チュートリアル

# CafeOBJ 入門 (3) 等式推論と項書換システム

中村 正樹　二木 厚吉　緒方 和博

等式推論は代数仕様言語 CafeOBJ の最重要の推論機構である．等式推論の効率的な実行を可能とする項書換システムについて，CafeOBJ 仕様の作成の助けとなる知識に焦点を当てて解説する．項書換システムの停止性，合流性，十分完全性などの基本的性質を満たす仕様作成の指針を示し，AC 演算子属性，条件付き等式などを含む仕様への適用を議論する．

Equational inference is the most fundamental inference mechanism for CafeOBJ algebraic specification language. Term rewriting system can realize equational inference in an efficient way. Several pieces of fundamental knowledge on term rewriting systems, which are valuable for CafeOBJ specification development, are described. We present the ways to describe specifications which satisfies fundamental properties of term rewriting systems: termination, confluence and sufficient completeness properties, and also discuss about applications to specifications including associative and/or commutative operators, conditional equations, and so on.

## 1 はじめに

等式を公理とする仕様がある性質を満たすかは，公理を構成する等式からその性質が導けるかを推論することにより決定される．等式に基づく推論は，理工学分野において日常的に使われ，等式推論として定式化される．等式は，問題領域の事実や規則だけでなく，問題を解析するための前提や規則をも表現できる．したがって，等式推論は，問題レベルの推論のみならず，問題を解析するレベルの推論をも定式化できるという特徴を持つ．

等式を左辺から右辺への書換規則とみなし，与えられた表現を書き換えることで，その表現と等価なより簡単な表現を効率よく求めることが出来る．等式を書換規則として用いる書換操作は，等式推論に忠実であるので，書き換えによる簡約は効率的な等式推論を可能にする．

書換規則により表現 (つまり項) を簡約するシステムを項書換システムと呼ぶ．CafeOBJ システムは，仕様の公理を構成する等式を書換規則とみなすことで，仕様から項書換システムを構成し，仕様を実行する．CafeOBJ による仕様の検証は，仕様が定義する項書換システムと対話的に仕様を実行することで行われる．

本稿では，等式推論と項書換システムについて解説する．まず等式推論を簡単に紹介する．つぎに，演算子属性 (`assoc`, `comm`)，条件付き等式を持たない単純なデータ型仕様に対する項書換システムについて紹介し，後半で CafeOBJ の各種機能に対する項書換システムについて議論する．本稿における項書換システムに関する各種表記，定義，命題などは文献[5][4][1][7]を参照する．また等式仕様，CafeOBJ については文献[2]を参照する．

---

Introducing CafeOBJ (3) : Equational Reasoning and Term Rewriting Systems.

Masaki Nakamura, 金沢大学理工学域電子情報学類, School of Electrical and Computer Engineering, Kanazawa University.

Kokichi Futatsugi, Kazuhiro Ogata, 北陸先端科学技術大学院大学 情報科学研究科, Graduate School of Information Science, Japan Advanced Institute of Science and Technology (JAIST).

コンピュータソフトウェア, Vol.25, No.3 (2008), pp.69–80.
[解説論文] 2007 年 3 月 28 日受付.

## 2 等式推論

CafeOBJ 仕様の公理は等式の集合で与えられる†1. 項の対 $(s,t)$ が公理 $E$ から等しいと導けるかどうかは，項の集合 $T(\Sigma,X)$ 上の合同関係 $=_E$ により定式化される．合同関係は，合同律 (congruence)，代入律 (substitutivity) を満たす最小の等価関係†2である．CafeOBJ 仕様の公理を構成する等式は，`ceq` $l$ `=` $r$ `if` $c$ の形をしている．条件部 `if` $c$ がない等式は，$c=$ `true` の条件付き等式と同一視できる．

**定義 2.1** シグネチャ$\Sigma$，公理 $E$，項 $s,t \in T(\Sigma,X)$ に対し，等式 $s=t$ が以下の推論規則で導けるとき，$s =_E t$ と書く．

- **反射律・対称律・推移律**

$$\frac{}{t=t} \quad \frac{s=t}{t=s} \quad \frac{s=u \;\; u=t}{s=t}$$

- **合同律**

$$\frac{t_1=t'_1 \;\; \cdots \;\; t_n=t'_n}{f(t_1,\ldots,t_n)=f(t'_1,\ldots,t'_n)}$$

- **代入律**

$$\frac{\theta(c)=\texttt{true}}{\theta(l)=\theta(r)}$$

ただし，各項，演算子などは以下の要素である．$s,t,u \in T(\Sigma,X)_s$, $f \in \Sigma_{s_1\cdots s_n\,s}$, $t_i,t'_i \in T(\Sigma,X)_{s_i}$ $(i \in \{1,\ldots,n\})$, `ceq` $l$ `=` $r$ `if` $c \in E$, $\theta : Y \to T(\Sigma,X)$. 変数から項への写像である代入 $\theta$ は，自然に項から項への写像に拡張できる．ここで $Y$ は，条件付き等式 `ceq` $l$ `=` $r$ `if` $c \in E$ に出現する変数の集合である．明らかな場合，$=_E$ の $_E$ を略すことがある．また，仕様 $SP$ とその公理を同一視して，$=_{SP}$ と記すことがある．

直感的には，項のある部分が等式の左辺 (または右辺) のインスタンスになっているとき，そのインスタンスを対応する右辺 (または左辺) のインスタンスに置き換える，という動作を繰り返して得られる項は，合同関係にある．項の対が合同関係にあることを確かめる行為を**等式推論 (equational reasoning)** と呼ぶ．

**モデルと合同関係：** 反射律，対称律，推移律は，等価関係が満たすべき性質である．合同律は，等しい入力に対して等しい値を返すことを表しており，演算子が関数であるための条件である．代入律は，公理に含まれる条件付き等式が正しいことを表している．すべての推論規則は，任意のモデルで保存される性質であり，以下の命題が成り立つ．

**命題 2.2** シグネチャ$\Sigma$，公理 $E$，項 $s,t \in T(\Sigma,X)$ に対して，$s =_E t$ ならば任意の $(\Sigma,E)$-代数 $M$ に対して $M \models s=t$.

仕様のモデルは，$(\Sigma,E)$-代数であるため，$s =_E t$ ならば任意のモデル $M$ で $M \models s=t$ が成り立つ．

**合同関係と項書換システム：** 命題 2.2 より，仕様の検証 (ある性質が任意のモデルで成り立つかどうか) は，等式推論を基礎に行われることになる．**項書換システム (term rewriting system, TRS)** は，等式を方向性を持った書換と見なすことで，等式推論を効率的に行う基礎理論である．CafeOBJ の仕様実行は，TRS に基づいて行われる．

## 3 項書換システム

本節では，条件付き等式のない単純な等式仕様に対する項書換システムを紹介する．条件付き等式などを含む仕様に関しては 7 節で議論する．

CafeOBJ の実行では，変数条件を満たす等式を書換規則として用いて，項の簡約を行う．ここで変数条件とは，(1) 左辺が変数でなく，(2) 右辺に現れる変数はすべて左辺に含まれる，という条件である．変数条件を満たす等式 `eq` $l$ `=` $r$ を書換規則と呼び，$l \to r$ と書く．書換規則の集合を項書換システム (TRS) という†3.

仕様 `NAT+`の 2 つの等式は，変数条件を満たす．

```
mod! NAT+{
 [Zero NzNat < Nat]
 op 0 : -> Zero
 op s_ : Nat -> NzNat
 op _+_ : Nat Nat -> Nat
```

†1 CafeOBJ の公理として書換論理の公理としての書換規則や一階述語論理の論理式を記述することも可能であるが，本チュートリアルでは等式だけを公理とする等式仕様に焦点を絞り解説する．

†2 反射律 (reflexivity)，対称律 (symmetry)，推移律 (transitivity) を満たす二項関係を等価関係と呼ぶ．

†3 理論と書換規則の集合の両方を TRS と呼んでいる．

```
 vars M N : Nat
 eq N + 0 = N .
 eq N + s M = s(N + M).
}
```

よって仕様実行時に書換規則として用いられる．等式の向きは仕様のモデルに影響を与えないが，実行には大きな影響を与える．等式宣言 `eq N + 0 = N` を逆向きの `eq N = N + 0` と変更しても，NAT+のモデルは変わらないが，TRS として見ると，後者は変数条件を満たさないため書換規則でない．したがって，仕様実行時には無視される．同様に `0 = N * 0` 等も仕様実行時には無視される．

CafeOBJ システムは，等式をそのまま左から右への書換規則と見なす．どちらを左辺にすべきかという判断は記述するユーザが行う必要がある．本稿で TRS の基礎を知ることで，どのような仕様が簡約コマンドを用いた仕様実行および検証に適しているかの理解が深まることが期待される．

**書換関係：** TRS $R$ による書換は，書換規則 $l \to r \in R$ に対し，左辺 $l$ のインスタンスを部分項として持つ項を，そのインスタンスを対応する右辺 $r$ のインスタンスで置き換えることで行われる．項 $t$ から項 $t'$ への書換を $t \to_R t'$ と記述し，二項関係 $\to_R$ を書換関係と呼ぶ．矢印表記の二項関係 $\to \subseteq A \times A$ に対して，以下の表記を用いる．

$$
\begin{aligned}
a \to \circ \to b &= \exists c \in A. a \to c \wedge c \to b \\
a \to^0 b &= a = b \\
a \to^{i+1} b &= a \to \circ \to^i b \\
a \to^+ b &= \exists i \in \mathcal{N}_+. a \to^i b \\
a \to^* b &= \exists i \in \mathcal{N}. a \to^i b \\
a \downarrow b &= \exists c \in A. a \to^* c, b \to^* c
\end{aligned}
$$

$i$ は自然数，$\mathcal{N}$ は自然数の集合，$\mathcal{N}_+$ は正数の集合とする．自然数 $i$ に対し，$t \to^i t'$ は $i$ ステップでの書換を意味する．$t \to^+ t'$ および $t \to^* t'$ は 1 回以上および 0 回以上 (自身を含む) の書換を意味する．$t \downarrow t'$ は，両辺が同じ項に書き換えられることを意味する．

**項の簡約：** ある書換規則の左辺のインスタンスをリデックスと呼ぶ．それ以上書き換えられない項を正規形と呼び，その集合を $NF_R$ と書く ($NF_R = \{u \mid \neg\exists t. u \to_R t\}$ )．リデックスを含まない項の集合と $NF_R$ は一致する．項 $t$ に対し，$t \to_R^* t'$ かつ $t' \in NF_R$ となる $t'$ を求めることを，項の簡約という．以下の命題が成り立つ．

**命題 3.1** 仕様 $SP$，項 $t, t' \in T(\Sigma_{SP}, X)$ に対して，$t \to_{SP} t'$ (または $t \to_{SP}^* t'$ または $t \downarrow_{SP} t'$ ) ならば，$t =_{SP} t'$，すなわち $t = t'$ を $SP$ の公理から導くことができる

**簡約コマンドとトレース：** CafeOBJ システムの簡約コマンド `red` は TRS による項の簡約を行う．トレースをオンにすることで，詳細な書き換えの過程を見ることができる．

```
NAT+> set trace on
NAT+> red 0 + s 0 .
-- reduce in NAT+ : 0 + s 0
1>[1] rule: eq N:Nat + s M:Nat = s (N + M)
    { N:Nat |-> 0, M:Nat |-> 0 }
1<[1] 0 + s 0 --> s (0 + 0)
1>[2] rule: eq N:Nat + 0 = N
    { N:Nat |-> 0 }
1<[2] 0 + 0 --> 0
s 0 : NzNat
```

まず `1>[1]` で書き換えに用いる書換規則 (等式) と変数への代入を示している．この書換規則と代入により書換関係 `1<[1]` が成り立つ．同様に `1>[2]` の書換規則と代入のもとで `1<[2]` の書換関係が成り立つため，項 `s(0 + 0)` の部分項 `0 + 0` が `0` に書き換えられ，全体として項 `s 0` に簡約される．命題 3.1 より `0 + s 0` $=_{\texttt{NAT+}}$ `s 0` が保証される．またモデルにおいても `0 + s 0` と `s 0` は同じ要素に解釈される．実際，NAT+のモデルである自然数代数 $N$ において $N_{0+\mathrm{s}(0)} = N_+(N_0, N_\mathrm{s}(N_0)) = 0 + (0 + 1) = 1$. $N_{\mathrm{s}\,0} = N_\mathrm{s}(N_0) = (0 + 1) = 1$ が成り立つ．

**等価述語と合流判定：** 簡約コマンド `red` は，基本的には引数を先に簡約する最左最内戦略 (leftmost-innermost strategy, eager evaluation) で項を簡約する．組込モジュール `EQL` の等価述語 `_=_` に関する等式 `eq (X = X) = true` を考える．モジュール $SP$ + `EQL` において，項 $t$ = $t'$ の簡約は，まず両辺 $t$, $t'$ を簡約し，次に等式 `eq (X = X) = true` の適用を行う．両辺の簡約結果が等しいときに全体が `true` に簡約され

る．これは，項の合流判定 $(t \downarrow t')$ を実現している．

```
NAT+ + EQL> red 0 + s 0 = s 0 + 0 .
-- reduce in NAT+ + EQL : 0 + s 0 = s 0 + 0
true : Bool
```

出力 true は，0 + s 0 ↓ s 0 + 0 が成り立つことを保証する．よって命題 3.1 より 0 + s 0 $=_{\text{NAT+}}$ s 0 + 0 が成り立つ．

**停止性**：無限書換列 $t_0 \to_R t_1 \to_R t_2 \to_R \cdots$ が存在しないとき，TRS $R$ は停止性を持つ (terminating) という．本稿では，TRS(あるいは仕様) は有限であると仮定する．すなわち書換規則 (等式) の数を有限とする．すると停止性は，ある項から書き換えで得られる項の数が有限であることを保証する．よって (書き換えの戦略によらず) 項の簡約が有限時間内に終了することを保証する．項の正規形を解と考えると，解の存在を保証する性質でもある．

**合流性**：項 $t$ からの任意の分岐 $t \to_R^* t_1$，$t \to_R^* t_2$ に対して，$t_1 \downarrow_R t_2$ が成り立つとき，TRS $R$ は合流性を持つ (confluent) という．合流性は，解の一意性を保証する．よって停止性と合流性を持つ TRS は任意の項に対して唯一の正規形を持つ．TRS による等式推論を効果的に用いるためには，対応する TRS が停止性，合流性を持つように仕様を作成することが望ましい．次の命題が成り立つ．

**命題 3.2** TRS $R$ が停止性かつ合流性を持つとき，$t =_R t' \Leftrightarrow t \downarrow_R t'$.

仕様 $SP$ が停止性かつ合流性を持つとき，$SP$ + EQL における $t$ = $t'$ が true を返さないならば，両辺の等価性が公理から導けないことを保証する．

```
NAT+ + EQL> red 0 + s(0) = 0 + 0 .
-- reduce in NAT+ + EQL : 0 + s 0 = 0 + 0
s 0 = 0 : Bool
```

NAT+は停止性かつ合流性を持つため，0 + s 0 $\neq_{\text{NAT+}}$ 0 + 0 が成り立つ．NAT+はきつい意味で宣言されているため，NAT+のモデルにおいてこの両辺が等しくならないことが保証される．

**十分完全性**：仕様の意味を考えるとき，演算子は 2 種類に分けられる．モデルにおいて要素を構築する**構成子**と関数に解釈される**関数演算子**である．NAT+では，0，s_が構成子で，_+_が関数演算子となる．本稿では，慣習としてモジュール BASIC-$M$ で宣言された演算子を構成子と見なし，それを保護輸入することで関数演算子を定義する．保護輸入の性質から，モジュール BASIC-$M$ で宣言されたソートの台集合は，輸入先のモジュールにおいても構成子から構成される項 (構成子項) で尽くされることが保証される．すべての関数演算子がすべての構成子項に対して定義されているとき，仕様は十分完全である (sufficiently complete) という．正確には，任意の関数演算子 $f \in \Sigma_{s_1,\ldots,s_n,s}$，構成子項 $t_i \in T_{s_i} (1 \leq i \leq n)$ に対し，$f(t_1,\ldots,t_n) =_E t$ となる構成子項 $t \in T_s$ が存在するとき，十分完全である．十分完全な仕様は，仕様自身および輸入される仕様すべてがきつい意味を持つとき，その仕様のモデルが (要素の名前の違いを無視すれば) 一意に定まることを保証する．NAT+は十分完全であり，_+_を加算に解釈する代数のみがモデルとなる．

## 4 停止性

TRS が停止性を持つかどうかは決定不能である．すなわち任意の TRS $R$ に対して，$R$ が停止性を持つか持たないかを判定するアルゴリズムは存在しない．そのため，停止性の十分条件が数多く研究されており，それらをもとにした判定手法，判定ツールが提案，開発されている．

### 4.1 再帰経路順序

関数演算子の再帰構造に着目した項上の順序関係に再帰経路順序 (RPO) がある．RPO は，$\Sigma$ 上の擬順序 $\trianglerighteq \subset \Sigma \times \Sigma$ に対して定義される[†4]．

**定義 4.1** 演算子の集合 $\Sigma$，変数記号の集合 $X$，演算子上の半順序 $\trianglerighteq \subset \Sigma \times \Sigma$ に対して，RPO $>_{rpo} \subset T(\Sigma,X) \times T(\Sigma,X)$ は以下で定義される．

$t >_{rpo} t' \overset{def}{\Longleftrightarrow}$

$t = f(t_1,\ldots,t_m)$ かつ

1. ある $i \in \{1,\ldots,m\}$ に対して $t_i \geq_{rpo} t'$，または
2. $t' = g(t'_1,\ldots,t'_n)$ かつ $f \rhd g$ かつ任意の

†4 集合 $A$ 上の擬順序 $\leq \subset A \times A$ とは，反射律 $(\forall a \in A.a \leq a)$，推移律 $(\forall a,b,c \in A.a \leq b \wedge b \leq c \Rightarrow a \leq c)$ を満たす二項関係である．

$j \in \{1,\ldots,n\}$ に対して $t >_{rpo} t'_j$, または

3. $t' = g(t'_1,\ldots,t'_n)$ かつ $f \sim g$ かつ $\{\!\{t_1,\ldots,t_m\}\!\} >^{mul}_{rpo} \{\!\{t'_1,\ldots,t'_n\}\!\}$.

ここで，$a \rhd b$ は $a \geq b$ かつ $a \neq b$，$a \sim b$ は $a \geq b$ かつ $b \geq a$ の略記である．$\{\!\{\ldots\}\!\}$ は多重集合 (multiset) を表す†5．$>^{mul}$ は $>$ の有限多重集合拡張である†6．

次の性質が成り立つ．

**命題 4.2** TRS $R$ に対し，ある $\geq$ が存在し，任意の $l \to r \in R$ が $l >_{rpo} r$ を満たすならば，$R$ は停止性を持つ．

TRS NAT+ の演算子上の半順序を $+ \rhd s \rhd 0$ と定義すると，`N + 0` $>_{rpo}$ `N` および `M + s(N)` $>_{rpo}$ `s(M + N)` が成り立つ．前者は，`N` $\geq_{rpo}$ `N` から自明 (定義 4.1 の 1)．後者は，まず `s(N)` $>_{rpo}$ `N` が成り立ち (同 1)，$\{\!\{$`M, s(N)`$\}\!\} >^{mul}_{rpo} \{\!\{$`M, N`$\}\!\}$ より，`M + s(N)` $>_{rpo}$ `M + N` が成り立つ (同 3)．よって，$+ \rhd s$ より `M + s(N)` $>_{rpo}$ `s(M + N)` が成り立つ (同 2)．

### 4.2 停止性を持つ仕様の作成法

RPO で停止性を示すことができる仕様の作成法を簡単に解説する．まずモジュール BASIC-$M$ で構成子を定義する．次に関数演算子 $\mathtt{f}_0$ を BASIC-$M$ を保護輸入するモジュール $M\mathtt{f}_0$ で定義する．以降，$\mathtt{f}_i$ を使って定義する関数演算子 $\mathtt{f}_{i+1}$ を $M\mathtt{f}_i$ を保護輸入するモジュール $M\mathtt{f}_{i+1}$ で定義していく．例えば，BASIC-NAT を保護輸入した NAT+で加算演算を定義し，NAT+を保護輸入した NAT*で乗算演算を_+_を使って定義する．このとき，各関数演算子 $\mathtt{f}_i$ を定義するモジュール $M\mathtt{f}_i$ の公理 (等式の集合) $R_{\mathtt{f}_i}$ を以下のように与える．

```
eq f_i(t_11, t_12,...,t_1m) = t_1 .
⋮
eq f_i(t_n1, t_n2,...,t_nm) = t_n .
```

各 $t_{jk}$ は構成子と変数のみから構成される項である．各右辺 $t_j$ は変数と構成子と $k \leq i$ を満たす関数演算子 $\mathtt{f}_k$ からなる．ただし，$\mathtt{f}_i$ の出現は再帰呼び出しの形に限定される．すなわち，$t_j$ の任意の $\mathtt{f}_i$ の出現 $\mathtt{f}_i(u_{j1},\ldots,u_{jm})$ に対して各 $u_{jk}$ は $t_{jk}$ の部分項 (自身を含む) であり，少なくとも 1 つの $u_{jk}$ は $t_{jk}$ の真部分項 (自身を含まない) である．このとき，すべての構成子 $c$ に対して $\mathtt{f}_0 \rhd c$ かつすべての $i \geq k > j$ に対して $\mathtt{f}_k \rhd \mathtt{f}_j$ で演算子上の半順序 $\geq$ を与えると，RPO により $\cup^i_{k=0} R_{\mathtt{f}_k}$ の停止性を示すことができる†7．実際，定義 4.1 の 1 より，一般に項 $t$ とその真部分項 $u$ は $t >_{rpo} u$ の関係にあるため，定義 4.1 の 3 より，$\mathtt{f}_i(t_{j1},t_{j2},\ldots,t_{jm}) >_{rpo} \mathtt{f}_i(u_{j1},t_{j2},\ldots,u_{jm})$ が成り立つ．再帰呼び出し $\mathtt{f}_i(u_{j1},t_{j2},\ldots,u_{jm})$ に構成子 $c$ および $i > j$ を満たす $\mathtt{f}_j$ を何度適用しても $\mathtt{f}_i \rhd c$，$\mathtt{f}_i \rhd \mathtt{f}_j$ および定義 4.1 の 2 より，順序関係は保存されるため，$\mathtt{f}_i(t_{j1},t_{j2},\ldots,t_{jm}) >_{rpo} t_j$ が成り立つ．

仕様 NAT+および以下の仕様 NAT*，NAT-fact は上記の作成法にしたがっている．

```
mod! NAT*{
  pr(NAT+)
  op _*_ : Nat Nat -> Nat
  vars M N : Nat
  eq N * 0 = 0 .
  eq M * s(N) = (M * N) + M .
}
mod! NAT-fact{
  pr(NAT*)
  op fact : Nat -> Nat
  var N : Nat
  eq fact(0) = s(0) .
```

†5 多重集合とは，要素の重なりを許したものの集まりである．例えば，$\{\!\{a,a,b\}\!\}$ と $\{\!\{a,b\}\!\}$ は異なる多重集合である．集合 $A$ の要素を要素とする有限多重集合の集合を $FM(A)$ と書く．

†6 半順序 $>\subseteq A \times A$ に対し $>^{mul} \subseteq FM(A) \times FM(A)$ は以下で定義される：$M_1 >^{mul} M2 \Leftrightarrow \exists X,Y \in FM(A).[X \neq \emptyset \wedge X \subseteq M_1 \wedge M_2 = (M_1 \setminus X) + Y \wedge \forall y \in Y.\exists x \in X.x > y]$．例えば ( $X = \{\!\{2\}\!\}$, $Y = \{\!\{1,1\}\!\}$ で) $\{\!\{2,2,3\}\!\} >^{mul} \{\!\{1,1,2,3\}\!\}$ が成り立つ．

†7 輸入を含む仕様の TRS は，自身および輸入されているすべてのモジュールの公理の和集合で与えられるため，停止性もその和集合に対して示さなければならない．

```
  eq fact(s(N)) = s(N) * fact(N) .
}
```

### 4.3 その他の判定手法

RPO で判定できる停止性は，単純停止性というクラスに分類される．停止性を持つが単純停止性は持たない TRS の停止性判定には，より高度な判定手法を用いる必要がある．近年では，書換規則の左辺のルートに現れる演算子に注目した依存対 (dependency pair) と呼ばれる概念を用いた判定手法が最も強力である [4]．また自動判定ツールの分野では，2004 年より毎年 TRS の停止性自動判定ツールの競技会 (Termination Competition †8) が開催されており，2004 年，2005 年，2006 年と最優秀ツールとして評価されている AProVE †9 を筆頭に数多くのツールが開発されている．RPO で停止性を示せないときは，これらのツールを用いて停止性を示すことが考えられる．

### 4.4 仕様作成の注意

本節で解説した仕様作成法は，データ型のモジュール BASIC-$M$，関数のモジュール $M$-$f$ という階層構造を持っているが，関数の階層性に注意を払っていれば，特にモジュール構造で対応させる必要はない．実際，次回以降に登場する例では，データ型と関数を分けずに 1 つのモジュールで与えることがある．

## 5 合流性

TRS が合流性を持つかどうかは決定不能である．ただし，停止性を持つ TRS に対しては決定可能である．停止性を持たない TRS に対する合流性の判定条件は数多く提案されているが，停止性に比べてツールなどは充実していない．

**リデックスの重なり：** 合流性は，ある項から書き換えられた 2 つの項は必ず同じ項に書き換えられるという性質である．NAT+の項 $t$ =(0 + 0) + (0 + 0) を考える．$t$ から第 1 引数，第 2 引数をそれぞれ書き換えた $t_1$ =0 + (0 + 0) および $t_2$ =(0 + 0) + 0 への分岐がある．このとき，それぞれ書き換えていない方の引数を書き換えることで $t_3$ =0 + 0 に合流できる．このように重なりのないリデックスを書き換えて得られた項は，お互いのリデックスを書き換えることで合流可能である．次の仕様を考える．

```
mod! NAT*'{
 pr(NAT*)
 var N : Nat
 eq N + (N + N) = N * s s s 0 .
}
```

NAT*' では，項 0 + (0 + 0) から 0 + 0 および 0 * s s s 0 への分岐がある．前者は等式 eq N + 0 = N を部分項 0 + 0 へ適用した書き換え，後者は NAT*' の等式の項全体への適用である．それぞれの書き換え後の項では，お互いのリデックスがなくなっている．このような重なりのあるリデックスから生じた分岐は，一般には合流するとは限らない．

**危険対：** 合流性判定では，重なりのあるリデックスから生じる分岐が重要となる．そのような分岐を解析する目的で提案されたのが危険対 (critical pair) の概念である．

**定義 5.1** 変数を共有しない項 $t, t' \in T(\Sigma, X)$ に対して，$\theta(t) = \theta(t')$ となる代入を単一化子 (unifier) といい，単一化子を持つ対 $t, t'$ を単一化可能という．また，任意の単一化子 $\sigma$ に対し，ある代入 $\tau$ が存在して，$\forall x \in X.\tau(\theta(x)) = \sigma(x)$ が成り立つとき，$\theta$ を最汎単一化子 (most general unifier) という．

項 s M + N，項 N' + N' に対し，$\sigma$ : {M → 0, N → s 0, N' → s 0} および $\theta$ : {M → X, N → s X, N → s X} はそれぞれ単一化子である．$\sigma$ は最汎単一化子ではない．$\theta$ は最汎単一化子である．代入 $\tau$ : X → 0 が存在して，$\forall x \in X.\tau(\theta(x)) = \sigma(x)$ が成り立つ．

特別な定数 □ を唯一つ含む項 $C \in T(\Sigma \cup \{\Box\}, X)$ を文脈と呼ぶ．$C$ 中の □ を項 $t$ に置き換えた項を $C[t]$ と書く．

**定義 5.2** TRS $R$ に対し，$l_1 \to r_1$，$l_2 \to r_2$ を互いに変数を共有しないように変数の名前変えをした $R$ の書換規則とする (同じ書換規則の名前変えでもよい)．項 $l_2$ と単一化可能な非変数項 $t \in T(\Sigma, X) \setminus X$ と文脈 $C$ に対し，$l_1 = C[t]$ が成り立つとする．このとき，

†8 http://www.lri.fr/~marche/termination-competition/

†9 http://aprove.informatik.rwth-aachen.de/

$l_2$ と $t$ の最汎単一化子 $\theta$ に対し，項の対 $(C[r_2]\theta, r_1\theta)$ を $R$ の危険対と呼ぶ．ただし $l_1 \to r_1$，$l_2 \to r_2$ が同じ書換規則からの名前変えの場合は，文脈 $C$ は□ではないと仮定する．

NAT*' は，危険対 ( 0 + 0 , 0 * s s s 0 ) および ( s N + s (s N + N) , s N * s s s 0 ) を持つ．それぞれ，項 0 + (0 + 0) および s N + (s N + s N) から得られる．以下の定理が成り立つ．

**命題 5.3** TRS $R$ が停止性を持つとき，以下は同値：(1) $R$ は合流性を持つ．(2) $R$ のすべての危険対は合流する．

停止性を仮定することにより，ある項から書き換えでたどり着けるすべての項を有限時間内に求めることが可能である．よって危険対が合流するかどうかは決定可能であり，停止性を持つ TRS の合流性判定は決定可能である．上記の NAT*' の危険対は，前者は 0 に合流するが後者は合流しない．よって NAT*' は合流性を持たない．

### 5.1 合流性を持つ仕様の作成法

停止性を仮定すれば，命題 5.3 より，すべての危険対が合流するように仕様を作成すればよい．まず，4.2 節にしたがって停止性を持つ仕様を作成する．各等式の左辺のルートシンボルは関数演算子であり，ルート以外に関数演算子は現れない．したがって左辺の重なりの検査は，$R_{\mathtt{f}_i}$ ごとに異なる等式の左辺のルート同士を比べるだけで十分である．すなわち，任意の異なる $j,k$ に対し，等式 eq $\mathtt{f}_i(t_{j1}, \ldots, t_{jm}) = t_j$ と eq $\mathtt{f}_i(t_{k1}, \ldots, t_{km}) = t_k$ について左辺のルートでの重なりを調べる．重なりがあれば危険対を求め，合流することを確かめる．すべての危険対が合流すれば，$\cup_{k=0}^{i} R_{\mathtt{f}_k}$ は合流性を持つ．

**例題 5.4** 仕様 NAT*の公理に計算の効率化のため，左零元の等式 eq 0 * N = 0 を加えた仕様 NAT*' を考える．

```
mod! NAT*'{ ...
  eq N * 0 = 0 .
  eq 0 * N = 0 .
  eq M * s(N) = (M * N) + M .
}
```

すると右零元と左零元の等式がルートで重なりを持つ．その危険対は，(0,0) であり，明らかに合流する．よって NAT*' は合流性を持つ．

### 5.2 その他の判定手法

停止性を仮定しない合流性の十分条件に以下がある．

**命題 5.5** 重なりがなく左線形な TRS は合流性を持つ．

すべての変数が 1 回のみしか出現しない項を線形であるといい，任意の書換規則の左辺が線形である TRS を左線形という．仕様 NAT+, NAT*, NAT-fact はこの条件を満たすため，(停止性の証明なしでも) 合流性を持つことが保証される．

## 6 十分完全性

十分完全性もまた決定不能問題であるが，停止性を仮定することで決定可能な判定法が得られる．停止性を仮定すると，すべての項は正規形を持つことが保証される．よってすべての正規形が関数演算子を含まなければ十分完全性が成り立つ．

公理 $R_{\mathtt{f}_i}$ が 4.2 節の条件および以下の条件を満たすとき，$\cup_{k=0}^{i} R_{\mathtt{f}_k}$ は十分完全性を持つ．

- 任意の基底項の組 $(t'_1, \ldots, t'_m) \in T(C,\emptyset)_{s_1} \times \cdots \times T(C,\emptyset)_{s_m}$ に対して，それをインスタンスとする左辺のパターン $(t_{j1}, t_{j2}, \ldots, t_{jm})$ が少なくとも 1 つ存在する．

条件より，もし基底項に $\mathtt{f}_i$ が出現するならば $\mathtt{f}_i(\ldots)$ はいずれかの左辺のインスタンスであるため正規形でない．よって正規形は構成子項である．仕様 NAT+, NAT*, NAT-fact は十分完全性を持つ．

## 7 CafeOBJ の書換エンジン

ここからは，CafeOBJ の仕様作成・検証のための各種機能について TRS の観点から議論する．各種機能の詳細は，チュートリアル第二回を参照されたい．

### 7.1 構造化モジュール

**輸入・パラメータ：** 前節で触れた通り，輸入を含む仕様の TRS は，すべての輸入関係にあるモジュール (および仮引数) の公理の和集合で与えられる．一般に，個々の公理の停止性，合流性などは，全体として

TRSの各性質を保証しない．輸入するモジュールの公理も考慮した仕様作成が必要である．

**パラメータ化モジュール：** パラメータ化モジュールに関する一般的な性質を検証する場合は，仮引数のまま仕様を実行させることになるため，単に仮引数を輸入した仕様と見なしてよい．一方，具現化された仕様は等式が大きく変化する．具現化仕様の実行では，仮引数の等式はなくなり，実引数の等式およびパラメータ化モジュールの等式をビューの対応表で置き換えた等式が書換規則として用いられる．具現化仕様の停止性，合流性，十分完全性を判定する際には，モジュールを表示するshowコマンドが有効である．具現化仕様へのshowコマンドの適用により，具現化仕様で有効な等式を確認できる．

### 7.2 組込モジュール

組込モジュールのいくつかは，公理を含まないため直接対応するTRSがない．CafeOBJシステムでは，そのような組込モジュールの関数演算子はCommon Lispによって定義(実装)されている．簡約コマンドでは，関数演算子 $f$ に対して $f(\vec{t})$ に対応する値がCommon Lispによって計算される．例えば，組込モジュールINTでは，red 1 + 2は，Common Lisp式(+ 1 2)の評価値である3を返す．対応するTRSとしては，すべての $f(\vec{t})$ とその返り値 $u$ に対して書換規則 $f(\vec{t}) \to u$ を含むTRSが考えられる．多くの組込モジュールに対して，このようなTRSは無限個の書換規則を含むことになる．

### 7.3 AC演算子

二項演算子の属性commは停止性を保つのに重要である．もしcommに相当する等式eq X Y = Y Xを公理に加えると明らかに停止性を満たさない．属性assocは，合流性に影響がある．結合律を直に公理に加えた例を考える．

```
mod! HEAD-TAIL(D :: TRIV){
  [Elt.D < Str]
  op _ _ : Str Str -> Str
  op head : Str -> Elt.D
  op tail : Str -> Elt.D
  vars X Y Z : Elt.D
  var S : Str
  eq (X Y) Z = X (Y Z) .
  eq head(X S) = X .
  eq tail(S X) = X .
}
```

関数head，tailはそれぞれ(要素が2以上の)列の最初および最後の要素を返す関数である．例えば，head((X Y) Z)は，1つ目の等式(結合律)よりhead(X (Y Z))と等しく，2つ目の等式よりXを返す．項tail((X Y) Z)を考える．3つ目の等式よりZに書き換えられる．一方，1つ目の等式よりtail(X (Y Z))にも書き換えられ，どちらも正規形である．よって合流性を満たさない．結合律の向きを逆のX (Y Z) = (X Y) Zとしてもheadに対して同様の現象が起きる．両方向の結合律を含めると停止性を満たさない．このように交換律，結合律を直に等式で定義すると対応するTRSの停止性，合流性の成立に影響を及ぼす．このような問題を解決すべく，これらを演算子の属性として記述することが推奨される．

**AC項書換システム：** assoc，comm演算子に対応するTRSに，Associative-Commutative TRS (AC-TRS)がある．一般には，等式の集合 $E$ を法としたTRS ($E$-TRS)の一種である．$E$-TRSは，書換規則の集合 $R$ と等式の集合 $E$ から成る．書換規則 $\to_{R/E}$ は，$t_1 =_E t_1'$ となる $t_1'$ が存在し，$t_1' \to_R t_2$ が成り立つとき，$t_1 \to_{R/E} t_2$ と定義される．A-TRS，C-TRS，または，AC-TRSは，$E$ としてそれぞれ結合律，交換律，または両者を取る $E$-TRSである．一般の $E$-TRSに対しては，$E$ 等価性の判定が必要なため実装は容易ではない．一方，AC書換規則は両辺で項の大きさが変わらず，変数の出現も同じであるため，ある項とAC等価な項の数は有限個である．CafeOBJにおけるAC-TRSの実装では，まず，与えられた項とAC等価な項を探索し，書換規則が適用できる項が見つかったら書き換える．有限性より1回のAC書き換えの計算が無限ループに陥ることはない．

AC属性を持った構成子上の関数を定義する際は，どのような項がAC等価となるかに注意する必要がある．例えば，上記のheadの定義は結合律のみを持つ構

成子には適切であるが，もし op __ : Str Str -> Str {assoc comm} で定義されていた場合，X Y $=_{AC}$ Y X より，head(X Y) $\rightarrow_{R/AC}$ X かつ head(X Y) $\rightarrow_{R/AC}$ Y となり，合流性を満たさない．

### 7.4 条件付き等式

書換規則に条件を付加可能な TRS として条件付き TRS(CTRS) がある．CTRS は条件付き書換規則の集合である．条件付き書換規則は，項 $l, r, t_1, \ldots, t_n, t'_1, \ldots, t'_n$ に対し $l \rightarrow r \Leftarrow t_1 = t'_1, \ldots, t_n = t'_n$ の形をしている．CTRS には変数条件の違いからいくつかの種類が存在する．また CTRS の書換関係も条件部の評価方法の違いから数種類存在する．詳しくは，文献[1][4]を参照．

**CafeOBJ の条件付き等式：** CafeOBJ の条件付き等式の構文は，通常の CTRS とは少し異なる．CafeOBJ 仕様における条件付き等式 ceq $l$ = $r$ if $c$ の条件部 $c$ には，任意の Bool ソートの項を記述できる．CafeOBJ では，すべての仕様は，ブール代数の組込モジュール BOOL を暗黙に輸入しているため，このような条件部の記述が可能となる．CafeOBJ の書換エンジンが条件付き等式を条件付き書換規則として認識するための条件は，すべての右辺および条件部に現れる変数が左辺にも現れることである．条件付き等式 ceq $l$ = $r$ if $c$ の適用は次のように行われる．項 $t$ が等式の左辺のインスタンス $l\theta$ を部分項として持ち，条件部のインスタンス $c\theta$ が true に簡約されるとき，$t$ の部分項 $l\theta$ を $r\theta$ に置き換える[†10]．

**停止性・合流性・十分完全性：** 条件付き等式を含む仕様に対して書換エンジンが適切に振舞うためには右辺だけでなく条件部にも注意を払う必要がある．書換エンジンの停止性は，以下の変換で得られる仕様 $SP'$ の停止性で保証される．仕様 $SP$ 中のすべてのソート $s$ に対して，演算子 op if_then : Bool $s$ -> $s$ を加える．すべての条件付き等式 ceq $l$ = $r$ if $c$ を eq $l$ = if_then($c$, $r$) に置き換える．等式 eq if_then(true, X) = X を加える．合流性判定は，左辺の重なりに合わせて条件部にも注意を向ける必要がある．十分完全性についても同様に条件部がすべての場合を尽くしていることを検査する必要がある．

**条件付き等式の例・階乗関数 FACT：** 条件付き等式を用いた階乗関数の仕様 FACT を考える．ただし，NAT' は各種演算子 (*, p_, _=_ など) が定義済みの自然数の仕様とする．

```
mod! FACT{
 pr(NAT')
 op fact : Nat -> Nat
 var N : Nat
 ceq fact(N) = N * fact(p N) if not N = 0 .
 ceq fact(N) = s 0 if N = 0 .
}
```

停止性判定のためには，条件部も 4.2 節の右辺が満たすべき条件を満たす必要がある．もし条件部に fact(N) が出現したと仮定すると，fact(0) の書き換えで，条件部内の fact(0) が呼び出され無限ループに陥る可能性があるため，条件部も簡約の停止性に大きな影響を持つ．条件部 not(N = 0) および N = 0 は，fact を含まず，NAT' で定義済みの演算子から構成されているため，停止性を崩さない．

合流性判定を考える．危険対の定義をそのままあてはめると危険対 (N * fact(p(N)),s(0)) が存在する．しかし実際には，条件部が互いに素であるため，重なり fact(N) のインスタンス fact($t$) から分岐が生じることはない．このように重なりがある条件付き書換規則同士は，条件部が互いに素となるように記述すれば，合流性判定のためにその危険対を検査する必要がなくなる．

条件部の和が true に等しいため，FACT は十分完全性を満たす．合流性かつ十分完全性のためには，同じ左辺を持つ条件付き書換規則に対し，条件部 $c_1, c_2, \ldots, c_n$ が互いに素 ($j \neq k \Rightarrow c_j$ and $c_k$ = false) かつすべての場合をつくす ($c_1$ or $\cdots$ or $c_n$ = true) ように注意する．

### 7.5 観測遷移システム

振舞仕様には，隠蔽ソートというシステムの状態空

†10 条件付き等式を含む CafeOBJ の書換エンジンに対応する書換関係の定義は，通常の CTRS の書換関係と同様に，条件呼び出しの深さを $n$ に制限した書換関係 $\rightarrow_R^n$ を定義し，$\rightarrow_R = \bigcup_{n \in \mathcal{N}} \rightarrow_R^n$ で定義できる．詳細は省略する．

間を表すソートがある．隠蔽ソートは，その名の通り隠蔽されており，直接 (構成子などを用いて) 表現されない．特別な演算である観測関数と遷移関数によって状態の観測や更新が可能となる．観測遷移システム (OTS) は，遷移関数が観測等価を保存する振舞仕様であり，CafeOBJ による抽象機械の仕様作成・検証に有効な概念である．

**観測遷移システムの条件：** 振舞仕様が OTS であるためには，遷移関数が観測等価を保存する必要がある．すなわち，観測等価な状態 $s$，$s'$，遷移関数 $\tau$ に対し，状態 $\tau(s)$ と $\tau(s')$ は観測等価でなければならない．状態 $s$ と $s'$ の観測等価は，任意の観測関数 $o$ に対して $o(s) = o(s')$ が成り立つことで定義される．OTS 仕様では，遷移後の観測値がすべての遷移関数と観測関数の組み合わせについて定義されているかどうかが，十分完全性に対応する性質になる．さらに，初期状態 `init` に対して，すべての観測関数 $o$ で $o$(`init`) が定義されていれば，抽象機械における初期状態からの任意の実行列 $\tau_1, \tau 2, \ldots, \tau_n$ に対して，実行列を適用後の状態 $\tau_n(\cdots(\tau_1(\texttt{init})))$ の観測値が決定する．

**観測遷移システムの仕様作成：** 振舞仕様が OTS であるための文献[6] の条件をもとに OTS 仕様の作成指針を以下に示す．

- 遷移関数 (を表す振舞演算子) `bop` $\tau$ : $h\ s_1 \cdots s_n$ `->` $s$ に対し，遷移の事前条件を判定する述語およびそれを定義する等式を宣言する．

  `op c-`$\tau$ : $h\ s_1 \cdots s_n$ `-> Bool`

  `eq c-`$\tau$`(S, X1,..., Xn) =` $t$
- 初期状態 `op init ->` $h$ を宣言する．
- 観測関数 (を表す振舞演算子)$o$ に対して，初期状態の観測値を定義する等式を宣言する．

  `eq` $o$`(init, X1, ... , Xn) =` $t$
- 遷移関数 $\tau$ が事前条件を満たさないとき，$\tau$ は状態は変化させないことを表す等式を宣言する．

  `ceq` $\tau$`(S, X1, ... , Xn) = S`

  `if not c-`$\tau$`(S, X1,..., Xn)`
- 観測関数 $o$ および遷移関数 $\tau$ のすべての組み合わせについて，$\tau$ 後の $o$ による観測値を定義する等式を宣言する．

  `ceq` $o$`(`$\tau$`(S, X1,..., Xn), Y1,..., Ym) =` $t$

  `if c-`$\tau$`(S, X1,..., Xn)`

ただし，$h$ は隠蔽ソート，`S` は隠蔽ソートの変数，`X1,...,Xn`, `Y1,...,Ym` は (振舞演算子の可視ソート引数に対応する) 変数とする．観測関数の入れ子 ( $o(\texttt{S}, o'(\texttt{S}))$ など ) はないものとする．また各等式の右辺の項 $t$ には，遷移関数 $\tau$ は出現しないものとする．上記の指針で作成された振舞仕様は，OTS 仕様すなわち遷移関数が状態の観測等価性を保存する仕様となる．実際，最後の等式より，遷移の後の状態の観測値は遷移前の状態の観測値に依存して決まるため，観測等価な状態は同じ状態に遷移する．書き換えにより $\tau$ の数が減るため明らかに停止性を満たす．重なりがある条件付き等式は，条件部が互いに素になるように与えられているため，合流性も満たす．

チュートリアル第 1 回の OTS 仕様 `QLOCK` は，この作成指針にしたがって記述されている．第 2 回の OTS 仕様 `ARRAY` も `set` の事前条件が常に真であるため，`c-set` を省略してあるが，この作成指針にしたがって記述されている．

## 8 仕様検証

命題 $s = t \Rightarrow s' = t'$ に対する証明譜は，仮定 `eq` $s$ `=` $t$ を宣言し，結論 `red` $s'$ `=` $t'$ を確かめる．したがって，仕様実行時 (簡約コマンド `red` 実行時) における TRS は仮定の等式を含む．したがって，仕様が停止性，合流性を満たしていても，仕様検証時の TRS はそれらを満たさないことがある．例えば，変数 `vars M N : Nat` に対する等式 `eq M + N = N + M` は停止性を持たない．一方，定数 `ops m n : -> Nat` に対する等式 `eq m + n = n + m` は停止性を壊さない．交換律の検証では，帰納法の仮定として後者が宣言される．交換律を補題として用いて，別の性質を検証する際は，前者を宣言しなくてならないが，演算子属性 `comm` を用いることで問題は回避できる．

停止性，合流性を満たす証明譜の作成指針はない．また多くの実例においても，仕様検証時の TRS は停止性，合流性 (特に合流性) を満たさないことが多い．しかし，結果として `true` が得られれば，その証明の正しさは保証される．

## 9 項書換システムの実装

TRS を実装するには，与えられた項のどのリデックスを書き換えるかを指定する書換戦略やリデックスに適用可能な複数の書換規則から 1 つを選ぶ優先順位などが必要になる．CafeOBJ の実装では，まず評価戦略 (E 戦略, Evaluation Strategy) によるリデックスの決定，次に書換規則の優先順位の順で書き換えが行われる．

### 9.1 評価戦略

CafeOBJ の書換エンジンの書換戦略は E 戦略で与えられる [3]．E 戦略は，演算子の属性として与えられる局所戦略 (local strategy) にもとづき決定されるユーザ定義の書換戦略である．デフォルトの局所戦略は，基本的には最左最内戦略であるが，簡単な厳密性解析 (strictness analysis) によりいくつかの引数の評価を後回し (lazy) にしている．詳細は，CafeOBJ マニュアルを参照せよ †11．

### 9.2 書換規則の優先順位

書換規則の優先順位は，複数のモジュール間では新しいモジュールの等式が優先され，同じモジュール内では上に書かれた等式が優先される．open コマンドも名前のないモジュールの宣言と考える．したがって open コマンド後に宣言した等式が最優先で用いられる．以下にチュートリアル第 2 回での配列仕様 ARRAY の証明譜の一部を再掲する．

```
mod! EQL-INT{ ...
 eq (X = Y) = (X <= Y) and (Y <= X) .
}
mod* ARRAY{
 pr(EQL-INT + ...)
 ...
 eq val(set(S, X, A), Y) =
  if X = Y then A else val(S, Y) fi .
}
open ARRAY + EQL .
...
-- case of i = x, i =/= y
eq (i = x) = true .
eq (i = y) = false .
red val(set(set(s, x, a), y, b), i)
   = val(set(set(s, y, b), x, a), i) .
close
```

open 後の簡約コマンド実行時では，リデックス i = x に等式 eq (X = Y) = (X <= Y) and (Y <= X) と等式 eq (i = x) = true の両方が適用できるが，この証明譜では後者が用いられ，結果 true が返される．

もし，eq (i = x) = true の代わりに eq i = x と書いてしまうと (i が左辺，x が右辺の等式)，その証明譜は，if (y <= x and x <= y) then ... という項を返す．これは，項 i = y の簡約で，まず E 戦略にしたがいリデックス i が項 x に書き換えられ，次に項 x = y が項 x <= y and y <= x に書き換えられるためである．仕様検証時には，E 戦略，書換規則の優先順位を念頭において証明譜を記述する必要がある．

## 10 おわりに

CafeOBJ の仕様検証の基礎となる等式推論を効率的に行う項書換システムについて，特に CafeOBJ に関連する項目を中心に解説した．より詳しい内容は，参考文献を参照されたい．文献 [1] は停止性，合流性などの TRS の基礎的な内容を含んでいる．文献 [4] は条件付き TRS を含んだより専門的な内容である．文献 [5] は書換戦略や高階変数を持つ TRS についての内容も含んでいる．文献 [7] は，日本語での TRS の解説論文であり本稿の邦訳の参考にした．

本稿で示した仕様作成法は，必ずしもすべてのシステムに有効であるわけではない．特に，仕様検証においては，停止性，合流性を持たない TRS を扱うことが多い．そのような TRS に対する効果的な証明譜の作成は，ユーザの工夫が必要である †12．こうした事

†11 http://www.ldl.jaist.ac.jp/cafeobj/

†12 停止性や合流性を持たない書換規則を用いて仕様検証のための証明譜を記述する方法に付いては第 4,5,6 回のチュートリアルで解説する．

情もふくめ，TRS による簡約コマンドの振舞を理解することは，仕様や証明譜の作成のため重要である．

## 謝辞

有益なコメントをいただいた査読者の方々と編集委員の方々に感謝いたします．

## 参考文献

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

[2] Diaconescu, R. and Futatsugi, K.: *CafeOBJ report*, AMAST Series in Computing, Vol. 6, World Scientific, 1998.

[3] Futatsugi, K., Goguen, J. A., Jouannaud, J.-P. and Meseguer, J.: Principles of OBJ2, in *POPL '85: Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, New York, NY, USA, ACM Press, 1985, pp. 52–66.

[4] Ohlebusch, E.: *Advanced Topics in Term Rewriting*, Springer, 2002.

[5] Terese: *Term Rewriting Systems*, Cambridge Tracts in Theoretical Computer Science, Vol. 55, Cambridge University Press, 2003.

[6] 中村正樹, 緒方和博, 二木厚吉: 項書き換えシステムにおける可簡約演算子とその応用, 情報処理学会論文誌：プログラミング, Vol. 46, No. SIG 6(2005), pp. 47–59.

[7] 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(1983), pp. 133–146.