o := union(N_o, delete(rewrite(N_c,\{n\})))$;
          $N_c := gc(N_c)$;
          $N_o := union(N_o, delete(deduce(n,N_c)))$
          $N_o := gc(delete(union(N_o, rewrite(N_o,N_c))))$
        end;
        $N_c := union(N_c,\{n\})$
      end
    end
  end
  return($\mathcal{R}[N_c,i]$) where $i = success(N_o,N_c)$
end mkb_{po}.

図 6 MKB_{po} 完備化手続き

4·5 MKB_{po} 完備化手続き

MKB_{po} におけるノード および推論規則を用いた完備化手続きを図 6 に示す．

基本となる考え方は人工知能における探索などで用いられる OPEN/CLOSED リストの手続きである．N_o, N_c をノードの集合とする．N_o は未推論ノード集合であり，推論が終わっていないノード (OPEN ノード) が格納され，OPEN リストに相当する．N_c は推論済ノード集合であり，相互の推論が済んだノード (CLOSED ノード) が格納され，CLOSED リストに相当する．

推論規則を単一のノードに適用される単一ノード規則と 2 つのノード間で適用される 2 ノード規則に分類する．DELETE, ORIENT, GC は前者，REWRITE-1 -2, DEDUCE, SUBSUME は後者に属する．

また，前節において，いかなる経路順序をも含まない論理関数を 0 としたが，その逆は考えうる経路順序を全て含む論理関数である．これを $wild$ とよぶこととする．これは経路順序として成立するうちで最も制約の緩い論理関数，$tr(X)\cdot as(X)$ を表現している．

この手続きは while ループによって繰り返しノードに対し推論が行われ，ループの最初で以下の条件を共に満たす経路順序が存在するときに成功と判定し停止する．

- 未推論ノード集合 N_o に属する全てのノードの全ての $label$，すなわち F_1,F_2,F_3 に環境として含まれない．この条件は実際には G を調べることで判定する．
- 推論済ノード集合 N_c に属する全てのノードの F_3 に環境として含まれない．

ループは，

(1) N_o が空ならば失敗と判定し，さもなくば次に進む．
(2) N_o から未推論ノードを 1 つ選択し，単一ノード規則を適用できなくなるまで繰り返し行う．
(3) 選択されたノードが単一ノード規則によって削除されていなければ引き続いて N_c ノード内の全てのノードとの間で 2 ノード推論規則を繰り返し適用する．
(4) この際生成されたノードは N_o に追加され，その後 N_o を N_c によって書換える．
(5) ステップ 2 で選択されたノードは推論規則の適用が完了したので推論済ノードとして N_c に追加される．

を繰り返す．さもなくば永久にループが繰り返される．

以上を踏まえ，中で用いられている細かな手続きについて具体的に述べる．

手続き $mkb_{po}(N_o,\mathcal{F})$ は MKB_{po} に基づいた完備化手続きを行い，完備化に成功すればその時点での完備な項

書換え系とその環境に対応する論理関数 (経路順序) を返す．初期値は N_o が与える等式仕様，$\mathcal{F}$ が関数記号の集合である．この手続きの実行中は，常に以下の条件を満たす．

- N_c に属する全ての CLOSED ノードは単一ノード規則の全てを適用済み．
- N_c に属する全ての CLOSED ノードの対は 2 ノード規則の全てを適用済み．

手続き $success(N_o, N_c)$ は完備化手続きが成功しているかどうか判定する．N_o の全てのノードに含まれず，また N_c の第 3 ラベルにも含まれないような経路順序を求めればよいので，考えうる全ての経路順序を表す論理関数 $wild$ から N_o 内の全てのノードに現れる論理関数 G を取り除き（否定との論理積をとる），さらに N_c 内の全ての第 3 ラベルに現れる論理関数を取り除いたものを考え，これが最後まで 0 にならず何らかの論理関数が残った場合，その論理関数の値を 1 とするような変数割り当てに対応する経路順序で完備化が成功したと判定する．なお，演算の途中で 0 になった時点で完備化は成功していないことになるのでそのときは即座に処理を次に移す．

手続き $choose(N_o)$ は N_o から未推論ノードを 1 つ選択する．この選択は公平に行われなければならない．ノードの選び方に制限はないが，選択が公平である限りにおいて $datum$ に現れる項のサイズがより小さいもの (サイズの尺度は，たとえば現れる関数記号の数など) を優先的に選んでいくことは，ヒューリスティックとして有効であることが知られている．

手続き $union(N, N')$ は N と N' の和集合を得る．この際，推論規則 Subsume を補助的に適用するが，適用の対象となるノードの対のうち少なくとも 1 つは N' から選択することで，適切なノード間での適用を行う．

手続き $delete(N)$ は N 内のノードに Delete を可能であれば適用し，その結果として削除されずに残ったノード集合を得る．この操作は Rewrite-1 -2 および Deduce によって生成されたノード集合に対しての適用のみが行われうることに注意する．

手続き $rewrite(N, N')$ は繰り返し Rewrite-1 -2 を $N \cup N'$ に対して行い，N' 内のノードに含まれるルールによって N 内のノードをそれ以上書換えが生じることがなくなるまで書換える．手続きの結果としてこの操作によって新たに生成されるノードの集合を得る．この際，N 内のノードは副作用的に変更されることに注意する．

手続き $orient(n)$ は Orient をノード n に対して適用し，結果として，適切な向き付けが済んだノードを得る．この際，ノード n 内の $label$ は副作用的に変更される．

手続き $gc(N)$ は N 内のノードに Gc を適用し，結果として消えずに残ったノード集合を得る．この処理は Rewrite によって変更されたノードに対して行われる．

4·6 簡単な例題

簡単な例題として，等式系 $\mathcal{E} = \{f(f(x)) \leftrightarrow g(x)\}$ の完備化を行い MKB_{po} の動作を概観する．初期値は

$$N_o = \left\{ \langle f(f(x)) : g(x),\ 0,\ 0,\ wild \rangle \right. \qquad (1)$$

$$N_c = \ \emptyset$$

であり（ノード内の第 5 スロット G は単に論理和 $L_1 + L_2 + L_3$ を保持するのみであるので，記述の簡便のため省略してある），$wild = x_{fg}\bar{x}_{gf} + \bar{x}_{fg}x_{gf} + \bar{x}_{fg}\bar{x}_{gf}$．次に手続き $choose$ によりノード (1) を選択し，手続き $orient$ で向き付けると

$$\langle f(f(x)) : g(x),\ x_{fg}\bar{x}_{gf},\ \bar{x}_{fg}x_{gf},\ \bar{x}_{fg}\bar{x}_{gf} \rangle \quad (1')$$

となる．次に，手続き $deduce$ によって，ノード $(1')$ とノード $(1')$ との間で作られる危険対 $f(g(x)) \leftrightarrow g(f(x))$ から新たにノード

$$\langle f(g(x)) : g(f(x)),\ 0,\ 0,\ x_{fg}\bar{x}_{gf} \rangle \qquad (2)$$

を得る．この時点でノード $(1')$ の推論が完了し，ノードの集合は

$$N_o = \left\{ \langle f(g(x)) : g(f(x)),\ 0,\ 0,\ x_{fg}\bar{x}_{gf} \rangle \right.$$

$$N_c = \left\{ \begin{array}{c} \langle f(f(x)) : g(x), \\ x_{fg}\bar{x}_{gf},\ \bar{x}_{fg}x_{gf},\ \bar{x}_{fg}\bar{x}_{gf} \rangle \end{array} \right.$$

となる．手続き $success$ で $wild \cdot \overline{x_{fg}\bar{x}_{gf}} \cdot \overline{\bar{x}_{fg}\bar{x}_{gf}} = \bar{x}_{fg}x_{gf}$ から，変数割り当て $\bar{x}_{fg}x_{gf} = 1$ すなわち優先順位 $g \succ f$ のもとで完備化が成功し，項書換え系

$$\mathcal{R} = \left\{ g(x) \rightarrow f(f(x)) \right.$$

を得る．

5. 二分決定グラフ

前章において論理関数を $label$ としたノードに対する完備化手続きを示したが，論理関数がどのような形で表現され処理されるのかということには触れていない．本論文では MKB_{po} 完備化手続きで扱う論理関数の表現として二分決定グラフ (BDD)[Akers 78, Bryant 86] を用いる [奥乃 96].

5·1 BDD の構造

BDD はグラフによる非常にコンパクトな論理関数表現である（図 7）[石浦 93]．円で囲んだノードは変数ノードとよばれ，論理変数でラベル付けされる．四角形で囲んだノードは定数ノードとよばれ，真理値 0 あるいは 1 でラベル付けされる．また，ノードから伸びる枝は，左の枝が 0 枝，右の枝が 1 枝とよばれ，論理変数の真偽に対応する．

BDD の各ノードは 1 つの論理関数を表す．根ノードから，変数に割り当てる真理値を元に枝を辿ってゆき，最終的に到達する定数ノードが，それまでに論理変数へ割り当てた真理値に対応する論理関数の真理値となる．

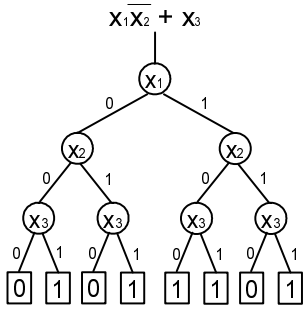

図 7 BDD

5·2 BDD の既約化と既約な BDD

ノード v の 0 枝と 1 枝が同じノードに伸びているとき，v は冗長なノードとよばれ削除できる．また，ノード u と v の 0 枝の先同士が同一のノードで，かつ 1 枝の先同士が同一のノードであるとき，u と v は等価なノードとよばれ一方を削除できる．冗長なノードと等価なノードの削除をそれ以上できなくなるまで繰り返す操作を既約化という．

ある変数の全順序が存在し，根から定数ノードに至る全てのパスについて，変数の順序がその全順序関係に矛盾しないとき，その BDD は順序付き BDD(orderedBDD) とよばれる．順序付き BDD に既約化をほどこして得られるものを既約な BDD という．既約な BDD は以下に示すような特長をもつ．

- 変数順序を固定すれば，既約な BDD は論理関数の標準形となる．
- 多くの実用的に重要な論理関数が現実的なノード数で表現可能（ $\mathcal{O}(n) \sim \mathcal{O}(n^2)$，最悪で $\mathcal{O}(2^n/n)$ ）．
- 論理関数に対する演算が表現のサイズに比例した時間で行える．

5·3 BDD の 効 率

標準的に実装をされた BDD は，以下のような利点をもつ．

- ある変数に 0 または 1 の値を代入して得られる関数の BDD を求める代入演算をグラフの大きさに比例した時間で行える．
- 充足可能かどうかの判定 (恒偽判定) をただちに行うことができる．
- 充足可能である場合，論理関数を 1 とするような変数割り当てを入力変数の数に比例した時間で求めることができる．

BDD は論理関数を計算機で扱う上で非常に優れた表現である [湊 93]．本論文では MKB_{po} において BDD を論理関数表現として用いる．

表 1 実験結果 1

等式系	順序の総数	MKB (sec)	MKB_{po} (sec)
E_3	$3! = 6$	5.42	5.57
E_4	$4! = 24$	31.68	32.40
E_6	$6! = 720$	36.01	35.11
E_7	$7! = 5040$	57.82	48.51
E_8	$8! = 40320$	221.70	110.01

6. 実　　験

前章までの準備に基づき，MKB および MKB_{po} を計算機に実装し，実際に両手法で等式系の完備化をそれぞれ行い処理時間を求め比較し問題の規模の増大に伴う変化を考察する．また，単一の簡約順序に対する完備化手続きである KB を複数並行して実行した PKB との処理時間の比を求め有効性を検証する．完備化のプログラムの実装は Java で行い，BDD を扱う機構として JavaBDD パッケージ（http://sourceforge.net/projects/javabdd）を利用した．JavaBDD は C で記述された BDD 処理ライブラリ BuDDy（http://www.itu.dk/research/buddy/）を JNI インタフェースによって利用し，Java プログラムで容易に扱えるクラスライブラリとして整備したものである．

6·1 実　　験　　1

関数記号の数および等式の数が異なる 5 つの等式系に対し，MKB および MKB_{po} で完備化を行う．問題の規模の爆発を抑えるため簡約順序を全順序とした．

5 つの等式系を完備化することで得られた書換え系を図 8 に示す．E_n は与えた等式仕様，R_n はそれから得られた完備な書換え系である．簡約順序の数と完備化に要した時間の関係を表 1 に示す．図 9 はそれをプロットしたグラフである．なお，横軸を対数スケールとしてある．

関数記号の数や等式の数によって所要時間は増大する傾向はあるが，生じる危険対の数や推論する対象のノードの選択のされ方によっても時間のかかり方は変わってくるため，数値自体よりも数値の相対的な違いに注目する．また，両手法間においての違いは ATMS ノードのデータ構造およびその推論規則の処理内容であり，手続きが実行する完備化の内容自体ついては同一である．よって，経過および結果は両者で完全に一致する．

E_3，E_4 ではごくわずかではあるが MKB の方が少ない所要時間で完備化を完了している．これは簡約順序の数がそれほど多くないことから，プロセスの数がさほど大きく影響していないためである．このような小規模な問題においては MKB_{po} で BDD のグラフを扱うオーバーヘッドが MKB でのプロセスの数によるオーバーヘッドよりも影響がやや大きいものと考えられる．E_6 において両者は逆転し，E_7，E_8 と問題の規模が大きくなるに従い両者の差はより顕著となっている．E_3 の段階で MKB_{po} が MKB と遜色ない性能を保持しており，本

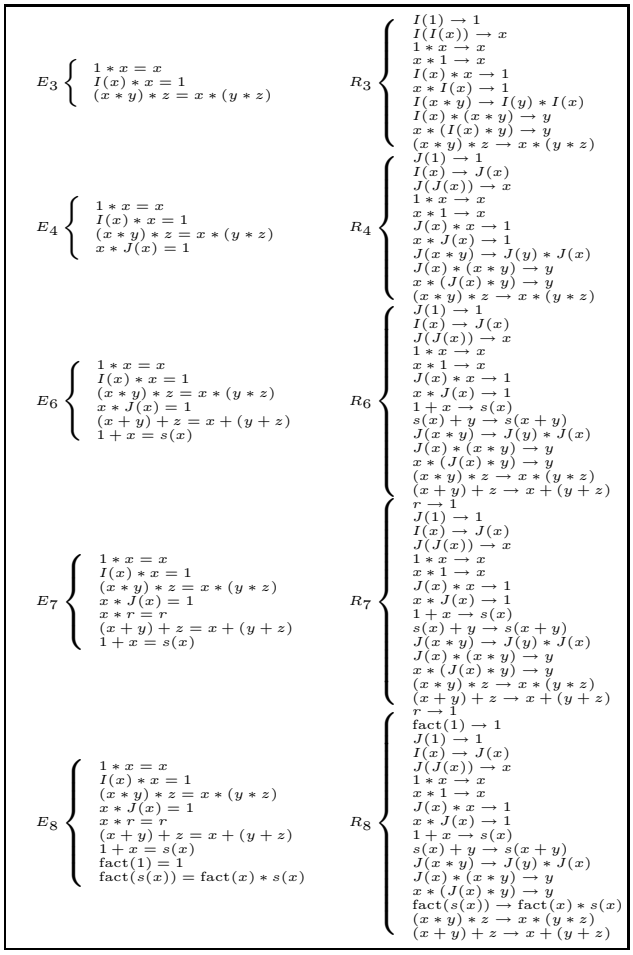

図 8 得られた完備な書換え系

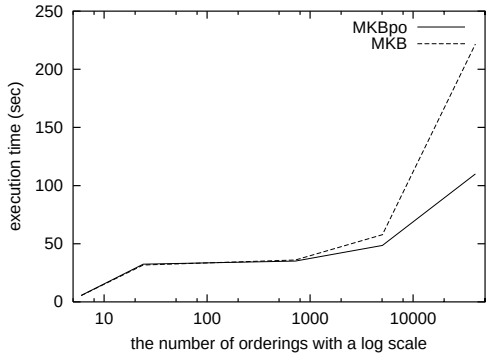

図 9 実験結果 1 のグラフ

表 2 実験結果 2（単位:sec）

タスク	簡約順序数	KB	PKB	MKB_{po}
P_1	1	1	1.03	1.10
P_2	6		5.13	2.18
P_3	30		25.62	2.29
P_4	120		119.10	2.28
P_5	360			2.52
P_6	720			3.61

手法は小規模問題においては MKB と同等，また大規模問題においては著しい効率の改善がみられた．

このような差異が生じる要因として，推論規則 ORIENT の処理の内容の違いが考えられる．MKB において ORIENT はノードの保持する全てのプロセスについて向き付けを行う．すなわちノード内に 40320 個の経路順序を保持している場合，たとえそれが明らかに全ての順序において項 s,t を $s \to t$ に向き付けられるとしても，必ず 40320 回の向き付けの反復処理が行われることとなる．一方，MKB_{po} では項 s,t 間の向き付けのできる経路順序を 1 度だけ論理関数の形で求め，それを用いてノード内の $label$ に保持する論理関数との論理演算をただ 1 度行うのみであるため，BDD による論理関数の効率的な処理の影響を含め大幅な効率の改善に寄与していると考えられる．

また等式の数が増えることによっても ORIENT を行う機会が増すため，同様にして両者のパフォーマンスの違いに影響を与えると考えられる．

6·2 実　　験　　2

本節では実験 1 における等式系 E_6 に対して，単一の簡約順序に対する完備化手続き KB を単純に複数並列に動作させる PKB と MKB_{po} の比較を行う．それぞれの所要時間を単一の簡約順序に対する単一の KB の所要時間で正規化し簡約順序の増大に伴う変化を検証する．タスクごとに与える簡約順序の数を別個用意する (関数記号は $\mathcal{F} = \{1, *, I, J, s, +\}$ の 6 つで最大 $6! = 720$ 通り)．各タスクにおける簡約順序は

- P_1 では $I \succ J \succ * \succ s \succ + \succ 1$ なる 1 個の簡約順序．
- P_2 では $I \succ J \succ * \succ s \succ$ を満たす 6 個の簡約順序．
- P_3 では $I \succ J \succ * \succ s$ を満たす 30 個の簡約順序．
- P_4 では $I \succ J \succ *$ を満たす 120 個の簡約順序．
- P_5 では $I \succ J$ を満たす 360 個の簡約順序．
- P_6 では全順序の制約を満たす 720 個の簡約順序．

とする．

簡約順序の数と完備化に要した時間の比 (KB における P_1 の所要時間を 1 とする) の関係を表 2 に示す．また，図 10 はそれをプロットしたグラフである．P_5 および P_6 において PKB は，360 個以上のプロセスを並行に動作することができなかったので空欄としてある．

PKB では，簡約順序の数と所要時間が正比例している．一方，$MKBpo$ においては，簡約順序の数が 720 倍に達していてもその所要時間は 3.5 倍程度に抑えられて

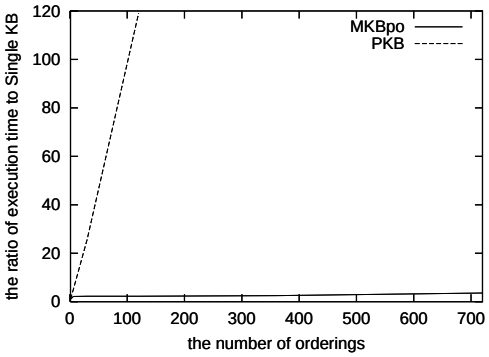

図 10 実験結果 2 のグラフ

おり，各プロセス間に存在する重複した推論が十分に効率的に処理されているといえる．

7. ま と め

本論文では，複数簡約順序に対する完備化手続きを辞書式経路順序に限定し，その構造から論理関数で表現することで一つの論理関数で複数の経路順序を一度に扱う MKB_{po} を提案し，実験により MKB_{po} の有効性を確認した．

また，論理関数表現として BDD を用いることによって効率的な処理を図り，実験により関数記号の増大に対する効率の維持を確認した．

今後の課題としては，他の論理関数表現での実装を行い比較することによる BDD での処理の優位性の検証や，OPEN/CLOSE ノードに基づく手続きのさらなる改善を考えている．

◇ 参 考 文 献 ◇

[Akers 78] Akers, S. B.: Binary decistion diagrams, *IEEE Trans. Comput.*, Vol.C-27, No.6, pp.509-516 (1978).

[Bachmair 86] Bachmair, L., Dershowitz, N., and Hsiang, J.: Orderings for equational proofs, *Proc. IEEE Symposium on Logic in Computer Science*, pp.346-357 (1986).

[Bryant 86] Bryant, R. E.: Graph-based algorithms for boolean function manipulation, *IEEE Trans. Comput.*, Vol.C-35, No.8, pp.677-691 (1986).

[Dershowitz 87] Dershowitz, N.: Termination of rewriting, *J. Symbolic Computation*, 3, pp.69-116 (1987).

[Dershowitz 90] Dershowitz, N. and Jouannaud, J.-P: Rewrite systems, *Handbook of Theoretical Computer Science*, Vol.B, pp.243-320 (1990).

[二木 83] 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).

[井田 91] 井田哲雄：計算モデルの基礎理論，岩波書店 (1991).

[石浦 93] 石浦菜岐佐: BDD とは，情報処理，Vol.34，No.5，pp.585-592 (1993).

[Knuth 70] Knuth, D. E. , Bendix, P. B. : Simple word promlems in universal algebras, in J.Leech(ed.), *Computational Problems in Abstract Algebra*, pp.263-297, Pergamon Press (1970).

[近藤 98] 近藤久，栗原正仁： 二分決定グラフを用いた項書換え系の停止性検証システム，人工知能学会誌，Vol.13，No.5，pp.822-834 (1998).

[Kurihara 99] Kurihara, M. and Kondo, H. : Completion for multiple reduction orderings, *Journal of Automated Reasoning*, Vol.23, No.1, pp.25-42 (1999).

[湊 93] 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).

[奥乃 96] 奥乃 博, 下國 治, 田中 英彦 : 二分決定グラフによる多重文脈型真偽維持システム BMTMS, 人工知能学会誌, Vol.11, No.3, pp.280-289 (1996).

[Terese 03] Terese : Term rewriting systems, Cambridge University Press, (2003).

[外山 01] 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).

〔担当委員：××○○〕

19YY 年 MM 月 DD 日　受理

著 者 紹 介

横山　壱星

2002 年 3 月 北海道大学工学部情報工学科卒業．2004 年 3 月 同大学大学院工学研究科システム情報工学専攻修士課程修了．同年，セイコーエプソン株式会社入社．在学中，項書換え系の研究に従事．情報処理学会会員．

栗原　正仁（正会員）

1978 年 3 月 北海道大学工学部電気工学科卒業．1980 年 3 月 同大学大学院工学研究科情報工学専攻修士課程修了．以後，同大学助手，講師，助教授および北海道工業大学教授を経て 2002 年 4 月より北海道大学教授．工学博士．知能情報学の研究に従事．情報処理学会，電子情報通信学会会員．