Vol. 49 No. SIG 0(PRO 68) 情報処理学会論文誌：プログラミング 2008

制約付き項書換え系の潜在帰納法を利用した手続き型プログラム検証の試み

古市祐樹[†1,†2] 西田直樹[†1] 酒井正彦[†1]
草刈圭一朗[†1] 坂部俊樹[†1]

手続き型プログラムの検証手法として，モデル検査やホーア論理に基づく検証手法が代表的である．一方，項書換えの分野では帰納的定理の証明手法として潜在帰納法や書換え帰納法などが広く研究されている．本論文では，帰納的定理の証明法を利用して手続き型プログラムの等価性の検証を試みる．具体的には，整数上の while 言語で定義された C 言語関数からプレスブルガー文を規則の制約として持つことが許された制約付き項書換え系への変換を与え，その変換により手続き型プログラムの等価性を書換え系の関数の等価性に帰着させ，潜在帰納法を用いて等価性検証を試みる．この手法を実現するために，合流性を保証するための危険対定理および完備化手続きを制約付き書換え上へ拡張する．

Approach to Procedural-Program Verification Based on Implicit Induction of Constrained Term Rewriting Systems

Yuki Furuichi [†1,†2] Naoki Nishida [†1] Masahiko Sakai [†1]
Keiichirou Kusakari [†1] and Toshiki Sakabe [†1]

In the field of procedural programming, several verification methods based on model checking or Hoare logic have been proposed. On the other hand, in the field of term rewriting, implicit induction and rewriting induction have been widely studied as methods for proving inductive theorems. In this paper, we try to take advantages of methods for proving inductive theorems in verifying procedural functions written in a subset of the C language with integer type. More precisely, we propose a transformation from procedural programs to constraint term rewriting systems whose constraints are in Presburger arithmetic, and show that the transformation reduces the equivalence of procedural programs to that of functions in rewrite systems. By verifying equivalence between rewrite systems, we verify that between the corresponding procedural functions. To establish this approach, we extend Critical Pair Theorem and the basic completion procedure for constraint systems.

1. はじめに

手続き型プログラムと関数型プログラムでは，それぞれにプログラム検証手法の研究がされている．手続き型プログラムに対しては，モデル検査[7),13),18)] やホーア論理に基づく検証手法[9),11),13)] が代表的である．しかし，ループ不変式の発見は原理的に不可能であることや，事前条件・事後条件の付与などのヒューリスティックな作業が必要であり，検証の自動化は困難である．一方，関数型プログラムに対しては，帰納的定理の自動証明手法である潜在帰納法[12),15)] や書換え帰納法[3),19)] などが項書換え系（TRS）上で広く研究されている．帰納的定理とは任意の基底項上で成立する等式である[6)]．関数の等価性は帰納的定理として定式化できるので，等価性検証に帰納的定理の自動証明法を利用できる．

本論文では，帰納的定理の証明法を利用し，手続き型プログラムの検証手法の枠組を提案する．基本アイディアは，手続き型プログラムを同等の計算を行う制約付き TRS に変換し，それが仕様として与えられる TRS と等価であることを帰納的定理の証明法を利用して検証するというものである（図 1）．なお，この手法は，2つの手続き型プログラムの等価性検証にも応用できる．このアイディアを実現するために，主に以下の3つのことを行う．

- 自然数上の while 言語で定義された C 言語関数からプレスブルガー文を規則の制約として持つこ

†1 名古屋大学大学院情報科学研究科
Graduate School of Information Science
†2 2007 年 3 月修了

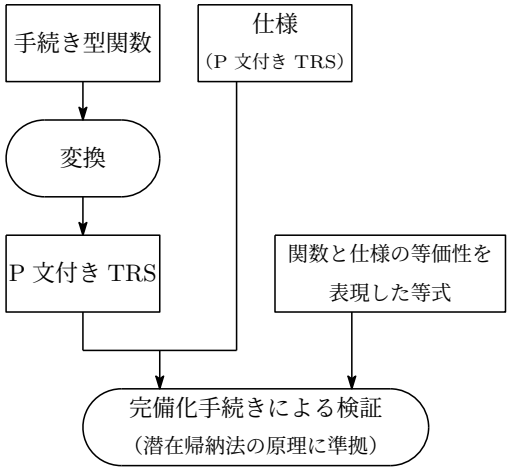


図 1 提案する検証手法の概要.
Fig. 1 Framework of a verification proposed in this paper.

とが許された制約付き TRS への変換を提案する．

- 合流性を保証するための危険対定理を制約付き書換え上へ拡張する．
- 完備化手続きを制約付き書換え上へ拡張する．

基本アイディアを具体化する際，通常の TRS では潜在帰納法に基づく自動証明[3),12),14)] に必要な完備化手続きに問題が生じる．完備化手続きは，与えられた TRS に対して停止性・合流性を持つ等価な TRS を構成する手続きであり，その手続きは必ず停止するとは限らない．例えば，比較演算子を含む項が完備化手続きを暴走させることも少なくない．これは完備化手続きそのものに潜在している問題が原因であり，本節の末尾で具体例（関数 max）を挙げて説明する．そこで，TRS を拡張し，真偽が決定可能であることが知られているプレスブルガー文を制約として付加した書換え規則からなる書換え系（P 文付き TRS）を導入する．それに伴い，書換え規則の制約部を考慮して推論規則を拡張することにより，完備化手続きを P 文付き TRS の枠組に拡張する．この拡張によって比較演算が項中に出現しなくなるので，通常の TRS の場合の完備化手続きの問題を解消する（関数 max の例を参照）．P 文付き TRS の枠組での完備化手続きについては，決定可能な一階述語論理式を制約に付加した TRS（制約付き TRS）とそれに対する完備化手続きに一般化して議論する．また，完備化で得られる書換え系の局所合流性を保証するための危険対定理についても制約付き TRS 上へ拡張する．制約付き TRS の制約部では，条件付き TRS とは異なり，その真偽が書換え関係には関係なく決定される．よって，本手法は条件付き TRS の枠組での完備化[10)] よりは TRS の完備化に近い．

定理自動証明の枠組の多くは，等式集合や規則集合に適用するいくつかの推論規則とそれらの適用戦略で構成される[3),5),12),15),19)]．本論文では，潜在帰納法の原理に準拠した完備化手続きに基づいた検証法の枠組を制約付き TRS 上へ拡張する．この枠組を基本にする理由は，完備化手続きが TRS 上ですでに非常によく研究され，基本的な理論として確立されていること，他の多くの枠組が持つ基本的な推論規則で構成されていることである．よって，将来的な拡張の際にも，これまでの多くの成果の利用が期待できる．また，本論文で提案する枠組が他の枠組を拡張する際にも非常に有効である．

検証手法は次の通りである．まず，手続き型プログラムを等価な P 文付き TRS に変換する．対象とする手続き型プログラムは，制御構造が while 文と if 文，データ型が int 型（本論文では特に自然数のみ扱う）に制限された C 言語で記述された関数である．ただし，条件文については bool 型の解釈に従う．状態遷移により意味論を定式化した上で，プログラムを忠実に模倣する P 文付き TRS を構成する．次に，手続き型プログラムから変換により得られた P 文付き TRS と仕様としてあらかじめ与える P 文付き TRS に対して，その両者が等価であることを帰納的定理の証明手法（本論文では，潜在帰納法の原理に基づいた完備化手続き）を適用して検証する．実際に，トイプログラムではあるが，自然数列の総和を求めるプログラムに対して検証が成功した例を紹介する．

最後に，通常の TRS ではなく，P 文付き TRS を導入する理由を説明する．手続き型プログラムから TRS へ直接変換する手法も試みたが，比較演算を表す項の存在が完備化を安易に暴走させることを実験で確認した．暴走の原因は完備化手続きにあり，関数型プログラムでの定理自動証明でも，判定条件に比較演算が用いられる if 文を含むようなプログラムでは頻繁に生じる暴走である．例えば，次の自然数上の最大値を求める TRS を考える．

$$\begin{cases} \mathsf{max}(x,y) \to \mathsf{if}(x < y, y, x) \\ \mathsf{if}(\mathtt{true}, x, y) \to x \\ \mathsf{if}(\mathtt{false}, x, y) \to y \\ 0 < \mathsf{s}(y) \to \mathtt{true} \\ x < 0 \to \mathtt{false} \\ \mathsf{s}(x) < \mathsf{s}(y) \to x < y \end{cases}$$

この TRS 上で等式 $\mathsf{max}(x,x) \approx x$ が帰納的定理であることを通常の KB 完備化手続き[1)] で証明しようとす

ると，等式 $\mathsf{if}(x < x, \mathsf{s}^i(x), \mathsf{s}^i(x)) \approx \mathsf{s}^i(x)$ （$i \geq 0$）が残り続けて，完備化が暴走する．このような暴走は完備化の戦略を工夫することで抑制出来る可能性もあるが，P 文付き TRS を用いることで，比較演算子の書換えに対する完備化作業が除去できること，それによりそのような戦略が不要になるという利点もある．さらに，それにより完備化の複雑さを本質的に問題のある箇所のみにしぼりこめ，定理自動証明で頻繁に必要とされる補題等式が構成しやすくなると期待できる．また，将来的には，負の数を扱えるように拡張する際に，コーディングや計算の規則の表現を単純化されると期待できる．よって，本論文で提案する P 文付き TRS での定理自動証明の枠組は，関数型プログラム上での定理自動証明においても有効であると考えられる．なお，上記の例を P 文付き TRS で表現すると以下のようになり，本論文で提案する完備化では検証は簡単に成功する．

$$\begin{cases} \mathsf{max}(x, y) \to y \Leftarrow x < y \\ \mathsf{max}(x, y) \to x \Leftarrow \neg(x < y) \\ \qquad\qquad \vdots \end{cases}$$

本論文は次のように構成される．2 節では項書換えに関する記法を説明する．3 節では検証の対象とする手続き型言語の構文と意味論を与える．4 節では検証に用いる制約付き TRS の枠組，手続き型プログラムから P 文付き TRS への変換アルゴリズムを提案し，その正当性を示す．5 節では制約付き TRS の危険対定理を与える．6 節では検証の実質的な作業にあたる制約付き TRS の完備化を提案する．7 節では潜在帰納法に基づいた検証手続きを提案し，自然数列の総和を求める関数での等価性検証の例を挙げる．8 節で関連研究との比較を述べ，9 節で今後の課題を挙げる．各定理の証明は付録に記載する．

2. 準　　備

本論文では，項書換えの一般的な記法に従う[1)]．

抽象書換え系 S は，対象となる集合 A と A 上の**簡約化関係**と呼ばれる二項関係 $\to$ の組 $(A, \to)$ である．$a \to b$ となるような b が存在しないとき，a は S の**正規形**という．このとき，$c \xrightarrow{*} a$ である $c \in A$ について，c **は** S **の正規形を持つ**という．S における正規形の集合を NF_S で表す．$a \xrightarrow{*} c \xleftarrow{*} b$ となるような $c \in A$ が存在するとき，a と b は**会同関係にある**といい，$a \downarrow b$ と表記する．$b \xleftarrow{*} a \xrightarrow{*} c$ ならば $b \downarrow c$ のとき，S は**合流性**を持つという．$b \leftarrow a \to c$ ならば $b \downarrow c$ のとき，S は**局所合流性**を持つという．任意の $a \in A$ に対して a から始まる $\to$ の無限系列が存在しないとき，S は**停止性**（または**強正規性**）を持つという．すべての $a \in A$ が正規形を持つとき，S は**弱停止性**（または**弱正規性**）を持つという．S が合流性と停止性を持つとき，S は**完備性**を持つという．

関数記号の集合 $\mathcal{F}$，**変数**の可算無限集合 $\mathcal{V}$，から生成されるすべての**項**の集合を $\mathcal{T}(\mathcal{F}, \mathcal{V})$ とする．項 t に現れるすべての変数の集合を $\mathcal{V}ar(t)$ で表す．項 s と t が同一であるときは $s \equiv t$ と記述する．項 t への 1 引数関数記号 f の n 回の適用は $f^n(t)$ と略記する．項 t における**位置**の集合を $\mathcal{O}(t)$ とする．位置 $p, q \in \mathcal{O}(t)$ に対して，$pp' = q$ を満たす p' が存在するとき，$p \leq q$ と書く．特に $p' \neq \varepsilon$ とき，$p < q$ と記す．$\mathtt{root}(s)$ は項 s の先頭（位置 ε）の記号を表す．項 s の位置 p にある項を t に置き換えて得られる項を $s[t]_p$ と書く．**ホール** $\Box \notin \mathcal{F}$ を特別な定数記号とする．**文脈**とは，$\Box$ を一つだけ含む項である．ホール自身も文脈であり，このような文脈を空の文脈という．文脈 $C[\,]$ において位置 p に出現するホール $\Box$ を項 t で置き換えることによって得られる項を $C[t]_p$ と記す．なお，p を省略してもよい．$\mathcal{F}, \mathcal{V}$ 上のすべての文脈の集合を $\mathcal{T}_\Box(\mathcal{F}, \mathcal{V})$ とする．項 t, u に対して $t \equiv C[u]_p$ となるような文脈 $C[\,]$ が存在するとき，u を t の**部分項**と呼ぶ．また，p における部分項 u を $t|_p$ と記す．

代入 σ の定義域と値域をそれぞれ $\mathcal{D}om(\sigma)$ $(= \{x \mid x \neq \sigma(x)\})$ と $\mathcal{R}an(\sigma)$ $(= \{\sigma(x) \mid x \in \mathcal{D}om(\sigma)\})$ で表す．値域に現れる変数の集合を $\mathcal{VR}an(\sigma) = \bigcup_{t \in \mathcal{R}an(\sigma)} \mathcal{V}ar(t)$ とする．$\mathcal{VR}an(\sigma) = \emptyset$ となるような代入を**基底代入**という．$\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\}$ であり，かつ $\sigma(x_i) \equiv t_i$ のとき，σ を $\{x_1 \mapsto t_1, \cdots, x_n \mapsto t_n\}$ と記す．項 t に対して，$\sigma(t)$ を t の**インスタンス**と呼び，$t\sigma$ と略記する．代入 σ, σ' について，$\mathcal{D}om(\sigma) = \mathcal{D}om(\sigma')$ かつすべての $x \in \mathcal{D}om(\sigma)$ において $\sigma(x) \equiv \sigma'(x)$ のとき，$\sigma = \sigma'$ と記述する．σ と σ' の合成は $\sigma\sigma'$ と記述し，$x(\sigma\sigma') \equiv \sigma'(\sigma(x))$ で定義する．σ の定義域を $X \subseteq \mathcal{V}$ に制限した代入 $\sigma|_X$ を $\{x \mapsto x\sigma \mid x \in \mathcal{D}om(\sigma) \cap X\}$ と定義する．代入 σ, θ に対して $\sigma\delta = \theta$ となる代入 δ が存在するとき，$\sigma \lesssim \theta$ と記す．

項 s と t が**単一化可能**とは，ある代入 σ が存在して $s\sigma \equiv t\sigma$ となることである．このとき，σ を s と t の**単一化子**という．s と t の単一化子 σ が，s，t の任意の単一化子 θ について $\sigma \lesssim \theta$ を満たすとき，σ は**最汎**であるという．

項上の半順序 $\succ$ が整礎であり，文脈と代入に閉じているとき，$\succ$ を**簡約化順序**と呼ぶ．

3. 手続き型言語の構文と意味論

本節では，検証対象の手続き型言語の仕様を定める．具体的には構文，評価意味論，計算意味論を与える．

3.1 構　　文

本論文で検証の対象プログラム（関数）の手続き部分は標準的な while 言語[20),26)] の構文に従ったステートメントの系列とする．対象とする手続き型言語を BNF 記法を用いて図 2 のように定義する．x は変数名を表す記号列とし，C 言語の記法に従う．n は自然数とする．大文字から始まる単語は非終端記号を表す．ϵ は空文を表す．また，宣言された変数の名前の重複もないこととする．ステートメント系列 ss の中で宣言される前に参照される（式などに現れる）変数を ss の**自由変数**と呼ぶ．さらに，if 文と while 文の内側では変数宣言は行われないこととする．

3.2 評価意味論と計算意味論

次に 3.1 小節で与えた手続き型言語の評価意味論 $\Rightarrow_A$，$\Rightarrow_B$ と計算意味論 $\Rightarrow_C$ を定義する[20),26)]．提案した言語の基本操作は代入文である．よって，意味論を定義する上で変数への値を保持するための**記憶**が必要である．記憶を M とすると，$\mathrm{M}[x]$ は M で記憶されている x の値を表す．$\mathrm{M}[x/v]$ を M での x の値の更新（追加）を表す．M の定義域を代入と同様に $\mathcal{D}om(\mathrm{M})$ で表す．$\mathrm{M}[x] = \mathrm{M}[y] = 1$ のとき，$\mathrm{M}[y/2, z/3]$ という操作を施して得られる記憶 M' では，$\mathrm{M}'[x] = 1$，$\mathrm{M}'[y] = 2$，$\mathrm{M}'[z] = 3$ となる．

評価関係 $\Rightarrow_A$ は一つの式 e と一つの記憶 M を取り，一つの値 n（自然数）を返す（$(\mathrm{e}, \mathrm{M}) \Rightarrow_A n$）．評価関係 $\Rightarrow_B$ は一つの論理式 b と一つの記憶 M を取り，真偽値（`true` または `false`）を返す（$(\mathrm{b}, \mathrm{M}) \Rightarrow_B c$ かつ c は `true` か `false` のどちらか）．$\Rightarrow_A$ と $\Rightarrow_B$ の意味論は文献[20),26)] に従うこととし，それぞれ一般の自然数上の四則演算[⋆1]とブール演算と一致する．なお，本論文では $\Rightarrow_A$ と $\Rightarrow_B$ の定義は省略する．

計算意味論 $\Rightarrow_C$ は，ステートメントの系列と一つの記憶を取り，ステートメントが実行（評価）され，残りの系列と更新された記憶が返される．その意味論を図 3 で与える．また，このように与えた計算意味論 $\Rightarrow_C$ による評価結果（最終的な記憶）は一意であることが知られている．

3.3 検証の対象とする手続き型関数の定義

本論文で検証の対象とするプログラム（関数）は，

ComR

$$\frac{(\mathrm{ss}_1, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}') \quad (\mathrm{ss}_2, \mathrm{M}') \Rightarrow_C (\epsilon, \mathrm{M}'')}{(\mathrm{ss}_1\,\texttt{;}\,\mathrm{ss}_2, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}'')}$$

InitR

$$\frac{(\mathrm{e}, \mathrm{M}) \Rightarrow_A v}{(\texttt{int}\ x\ \texttt{=}\ \mathrm{e}, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}[x/v])}$$

AsR

$$\frac{(\mathrm{e}, \mathrm{M}) \Rightarrow_A v}{(x\ \texttt{=}\ \mathrm{e}, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}[x/v])}$$

IfR

$$\frac{(\mathrm{b}, \mathrm{M}) \Rightarrow_B \texttt{true} \quad (\mathrm{ss}_1, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}')}{(\texttt{if(b)\{}\mathrm{ss}_1\texttt{\}else\{}\mathrm{ss}_2\texttt{\}}, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}')}$$

$$\frac{(\mathrm{b}, \mathrm{M}) \Rightarrow_B \texttt{false} \quad (\mathrm{ss}_2, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}')}{(\texttt{if(b)\{}\mathrm{ss}_1\texttt{\}else\{}\mathrm{ss}_2\texttt{\}}, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}')}$$

WhileR

$$\frac{(\mathrm{b}, \mathrm{M}) \Rightarrow_B \texttt{false}}{(\texttt{while(b)\{}\mathrm{ss}\texttt{\}}, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}')}$$

$$\frac{(\mathrm{b}, \mathrm{M}) \Rightarrow_B \texttt{true} \quad (\mathrm{ss}\texttt{;}\ \texttt{while(b)\{}\mathrm{ss}\texttt{\}}, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}')}{(\texttt{while(b)\{}\mathrm{ss}\texttt{\}}, \mathrm{M}) \Rightarrow_C (\epsilon, \mathrm{M}')}$$

図 3 手続き型言語の計算意味論：$\Rightarrow_C$
Fig. 3 Semantics $\Rightarrow_C$ of the procedural language.

図 2 の構文に従ったステートメント系列と，末尾で値を返すための return 文で定義される関数とする．ss をステートメントの系列，$x_1, \ldots, x_n$ を ss の自由変数，e を式（出現する変数はすべて ss に出現）とすると，n 引数関数 f の関数定義は以下のようになる：

`int` f`(` `int` x_1`,` $\cdots$`,` `int` x_n `){` ss`;` `return` e`;` `}`

f は関数名を表す記号列とし，C 言語の記法に従う．さらに，そのように定義される関数のうち，正常にコンパイルできる C 言語の関数を扱うこととする．C 言語にはブール型はないが，ブール型の概念に矛盾しない条件文，すなわち，ブール型を導入してもそのままブール式として型が付けられる条件文を扱う．E と E_a，B から生成される数式，ブール式の括弧は慣例に従い，省略してもよいこととする．

本論文で検証の対象とする手続き型関数の特長は以下のようにまとめられる．

- 制御文として if 文と while 文を書ける．
- 型は自然数（`int`）[⋆2]に限る．
- 関数呼び出しはない．
- 変数宣言の際に必ず初期化される．
- 関数定義の末尾で return 文により値を返す．

⋆1 自然数上では $m - n < 0$ のときは $m - n$ を 0 と評価するが，本論文では $m - n < 0$ となる状況に直面しない例を扱う．

⋆2 `int` は本来，整数に対して用いられる型名であるが，本論文で扱う手続き型プログラムは C 言語の記法に準拠するため，本論文では `int` を自然数の型名として用いる．

$$
\begin{aligned}
S &::= \epsilon \mid S\ \texttt{;}\ S \mid x\ \texttt{=}\ E \mid \texttt{int}\ x\ \texttt{=}\ E \mid \texttt{if(}\ B\ \texttt{)\{}\ S\ \texttt{\}else\{}\ S\ \texttt{\}} \mid \texttt{while(}\ B\ \texttt{)\{}\ S\ \texttt{\}}\\
E &::= x \mid n \mid \texttt{(}\ E\ \texttt{)} \mid \texttt{(}\ E\ \texttt{+}\ E\ \texttt{)} \mid \texttt{(}\ E\ \texttt{-}\ E\ \texttt{)} \mid \texttt{(}\ E\ \texttt{*}\ E\ \texttt{)} \mid \texttt{(}\ E\ \texttt{/}\ E\ \texttt{)}\\
B &::= B_E \mid \texttt{(}\ B\ \texttt{)} \mid \texttt{(}\ B\ \texttt{||}\ B\ \texttt{)} \mid \texttt{(}\ B\ \texttt{\&\&}\ B\ \texttt{)} \mid \texttt{(!}B\ \texttt{)}\\
B_E &::= \texttt{false} \mid \texttt{true} \mid E_a\ \texttt{==}\ E_a \mid E_a\ \texttt{!=}\ E_a \mid E_a\ \texttt{<}\ E_a \mid E_a\ \texttt{<=}\ E_a \mid E_a\ \texttt{>}\ E_a \mid E_a\ \texttt{>=}\ E_a\\
E_a &::= x \mid n \mid \texttt{(}\ E_a\ \texttt{)} \mid \texttt{(}\ E_a\ \texttt{+}E_a\ \texttt{)} \mid \texttt{(}\ E_a\ \texttt{-}E_a\ \texttt{)}
\end{aligned}
$$

図 2 手続き型言語の構文
Fig. 2 Syntax of our procedural language.

```
int sum1( int n ){
    int i = 1;
    int z = 0;
    while( i <= n ){
        z = z + i;
        i = i + 1;
    }
    return z;
}
```

図 4 手続き型プログラム sum1
Fig. 4 Procedural program sum1.

本論文では特に自然数のみを扱う．また，手続き型プログラムの return 文の位置の制限が強いが，これは制約付き TRS への変換の正当性の証明を簡潔にするためであり，本質的ではない．

例 3.1 図 4 に示す手続き型プログラムは自然数 n までの総和を求める関数である． □

4. 制約付き TRS への変換

本節では，3 節で定めた手続き型言語プログラムを検証するための関数型言語のモデルとして制約付き TRS の枠組を与え，制約付き TRS に属するプレスブルガー文制約付き TRS への変換アルゴリズムを提案する．また，変換アルゴリズムの正当性を示す．

4.1 制約付き項書換え系

関数記号の集合を $\mathcal{F}$ とする．$\mathcal{F}$ の部分集合 $\mathcal{G}$ の基底項上の**一階述語**を $\mathcal{G}$ **上の述語**と呼び，意味を与えられているとする．すなわち，$\mathcal{G}$ の基底項が引数に与えられたアトムの真偽値が定められていることとする．p を $\mathcal{G}$ 上の一階述語（n 引数），$t_1, \ldots, t_n$ を $\mathcal{G}$ 上の基底項としたとき，$p(t_1, \ldots, t_n)$ を**アトム**と呼ぶ．$\mathcal{G}$ 上の**一階述語論理式**とは真偽値（$\texttt{true}, \texttt{false}$），アトム，$\wedge$，$\vee$，$\neg$，$\forall$，$\exists$ から構成される論理式であり，その論理式すべての集合を $\mathcal{P}$ と表記する．以降では，$\mathcal{G}$ を $\mathcal{F}_{\mathcal{P}}$ と記述する．制約 c に現れるすべての**自由変数**の集合を $\mathrm{fv}(c)$ と表記する．σ を $\mathcal{R}an(\sigma) \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$ を満たす代入とする．$\mathcal{R}an(\sigma|_{\mathrm{fv}(c)}) \subseteq \mathcal{T}(\mathcal{F}_{\mathcal{P}}, \mathcal{V})$ であるときのみ σ を c へ適用することを許し，c へ σ を適用（入力）して得られる論理式を $c\sigma$ と表記する（$c\sigma \in \mathcal{P}$ である）．

以降では，論理式として，その閉論理式の真偽値が決定可能であるもののみを扱い，$\mathcal{P}$ でそれらの集合を表す．$(\mathcal{F}, \mathcal{P})$ 上の**制約付き書換え規則** (l, r, c) は，$l \notin \mathcal{F}$，$l, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$，$\mathcal{V}ar(l) \supseteq \mathcal{V}ar(r)$ を満たす**左辺**項 l，**右辺**項 r と，$\mathcal{V}ar(l) \supseteq \mathrm{fv}(c)$ と $c \in \mathcal{P}$ を満たす**制約部** c の組であり，$l \to r \Leftarrow c$ と記す．c が $\texttt{true}$ のときは制約部を省略して $l \to r$ と書くこともある．

R を $(\mathcal{F}, \mathcal{P})$ 上の制約付き書換え規則の有限集合とする．R で定まる書換え関係 $\to_R$ を，$\to_R = \{(C[l\sigma]_p, C[r\sigma]_p) \mid l \to r \Leftarrow c \in R,\ C[\ ] \in \mathcal{T}_{\Box}(\mathcal{F}, \mathcal{V}),\ c\sigma \text{ が真 }\}$ と定義する[⋆1]．$(\mathcal{F}, \mathcal{P})$ 上の**制約付き項書換え系**（constraint TRS）は項の集合 $\mathcal{T}(\mathcal{F}, \mathcal{V})$ と書換え関係 $\to_R$ で定められる抽象書換え系 $(\mathcal{T}(\mathcal{F}, \mathcal{V}), \to_R)$ であり，書換え規則の集合 R で表す．制約部が $\texttt{true}$ である規則のみからなる制約付き TRS は**項書換え系**（TRS）である．R の $\mathcal{F}, X$ 上のすべての正規形の集合を $NF_R(\mathcal{F}, X)$ とし，$X = \emptyset$ のときは $NF_R(\mathcal{F})$ と書く．

本論文では，プレスブルガー文[8] 上の制約付き TRS を扱う．**プレスブルガー算術**とは加算を持つ整数の理論（整数，整数上の変数，整数の加減演算 $\texttt{+}$，$\texttt{-}$，整数の比較演算 $\texttt{==}$，$\texttt{!=}$，$\texttt{<}$，$\texttt{<=}$，$\texttt{>}$，$\texttt{>=}$，論理演算 $\vee$，$\wedge$，$\neg$，限量子 $\forall$ と $\exists$ からなる理論）であり，プレスブルガー算術上の閉論理式を**プレスブルガー文**[⋆2]（**P 文**）という．なお，本論文では P 文の算術演算しと比較演算子の記法は C 言語に従い，自然数上に制限した P 文を扱うこととする．P 文の真偽判定は決定

⋆1 基底項上の述語を扱うので，$c\sigma$ の真と解釈されるときは $\mathrm{fv}(c\sigma) = \emptyset$ である．よって，$\mathcal{R}an(\sigma|_{\mathrm{fv}(c)}) \subseteq \mathcal{T}(\mathcal{F}_{\mathcal{P}})$ である．
⋆2 本論文では閉じていないものも P 文と呼ぶ．

可能であることが知られている[17),22),24)⋆1]．制約部を P 文に制限した制約付き TRS を **P 文付き項書換え系**（Presburger-constraint TRS）と呼ぶ．すなわち，P 文の集合を $\mathcal{P}bgr$ とすると，P 文付き TRS は $(\mathcal{F}, \mathcal{P}bgr)$ 上の制約付き TRS である．本論文では，P 文の変数の値域となる項を表現する関数記号の集合 $\mathcal{F}_{\mathcal{P}bgr}$ を $\{+, -, \mathsf{s}, \mathsf{0}\}$ とする．中置記法で記述される 2 引数関数記号 $+$，$-$ はそれぞれ加算，減算と解釈し，括弧が省略されている場合は左結合として扱う．項 $\mathsf{s}^n(\mathsf{0})$ は自然数 n と解釈する．算術式（`+`，`-`，`*`，`/`，変数，自然数からなる式）から $\mathcal{F}_{\mathcal{P}bgr} \cup \{\times, /\}$ 上の項への変換 $\mathbb{E}$ は以下のように定義される．

- $\mathbb{E}(x) = x$.
- $\mathbb{E}(n) = \mathsf{s}^n(\mathsf{0})$.
- $\mathbb{E}(\mathrm{e}_1 \,\mathtt{+}\, \mathrm{e}_2) = \mathbb{E}(\mathrm{e}_1) + \mathbb{E}(\mathrm{e}_2)$（`-`，`*`，`/`の場合は $+$ をそれぞれ $-$，$\times$，$/$ で置換える）．

P 文への算術式（`+`，`-`，変数，自然数からなる式）の代入は項の代入と同様にする．P 文への $\mathcal{F}_{\mathcal{P}bgr}$ 上の項の代入は，項を対応する算術式に変換し P 文へ代入し，多項式へ正規化して得られる P 文を結果とする．つまり，P 文 c への代入 θ の適用 $\theta(c)$ は，対応する項代入 σ（$\mathcal{D}om(\sigma) = \mathcal{D}om(\theta)$ かつ $\forall x.\mathbb{E}(x\theta) \equiv x\sigma$）を用いて，$\theta(c) = Norm(\mathbb{E}^{-1}(\mathbb{E}(c)\sigma))$ と定義できる．ここで，$Norm$ は算術式を多項式へ正規化する関数，$\mathbb{E}^{-1}$ は算術式から $\mathcal{F}_{\mathcal{P}bgr} \cup \{\times, /\}$ 上の項への変換 $\mathbb{E}$ の逆変換とする．なお，算術式を代入して正規化した P 文も P 文になる．算術式の多項式への正規化は一般的な方法を用いることとし，本論文では詳細については割愛する．

R を $(\mathcal{F}, \mathcal{P})$ 上の制約付き TRS とする．**被定義記号**の集合を $\mathcal{D}_R = \{\mathtt{root}(l) \mid l \to r \Leftarrow c \in R\}$，**構成子**の集合を $\mathcal{C}_R = \mathcal{F} \setminus \mathcal{D}_R$ と定義する．どの規則の左辺も根の位置以外に被定義記号を持たないとき，R を**構成子システム**と呼ぶ．本論文では $\mathcal{C}_R \subseteq \mathcal{F}_{\mathcal{P}}$ を仮定する．

4.2 P 文付き TRS への変換アルゴリズム

本小節では，図 2 で定義されたステートメントの系列を基に定められる手続き型プログラムを P 文付き TRS に変換するアルゴリズム $\mathbb{TR}$ を提案する．

手続き型の関数定義を P 文付き TRS に変換するアルゴリズム $\mathbb{TR}$ を，図 2 の構文で記述されたステートメントの系列を P 文付き TRS を変換するアルゴリズム $\mathbb{T}$（図 5）を基に，図 6 のように与える．図 4 のプログラム `sum1` を $\mathbb{TR}$ で変換した例については，例 4.1 で示す．

まずは，図 2 の構文で記述されたステートメントの系列を P 文付き TRS を変換するアルゴリズム $\mathbb{T}$（図 5）について説明する．変換 $\mathbb{T}$ はステートメントを一つ取り出して，その種類で場合分けしてそのステートメントの動作に対応した書換え規則を生成する．$\overrightarrow{\mathcal{V}ar(t)}$ は t に含まれる変数を特定の順序で並べた列である．$\mathbb{T}$ で生成された規則は元のプログラムの制御の流れを表現する．この変換は決定性を持たないが，ラベルの重複利用が起こらないので，生成される結果は一意である．

$\mathbb{T}(t, \mathrm{ss}, i) = (t', R, i')$ とする．このとき，ss の自由変数は t に出現していることとする．t は計算したい項であり，次に生成する規則の左辺となる．ss はステートメントの系列で，これから P 文付き TRS へと変換するプログラムである．R は t を計算するために変換で得られる規則の集合である．自然数 i はプログラムの内部状態を表す関数記号 u を識別するためにラベルとして利用する．$\mathrm{u}_i(x_1, \ldots, x_m)$ は，ラベル i が指す（ラベル i を u に付ける際に読んでいる）命令文を評価する直前のメモリの状態を表している．u_i の導入は条件付き TRS から TRS への変換[16)] に基づいている．if と while の構造で生成される規則の役割を以下にまとめる．

（if の場合）規則 $t \to \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}) \Leftarrow b$ は if-then-else 構造の then 節への遷移，規則 $t \to \mathrm{u}_{i_1}(\overrightarrow{\mathcal{V}ar(t)}) \Leftarrow \neg b$ は else 節への遷移を表す．P 文付き TRS R_1 と R_2 はそれぞれ then 節と else 節の処理を定める規則集合であり，項 t_1 と t_2 はそれぞれ then 節と else 節の処理が終了した際の状態を表現する．規則 $t_i \to \mathrm{u}_{i_2}(\overrightarrow{\mathcal{V}ar(t)})$（$i = 1, 2$）はそれぞれ then 節と else 節から if-then-else 節を終了した状態を表す項 $\mathrm{u}_{i_2}(\overrightarrow{\mathcal{V}ar(t)})$ への遷移を表現する．

（while の場合）規則 $t \to \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}) \Leftarrow b$ と $t \to \mathrm{u}_{i'}(\overrightarrow{\mathcal{V}ar(t)}) \Leftarrow \neg b$ は while ループの内部処理を繰り返すかどうかの分岐を表現し，前者は while ループの内部への遷移を，後者は while ループが終了した状態を表す項 $\mathrm{u}_{i'}(\overrightarrow{\mathcal{V}ar(t)})$ への遷移を表現する．項 t' は while ループの内部の処理が終了した状態を表し，規則 $t' \to t$ により再び while ループの条件を判定するために，項 t へ遷移する．

条件文の P 文では，ブール値とブール演算子 `0`，`1`，`&&`，`||`，`!` はそれぞれ `true`，`false`，$\wedge$，$\vee$，$\neg$ に置き換える．

式を評価する P 文付き TRS R_c は $\{\, \mathsf{0} + y \to y,$

⋆1 恒真性・充足可能性は自由変数をそれぞれ $\forall$，$\exists$ で束縛して真偽判定することで決定可能である．

- $\mathbb{T}(\ t,\ \epsilon,\ i\ ) = (\ t,\ \emptyset,\ i\ )$.
- $\mathbb{T}(\ t,\ \mathrm{ss}_1\ ;\ \mathrm{ss}_2,\ i\ ) = (\ t_2,\ R_1 \cup R_2,\ i_2\ )$
  ただし，$\mathbb{T}(\ t,\ \mathrm{ss}_1,\ i\ ) = (\ t_1,\ R_1,\ i_1\ )$ かつ $\mathbb{T}(\ t_1,\ \mathrm{ss}_2,\ i_1\ ) = (\ t_2,\ R_2,\ i_2\ )$.
- $\mathbb{T}(\ t,\ \texttt{int}\ x\ \texttt{=}\ \mathrm{e},\ i\ ) = (\ \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}, x),\ \left\{\ t \to \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}, \mathbb{E}(\mathrm{e}))\ \right\},\ i+1\ )$.
- $\mathbb{T}(\ t,\ x\ \texttt{=}\ \mathrm{e},\ i\ ) = (\ \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}),\ \left\{\ t \to \left(\mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)})\right)\{x \mapsto \mathbb{E}(\mathrm{e})\}\ \right\},\ i+1\ )$.
- $\mathbb{T}(\ t,\ \texttt{if(}\ \mathrm{b}\ \texttt{)\{}\ \mathrm{ss}_1\ \texttt{\}else\{}\ \mathrm{ss}_2\ \texttt{\},}\ i\ ) = (\ \mathrm{u}_{i_2}(\overrightarrow{\mathcal{V}ar(t)}),\ R_1 \cup R_2 \cup \left\{\begin{array}{l} t \to \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}) \Leftarrow \mathrm{b}, \\ t \to \mathrm{u}_{i_1}(\overrightarrow{\mathcal{V}ar(t)}) \Leftarrow \neg\mathrm{b}, \\ t_1 \to \mathrm{u}_{i_2}(\overrightarrow{\mathcal{V}ar(t)}), \\ t_2 \to \mathrm{u}_{i_2}(\overrightarrow{\mathcal{V}ar(t)}) \end{array}\right\},\ i_2+1\ )$,
  ただし，$\mathbb{T}(\ \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}),\ \mathrm{ss}_1,\ i+1\ ) = (\ t_1,\ R_1,\ i_1\ )$ かつ $\mathbb{T}(\ \mathrm{u}_{i_1}(\overrightarrow{\mathcal{V}ar(t)}),\ \mathrm{ss}_2,\ i_1+1\ ) = (\ t_2,\ R_2,\ i_2\ )$.
- $\mathbb{T}(\ t,\ \texttt{while(}\ \mathrm{b}\ \texttt{)\{}\ \mathrm{ss}\ \texttt{\},}\ i\ ) = (\ \mathrm{u}_{i'}(\overrightarrow{\mathcal{V}ar(t)}),\ R \cup \left\{\begin{array}{l} t \to \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}) \Leftarrow \mathrm{b}, \\ t \to \mathrm{u}_{i'}(\overrightarrow{\mathcal{V}ar(t)}) \Leftarrow \neg\mathrm{b}, \\ t' \to t \end{array}\right\},\ i'+1\ )$,
  ただし，$\mathbb{T}(\ \mathrm{u}_i(\overrightarrow{\mathcal{V}ar(t)}),\ \mathrm{ss},\ i+1\ ) = (\ t',\ R,\ i'\ )$.

図 5 ステートメント系列から P 文付き TRS への変換アルゴリズム $\mathbb{T}$
Fig. 5 Transformation $\mathbb{T}$ from statement sequences into Presburger-constraint TRSs.

$\mathbb{TR}(\texttt{int}\ f\texttt{(int}\ x_1, \ldots, \texttt{int}\ x_n\texttt{)\{}\ \mathrm{ss};\ \texttt{return}\ \mathrm{e};\texttt{\}}) = R \cup \{\ t \to \mathrm{u}_i(\mathbb{E}(\mathrm{e})),\ \mathrm{u}_i(y) \to y\ \}$
ただし，$\mathbb{T}(f(x_1, \ldots, x_n), \mathrm{ss}, 1) = (t, R, i)$ とする（u のラベルの開始番号は 1 でなくてもよい）.

図 6 C 言語の関数から P 文付き TRS への変換 $\mathbb{TR}$.
Fig. 6 Transformation $\mathbb{TR}$ from procedural functions into Presburger-constraint TRSs.

$\mathsf{s}(x) + y \to \mathsf{s}(x+y), \cdots\}$ のようにあらかじめ完備性と十分完全性を持つように与えておく．すなわち，任意の記憶 M と項 $t \in \mathcal{T}(\{+, -, \times, /, \mathsf{s}, 0\}, \mathcal{V})$ と基底代入 θ（$\mathbb{E}(x\theta) = \mathrm{M}[x]$）自然数 n について，$(\mathbb{E}^{-1}(t), \mathrm{M}) \Rightarrow_A n$ のとき，かつそのときに限り，$t\theta \xrightarrow{*}_{R_c} \mathsf{s}^n(0)$ を満たすように R_c を与える．除算は例外処理がありこの条件を満たさないので，今回は用いないこととする．

最後に，関数定義を P 文付き TRS に変換するアルゴリズム $\mathbb{TR}$（図 6）を説明する．u_i は return 文で返す式の値を評価する状態を表現し，規則 $\mathrm{u}_i(y) \to y$ で値を返す．

$$R_{\mathtt{sum1}} = \left\{\begin{array}{rcl} \mathtt{sum1}(n) & \to & \mathrm{u}_1(n, \mathsf{s}(0)) \\ \mathrm{u}_1(n, i\ \ ) & \to & \mathrm{u}_2(n, i, 0) \\ \mathrm{u}_2(n, i, z) & \to & \mathrm{u}_3(n, i, z) \Leftarrow i \mathtt{<=} n \\ \mathrm{u}_3(n, i, z) & \to & \mathrm{u}_4(n, i, z+i) \\ \mathrm{u}_4(n, i, z) & \to & \mathrm{u}_5(n, i+\mathsf{s}(0), z) \\ \mathrm{u}_5(n, i, z) & \to & \mathrm{u}_2(n, i, z) \\ \mathrm{u}_2(n, i, z) & \to & \mathrm{u}_6(n, i, z) \Leftarrow \neg(i \mathtt{<=} n) \\ \mathrm{u}_6(n, i, z) & \to & \mathrm{u}_7(z) \\ \mathrm{u}_7(y) & \to & y \end{array}\right.$$

例 4.1 図 4 のプログラム `sum1` を $\mathbb{TR}$ で変換して得られる P 文付き TRS は次のようになる．

$\mathtt{sum1}(\mathsf{s}^{10}(0))$ は最左最内書換えで次のように計算される．

$$
\begin{aligned}
&\mathtt{sum1}(\mathsf{s}^{10}(0)) \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_1(\mathsf{s}^{10}(0), \mathsf{s}(0)) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_2(\mathsf{s}^{10}(0), \mathsf{s}(0), 0) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_3(\mathsf{s}^{10}(0), \mathsf{s}(0), 0) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_4(\mathsf{s}^{10}(0), \mathsf{s}(0), 0 + \mathsf{s}(0)) \\
&\quad \overset{*}{\to}_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_4(\mathsf{s}^{10}(0), \mathsf{s}(0), \mathsf{s}(0)) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_5(\mathsf{s}^{10}(0), \mathsf{s}(0) + \mathsf{s}(0), \mathsf{s}(0)) \\
&\quad \overset{*}{\to}_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_5(\mathsf{s}^{10}(0), \mathsf{s}^2(0), \mathsf{s}(0)) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_2(\mathsf{s}^{10}(0), \mathsf{s}^2(0), \mathsf{s}(0)) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \cdots \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_2(\mathsf{s}^{10}(0), \mathsf{s}^{11}(0), \mathsf{s}^{55}(0)) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_6(\mathsf{s}^{10}(0), \mathsf{s}^{11}(0), \mathsf{s}^{55}(0)) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{u}_7(\mathsf{s}^{55}(0)) \\
&\quad \to_{R_{\mathtt{sum1}} \cup R_c} \mathsf{s}^{55}(0)
\end{aligned}
$$

一方，$\mathtt{sum1}(\mathsf{s}(x))$ は $\mathsf{u}_2(\mathsf{s}(x), \mathsf{s}(0), 0)$ までは書き換えられるが，これ以上書き換えることはできない． □

4.3 変換アルゴリズムの正当性

最後に，提案した変換アルゴリズム $\mathbb{TR}$ の正当性を示す．ここでいう正当性とは手続き型プログラムでの計算とその手続き型プログラムを変換した P 文付き TRS での計算は一致することである．以降では，式 e を項に変換した $\mathbb{E}(\mathrm{e})$ を t_{e} と書く．記憶 M に対して代入 θ_{M} を $\mathcal{D}om(\theta_{\mathrm{M}}) = \mathcal{D}om(\mathrm{M})$ かつ，すべての $x \in \mathcal{D}om(\theta_{\mathrm{M}})$ に対して $x\theta_{\mathrm{M}} = t_{\mathrm{M}[x]}$ と定義する．

定理 4.2 ss をステートメントの系列，プログラム P を $\mathtt{int}\ f(\mathtt{int}\ x_1, \ldots, \mathtt{int}\ x_n)\{\mathrm{ss};\ \mathtt{return}\ \mathrm{e};\}$，M と M′ を記憶，$m$ を自然数とする．$(\mathrm{ss}, \mathrm{M}) \overset{*}{\Rightarrow}_C (\epsilon, \mathrm{M}')$ かつ $(\mathrm{e}, \mathrm{M}') \Rightarrow_A m$ のとき，かつこのときに限り，$f(x_1, \ldots, x_n)\theta_{\mathrm{M}} \overset{*}{\to}_{\mathbb{TR}(P) \cup R_c} \mathsf{s}^m(0)$．

［証明］ 付録 A.1 を参照． □

上記の定理は計算が停止する場合に，P の内部（記憶）の状態の遷移とそれに対応する $\mathsf{u}_i(\cdots)$ の項の書換えが一致していることを示している．一方，本論文では証明は与えないが，停止性がない（値を返さない）ような場合でも同様の対応関係が保たれている．

5. 制約付き TRS の危険対定理

本論文は，制約付き TRS 上の完備化手続きに基づいた検証法の実現をめざしている．そのためには，完備化で得られる TRS の合流性を保証するための基本原理である**危険対定理**を制約付き TRS 上へ拡張する必要がある．本節では，制約付き TRS の危険対の概念を導入し，危険対定理が制約付き TRS で成立する条件を示す．そして，変換で得られる P 文付き TRS が局所合流性を持つことを示す．

制約付き TRS の危険対の概念を CTRS の場合[23]と同様に与える．$l_1 \to r_1 \Leftarrow c_1$，$l_2 \to r_2 \Leftarrow c_2$ を $\mathcal{V}ar(l_1) \cap \mathcal{V}ar(l_2) = \emptyset$ となるように変数を名前替えした規則とする．$p \in \mathcal{O}(l_1)$ を $l_1|_p$ が変数でない位置，$l_1|_p\sigma \equiv l_2\sigma$ となるような最汎単一化子 σ が存在したとき，$\langle r_1\sigma, (l_1[r_2]_p)\sigma \rangle \Leftarrow c_1\sigma \wedge c_2\sigma$ を**制約付き危険対**と呼ぶ[⋆1]．ただし，2 つの規則が同じ（変数の名前替え同士）である際には，$C \neq \Box$ とする．このような危険対が存在するとき，2 つの規則は**重なる**という．制約付き TRS R のすべての危険対の集合を $CP(R)$ で表す．さらに，$l \to r \Leftarrow c$ と R の規則との間で形成される危険対の集合を $CP(l \to r \Leftarrow c, R)$ と書く．ある制約付き危険対 $\langle s, t \rangle \Leftarrow c$ に対して，制約 c を充足するような代入 σ が存在するときその危険対は**可能**（feasible）であるという．また，そうでないときその危険対は**不能**（infeasible）であるという．本論文では，危険対が不能になる 2 つの規則については，その危険対を構成した位置（危険対の定義中の p）に関しては重なりがないとみなす．

R を制約付き TRS とする．項 s と t が**制約 c の下で会同関係にある**とは，任意の代入 σ に対して，$c\sigma$ が真ならば $s\sigma \downarrow_R t\sigma$ が成り立つことである．また，制約付き項対 $\langle s, t \rangle \Leftarrow c$ が**会同関係にある**とは，s と t が c の下で会同関係にあることである．

TRS の合流性に関する危険対補題は，制約付き TRS 上ではそのまま成り立たない．書換えの分岐が重なりのない上下の位置で起こる場合に問題が生じる．例えば，述語を項の構造等価関係 $\equiv$ とした制約付き TRS $\{\, \mathsf{f}(x) \to \mathsf{c} \Leftarrow x \equiv \mathsf{a},\ \mathsf{a} \to \mathsf{b} \,\}$ を考える．すると，項 $\mathsf{f}(\mathsf{a})$ は c と $\mathsf{f}(\mathsf{b})$ のどちらにも書換えられる．しかし，これら 2 つの項は会同関係にないため，この制約付き TRS は規則に重なりがないにもかかわらず合流性を持たない．これは制約 $x \equiv \mathsf{a}$ の a が暗に規則 $\mathsf{a} \to \mathsf{b}$ と重なりと同様の現象を起こしているからである．そこで，関連の規則によって制約部の真偽が変化しないという性質を考える．

定義 5.1 (制約部安定性) 制約 $c \in \mathcal{P}$ が任意の代入 θ に対して以下を満たすとき，c は**二項関係 $\to$ に関し**

⋆1 危険対 $\langle s, t \rangle \Leftarrow c$ は一般には $\mathcal{V}ar(s, t) \supseteq \mathrm{fv}(c)$ を満たす（c 中の変数が必ずしも s, t に出現する）とは限らない．$\mathrm{fv}(c) \setminus \mathcal{V}ar(s, t)$ の変数は，規則の場合と異なり，限量子による束縛が必要な際には $\exists$ で束縛する．

て真偽安定性を持つという．

“任意の自由変数 $x \in \mathrm{fv}(c)$ について $x\theta$ $(\to \cup \equiv)$ $x\sigma$” を満たすすべての代入 σ に対して，$c\theta$ と $c\sigma$ の真偽値が一致する．

R のすべての規則の制約部が $\to$ に関して真偽安定性を持つとき，制約付き TRS R は $\to$ **に関して制約部安定性**を持つといい，特に，$\to$ が $\xrightarrow{*}_R$ に相当するときは，単に**制約部安定性**を持つという． □

自然数の加減算を表現した規則 R_c が自然数の一般の計算法則に矛盾しなければ，P 文付き TRS は制約部安定性を持つ．

命題 5.2 P 文付き TRS R $(\supseteq R_c)$ は $R \setminus R_c$ に $+, -$ の規則は持たないとする[⋆1]．このとき，R は制約部安定性を持つ．

［証明］ 付録 A.2 を参照． □

制約部安定性を持つ制約付き TRS において，以下の制約付き危険対に関する補題と定理が成り立つ．

補題 5.3 (制約付き危険対補題) R を制約部安定性を持つ制約付き TRS，s, t_0, t_1 を項とする．$s \to_R t_i$ $(i = 0, 1)$ ならば，以下のどちらかを満たす．

- $t_0 \downarrow_R t_1$.
- $t_i = s[u_i]_p$ $(i = 0, 1)$．ただし，$c\sigma$ が真となる危険対 $\langle v_0, v_1 \rangle \Leftarrow c \in CP(R)$ と代入 σ が存在して，$u_i \equiv v_i\sigma$ $(i = 0, 1)$ または $u_{1-i} \equiv v_i\sigma$ $(i = 0, 1)$．

［証明］ 付録 A.3 を参照． □

制約付き危険対補題より以下の定理が得られる．

定理 5.4 (制約付き危険対定理) R を制約部安定性を持つ制約付き TRS とする．R が局所合流性を持つとき，かつそのときに限り，R から作られるすべての可能な R の制約付き危険対は合同関係にある． □

最後に変換で得られる P 文付き TRS が局所合流性を持つことを示す．

命題 5.5 R_c が完備性を持つとき，$\mathbb{TR}(P) \cup R_c$ は局所合流性を持つ P 文付き TRS である．

⋆1 すなわち，$\{\mathtt{root}(l) \mid l \to r \Leftarrow c \in R \setminus R_c\} \cap \{+, -\} = \emptyset$.

［略証］ 図 5 の $\mathbb{T}$ の定義中で，t を左辺とする規則は高々 2 つしか生成されず，2 つ生成されるときは制約部にそれぞれ b, $\neg b$ が付加される．よって，$\mathbb{TR}$ で生成される規則によって生じる危険対は $b \wedge \neg b$ を制約として持つ．また，$\mathbb{TR}(P)$ と R_c の間に危険対は生じない．これらのことから，$\mathbb{TR}(P)$ はそれ自身もしくは R_c との間に可能な危険対を持たない．また，R_c は完備性を持つように与えられている．よって，$\mathbb{TR}(P) \cup R_c$ は局所合流性を持つ． □

$CP(\mathbb{TR}(P)) = \emptyset$ でかつ $\mathbb{TR}(P)$ と R_c の規則は重ならないので危険対は存在しない．よって，R_c の性質次第では，$\mathbb{TR}(P) \cup R_c$ が停止性を持たずとも合流性を持つ場合がある．例えば，R_c が直交性を持つならば $\mathbb{TR}(P) \cup R_c$ も直交性を持つことが期待できる．この性質が証明できれば，R_c が直交性を持つときに $\mathbb{TR}(P) \cup R_c$ は合流性を持つことを示せる．

6. 制約付き TRS における完備化手続き

7 節で述べるように，完備化手続きを基にした帰納的定理の証明では，証明したい等式と書換え系の対から手続きを開始し，基底正規形の集合を変化させずに等式集合を空にすることを目標とする．そこで，本節では，TRS における完備化[1),2)] を制約付き TRS に拡張する．まずは，TRS の完備化手続き[1),2)] を基に制約付き TRS の完備化のための推論規則，完備化手続きを与える．また，制約付き TRS 固有の推論規則を提案する．

6.1 制約付き書換え

完備化では制約付きの等式の両辺や規則の右辺を書き換える必要がある．そこで，制約を環境とした下での書換えを定義する．c を充足可能な論理式としたとき，任意の代入 θ に対して $c\theta$ が真ならば $d\theta$ も真（すなわち，$\vec{x}$ を c, d のすべての自由変数の並びとすると，“$\forall \vec{x}.(\neg c \vee d)$” が真[⋆2]）であることを $c \models d$ と書く．**充足可能な制約 c の下での書換え関係** $\underset{c}{\to}_R$ を $\{ (C[l\sigma], C[r\sigma]) \mid l \to r \Leftarrow d \in R,\ C[\,] \in \mathcal{T}_\Box(\mathcal{F}, \mathcal{V}),\ c \models d\sigma \}$ と定義する．

命題 6.1 R を制約付き TRS，$\succ$ を簡約化順序とする．このとき，以下のすべては等価である．

(1) $\to_R \subseteq \succ$.

(2) 任意の規則 $l \to r \Leftarrow c \in R$，$c\theta$ が真である任

⋆2 “$\forall \vec{x}.(\neg c \vee d)$” は c が満たされるときには必ず d も満たされることを意味する．c が充足不能のときには，制約付き書換えが停止しないことがあり，完備化手続きの 1 ステップが停止しなくなるという問題が生じる．

意の代入 θ について，$l\theta \succ r\theta$.

(3) 任意の制約 c について，$s \underset{c}{\rightarrow}_R t$ ならば，$c\theta$ が真である任意の代入 θ について $s\theta \succ t\theta$.

［証明］ 付録 A.4 を参照. □

この命題より，$\rightarrow_R$ が停止性を持つとき，$\underset{c}{\rightarrow}_R$ も停止性を持つことが保証される．

6.2 制約付き TRS 完備化の推論規則

まず，制約付き TRS の完備化のための推論規則を与える．規則や等式の制約部の考慮以外は，基本的には TRS の場合[1),2)] と同様である．等式では左辺と右辺を区別する必要がない．よって，$s \simeq t \Leftarrow c$ と書いたときは，$s \approx t \Leftarrow c$ と $t \approx s \Leftarrow c$ のどちらかを表すこととする．

図 7 で与える推論規則は，等式の有限集合 E と書換え規則の有限集合 R の組 (E, R) に作用する．直観的に，E には入力の等式（の両辺を書換えた等式）か，まだ書換え規則へと変換していない危険対（の両辺を書換えた等式）が含まれる．また，R は完備性を持つ書換え規則の集合を仮定する．簡約化順序 $\succ$ は，R の書換えと簡約化順序が矛盾しない，すなわち，$\rightarrow_R \subseteq \succ$ を満たすように与える．目標は (E_0, R) から $E_0 \cup R$ に等価である完備な制約付き TRS R' を含む $(\emptyset, R')$ を導出することである．しかし，完備化手続きは一般に（成功して）停止するとは限らない．なお，TRS の完備化では内側で他の規則が必ず適用できる規則を除去する推論規則として Collapse がある[1)]．推論規則 Collapse は制約付き TRS のために拡張されている[25)] が，議論および証明が複雑になる上に，この推論規則は実装の際の効率化に有効であるものの帰納的定理の証明には本質的には影響がないことから，本論文では割愛する．

図 7 の推論規則の説明をする．Expansion[⋆1] は E から等式を一つ選び，方向付けして R に加え，R との間にできる危険対を E に加える．Composition，Simplification はそれぞれ規則の右辺，等式の一辺をその制約部が成り立つ範囲で書き換える．Deletion は両辺が等しい等式や制約部が充足不能な等式を取り除く．図 7 の推論規則を (E, R) に 1 回適用して (E', R') が得られたとき，$(E, R) \vdash_{\mathscr{C}} (E', R')$ と書く．与える等式集合と P 文付き TRS が制約を持たない場合，図 7 の推論規則は TRS の KB 完備化手続きの推論規則と一致する．

完備化の推論規則の性質を議論するために等式集合 E で定まる関係 $\leftrightarrow_E$ を定義する必要がある．$s \simeq t \Leftarrow c$ は一般には $\mathrm{fv}(c) \subseteq \mathcal{V}ar(s, t)$ を満たさないので，書換え関係と同様には定義できない．本論文では関係 $\leftrightarrow_E$ を $\{ (C[s\sigma], C[t\sigma]) \mid s \simeq t \Leftarrow c \in R, C[\,] \in \mathcal{T}_{\Box}(\mathcal{F}, \mathcal{V}), c\sigma \text{ は真} \}$ と定義する[⋆2].

6.3 制約部の分解のための推論規則の導入

図 7 の規則だけでは等式の制約をうまく場合分けできず，検証が成功しない（等式集合を空にできない）場合がある．これは TRS の完備化では直面しない，制約付きの場合固有の問題である．以下では，そのような場合を例を挙げて説明し，さらにそれを解決するための推論規則（図 8）を提案する．

P 文付き TRS $R_0 = \{ \mathsf{f}(x, y) \to \mathsf{0} \Leftarrow x \mathsf{==} y,\ \mathsf{f}(x, y) \to \mathsf{f}(\mathsf{s}(x), y) \Leftarrow x \mathsf{<} y \}$ と等式集合 $E_0 = \{ \mathsf{f}(x, y) \approx \mathsf{0} \Leftarrow x \mathsf{<=} y \}$ を考える．これらに Expansion を適用すると，$R_1 = R_0 \cup \{ \mathsf{f}(x, y) \to \mathsf{0} \Leftarrow x \mathsf{<=} y \}$ と $E_1 = \{ \mathsf{f}(\mathsf{s}(x), y) \approx \mathsf{0} \Leftarrow x \mathsf{+} \mathsf{1} \mathsf{<=} y \}$ が得られる．この後も Expansion のみが適用でき，$R_n = R_0 \cup \{ \mathsf{f}(\mathsf{s}^i(x), y) \to \mathsf{0} \Leftarrow x \mathsf{+} i \mathsf{<=} y \mid 1 \le i < n \}$ と $E_n = \{ \mathsf{f}(\mathsf{s}^n(x), y) \to \mathsf{0} \Leftarrow x \mathsf{+} n \mathsf{<=} y \}$ を生成し続ける．しかし，E_1 の等式を考えると，任意の基底代入を両辺に適用すると，両辺とも 0 に書き換えられるので，E_1 の等式は帰納的定理といえる．これがうまく証明できない理由は E_1 の等式の制約 $x \mathsf{+} \mathsf{1} \mathsf{<=} y$ が真になるときに複数の書換え規則が適用できるために，等式を正規化できないことが原因である．E_1 の等式を $x \mathsf{+} \mathsf{1} \mathsf{==} y$ の場合と $x \mathsf{+} \mathsf{1} \mathsf{<} y$ の場合に分けるとそれぞれの等式が正規化できて，$\mathsf{0} \approx \mathsf{0} \Leftarrow \cdots$ になる．このような等式の制約部を分解するための推論規則が，図 8 の推論規則 E-Decomposition である．(E_1, R_1) に R_1 の最初の規則に注目して E-Decomposition を適用すると，$E_1' = \{ \mathsf{f}(\mathsf{s}(x), y) \approx \mathsf{0} \Leftarrow x \mathsf{+} \mathsf{1} \mathsf{<=} y \wedge \neg(x \mathsf{+} \mathsf{1} \mathsf{==} y),\ \mathsf{0} \approx \mathsf{0} \Leftarrow x \mathsf{+} \mathsf{1} \mathsf{<=} y \wedge x \mathsf{+} \mathsf{1} \mathsf{==} y \}$ が得られる．(E_1', R_1) に Simplification と Deletion を適用すると，$(\emptyset, R_1)$ となる．このように，推論規則 E-Decomposition により等式を規則が適用できる制約で場合分けすることが可能になる．

次に P 文付き TRS での問題について説明する．P

⋆1 Expansion の付帯条件の一つ “$\mathtt{root}(s) \notin \mathcal{F}_{\mathcal{P}}$” は理論的には “$s$ は $\mathcal{F}_{\mathcal{P}}$ 以外の記号を含む（$s \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \setminus \mathcal{T}(\mathcal{F}_{\mathcal{P}}, \mathcal{V})$）” に緩められる．

⋆2 $\mathrm{fv}(c) \setminus \mathcal{V}ar(s, t)$ の変数に代入される項は $c\sigma$ を真にする項が一つでも存在していれば，$C[s\sigma]$ と $C[t\sigma]$ は関係付けられる．$E_{\exists}$ を $\{ s \approx t \Leftarrow \exists x_1 \cdots \exists x_n . c \mid s \simeq t \Leftarrow c \in E, \{x_1, \ldots, x_n\} \subseteq (\mathrm{fv}(c) \setminus \mathcal{V}ar(s, t)) \}$ とすると，$\leftrightarrow_E = \leftrightarrow_{E_{\exists}}$ が成り立つ．よって，等式の制約部に現れて等式自体には現れない自由変数は $\exists$ で束縛して評価されるべきである．

EXPANSION

$$\frac{(E \cup \{s \simeq t \Leftarrow c\}, R)}{(E \cup CP(s \to t \Leftarrow c, R \cup \{s \to t \Leftarrow c\}), R \cup \{s \to t \Leftarrow c\})}$$

ただし，$s \succ t$ かつ $\mathtt{root}(s) \notin \mathcal{F}_{\mathcal{P}}$ かつ $\mathrm{fv}(c) \subseteq \mathcal{V}ar(s)$

COMPOSITION

$$\frac{(E, R \cup \{s \to t \Leftarrow c\})}{(E, R \cup \{s \to u \Leftarrow c\})}$$

ただし，$t \underset{c}{\to}_R u$

SIMPLIFICATION

$$\frac{(E \cup \{s \simeq t \Leftarrow c\}, R)}{(E \cup \{s \approx u \Leftarrow c\}, R)}$$

ただし，$t \underset{c}{\to}_R u$

DELETION

$$\frac{(E \cup \{s \simeq t \Leftarrow c\}, R)}{(E, R)}$$

ただし，$s \equiv t$ または c は充足不能

図 7 制約付き TRS 完備化のための推論規則
Fig. 7 Inference rules of completion for constraint TRSs.

E-DECOMPOSITION

$$\frac{(E \cup \{s \simeq C[t\sigma] \Leftarrow c\}, R)}{(E \cup \{s \approx C[u\sigma] \Leftarrow c \wedge d\sigma, s \approx C[t\sigma] \Leftarrow c \wedge \neg(d\sigma)\}, R)}$$

ただし，$t \to u \Leftarrow d \in R$ かつ $c \models d\sigma$ かつ $\mathrm{fv}(d\sigma) \subseteq \mathrm{fv}(c)$

MGU-DELETION（P 文付き TRS の場合のみ）

$$\frac{(E \cup \{s \simeq t \Leftarrow c\}, R)}{(E \cup \{s \approx t \Leftarrow c \wedge \neg\left(\bigwedge_{x \in \mathcal{D}om(\sigma)}(x \ \mathtt{==}\ \mathbb{E}^{-1}(x\sigma))\right)\}, R)}$$

ただし，σ は s と t の最汎単一化子であり，$\mathcal{D}om(\sigma) \subseteq \mathrm{fv}(c)$ かつ $\mathcal{R}an(\sigma) \subseteq \mathcal{T}(\mathcal{F}_{\mathcal{P}bgr}, \mathrm{fv}(c))$

図 8 推論規則 E-DECOMPOSITION，MGU-DELETION
Fig. 8 Added inference rules E-DECOMPOSITION and MGU-DELETION.

文付き TRS の場合には，$x \approx \mathsf{s}(y) \Leftarrow x \mathtt{<=} y+1 \wedge x \mathtt{>} y$ のような等式は制約部から $x \mathtt{==} y+1$ が導かれ，その結果，両辺が等しいことがわかる．しかし，これを機械的に行うことは難しい．このような等式を取り除くために，図 8 の推論規則 MGU-DELETION を導入する．この推論規則では，等式の最汎単一化子を求め，その最汎単一化子が論理式に変換できるようであれば，その論理式の否定を制約部に追加することで，両辺の構造が等しくない場合に限定した等式へと変形する．この後に DELETION を適用することで，前述のような等式は除去できる．前述の等式は $x \approx \mathsf{s}(y) \Leftarrow (x \mathtt{<=} y+1 \wedge x \mathtt{>} y) \wedge \neg(x \mathtt{==} y+1)$ に変形され，制約部が充足不能になるので削除される．

E-DECOMPOSITION と MGU-DELETION は同じ等式に繰り返し使うことが出来る．しかし，E-DECOMPOSITION は等式中の一つの位置に対して規則毎に高々 1 回適用すれば，その候補は高々有限個しか存在しない．また，MGU-DELETION も一つの等式に高々 1 回適用すれば十分である．

図 8 の推論規則を 1 回適用することを $\vdash_{\mathscr{C}'}$ と記述し，図 7 もしくは図 8 の推論規則を 1 回適用すること，すなわち，$\vdash_{\mathscr{C}} \cup \vdash_{\mathscr{C}'}$ を $\vdash_{\mathscr{C},\mathscr{C}'}$ と書く．これらの推論規則の有効性は 7 節の検証例でも示されている．

6.4 完備化の正当性

本小節では，完備化手続きによって得られる制約付き TRS が合流性を持ち，さらに入力の等式と制約付き TRS と等価であることを示す．

TRS の完備化では成功した場合には合流性を持つ等価な TRS が得られるが，制約付き TRS の枠組では合流性を持つ TRS が得られるとは限らない．これは E-DECOMPOSITION と MGU-DELETION という等式の制約部を分解する推論規則を導入したことが原因である．しかし，これらの推論規則は基底項上の等価関係は保存する．また，潜在帰納法により検証を行う際には，基底項に関してのみ合流性があればよい．危険対定理は，基底項上のみでも成り立つので，完備化は帰納的定理の証明に十分に役立つ．

定義 6.2 R を制約付き TRS とする．抽象書換え系 $(\mathcal{T}(\mathcal{F}), \to_R)$ が局所合流性を持つとき，R は**基底項上で局所合流性を持つ**という．また，R の制約付き危険対 $\langle s, t\rangle \Leftarrow c$ が $c\theta_g$ を満たす任意の基底代入 θ_g について $s\theta_g$ と $t\theta_g$ が抽象書換え系 $(\mathcal{T}(\mathcal{F}), \to_R)$ で会同関係にあるとき，$\langle s, t\rangle \Leftarrow c$ は**基底項上で会同関係にある**という． □

定理 6.3 R を制約部安定性を持つ制約付き TRS とする．R が基底項上で局所合流性を持つとき，かつそのときに限り，R から作られるすべての可能な R の制約付き危険対は基底項上で会同関係にある． □

最後に，本節で提案した推論規則（図 7, 8）から構成される完備化手続きの正当性を示す．

定理 6.4 E_0 を等式の集合，$\succ$ を簡約化順序，R_0 を $\to_{R_0} \subseteq \succ$ を満たし基底項上で完備性を持つ制約付き TRS とし，E_0 のすべての等式の制約部と R_0 のすべての規則の制約部が $\xrightarrow{*}_{R_0}$ の下で真偽安定性を持ち，$(E_0, R_0) \vdash_{\mathscr{C},\mathscr{C}'} \cdots \vdash_{\mathscr{C},\mathscr{C}'} (\emptyset, R_n)$ を得たとする．このとき，以下のすべてが成り立つ．

(1) R_n は $E_0 \cup R_0$ と基底項上で等価（$\overset{*}{\leftrightarrow}_{E_0 \cup R_0} = \overset{*}{\leftrightarrow}_{R_n}$）である．
(2) R_n は真偽安定性を持つ．
(3) R_n は基底項上で完備である．
(4) $\mathcal{T}(\mathcal{C}_{R_0}, \mathcal{V}) = \mathcal{T}(\mathcal{C}_{R_n}, \mathcal{V})$ $(\mathcal{C}_{R_0} = \mathcal{C}_{R_n})$．

［証明］ 付録 A.5 を参照． □

なお，$\vdash_{\mathscr{C}'}$ が出現しなければ，“基底項上で” という制限は不要である．

図 7, 8 の推論規則をどのように適用していくかを定めた戦略の下で，等式集合に規則を適用して等式集合を空にすることをめざす手続きが**完備化手続き**である．定理 6.4 より，図 7, 8 の推論規則で構成される任意の完備化手続きは正しいといえる．

7. 手続き型プログラムの検証手続き

本節では，本論文で提案する手続き型プログラムの検証手順を説明する．検証する性質は 3 節で提案した手続き型言語で記述されたプログラムと P 文付き TRS で記述された仕様を表す関数の等価性とする．等価性判定には 7.2 小節で紹介する潜在帰納法の原理とそれに基づく証明法を利用する．そして，本手法による検証例を挙げる．

7.1 検証対象プログラムの制限

検証対象のプログラムは以下を満たすと仮定する．

- オーバーフローは起こらない．
- プログラムは停止性を持ち，さらに $\mathbb{TRS}$ で得られた P 文付き TRS も停止性を持つ．

一般には，停止性を持つプログラムから変換 $\mathbb{TRS}$ で得られた P 文付き TRS が停止性を持つとは限らない．例えば，図 4 の while 文の制約を `i != n+1` に変更した手続き型プログラムは n を自然数に限定した際には停止性を持つが，変換して得られる P 文付き TRS は停止性を持たないだけでなく，弱停止性でさえ持たない．例えば，$\mathsf{u}_2(\mathsf{s}(\mathsf{0}), \mathsf{s}^2(\mathsf{0}), \mathsf{0})$ は正規形を持たない．

7.2 潜在帰納法の原理と帰納的定理の証明

本小節では，等価性判定に利用する潜在帰納法[3),6),12),15)] の原理とそれに基づく既存の定理自動証明[14)] の概要を説明する．

等式 $s \approx t$ が項書換え系 R における**帰納的定理**であるとは，任意の基底代入 σ_g について $s\sigma_g \overset{*}{\leftrightarrow}_R t\sigma_g$ となることである[6)]．また，**制約付き等式** $s \approx t \Leftarrow c$ が制約付き TRS R における帰納的定理であるとは，$c\sigma_g$ は真である任意の基底代入 σ_g について $s\sigma_g \overset{*}{\leftrightarrow}_R t\sigma_g$ となることである．本論文では，制約付き等式を単に等式という場合もある．

2 つの抽象書換え系 $S_1 = (A, \to_1)$，$S_2 = (A, \to_2)$ が $\overset{*}{\leftrightarrow}_1 = \overset{*}{\leftrightarrow}_2$ を満たすとき，S_1 と S_2 は**等価**であるという．**潜在帰納法**とは適当な集合上における 2 つの抽象書換え系の等価性を判定する手法である．

定理 7.1 (潜在帰納法[15),21)]) 以下のすべてを満たすとき，$\overset{*}{\leftrightarrow}_1 = \overset{*}{\leftrightarrow}_2$．

- $\overset{*}{\leftrightarrow}_1 \subseteq \overset{*}{\leftrightarrow}_2$．
- S_1 が弱正規性を持つ．
- S_2 が合流性を持つ．
- $NF_{S_1} = NF_{S_2}$． □

この原理を用いると，等式集合 E の各等式が R の帰納的定理であることの証明は以下の手順で行える[14)]

(1) $\to_R \subseteq \succ$ となる簡約化順序 $\succ$ を与える[⋆1]．
(2) (E, R) を $\succ$ の下で完備化．
(3) 完備化が成功して得られた制約付き TRS R' が $NF_R(\mathcal{F}) = NF_{R'}(\mathcal{F})$ を満たすか判定．

7.3 小節では，上記の手法を制約付き TRS に拡張する．

文献[14)] の証明法では，完備化で規則を追加する毎に $NF_R(\mathcal{F})$ が変化しないかを判定する．本手法では，

⋆1 本論文では，Expansion での等式の規則化の際に R の停止性を保持するように，人間が適当に等式を方向付けする．

関数は必ず結果を返すという性質（十分完全性）を利用し，$NF_R(\mathcal{F})$ が変化しないように規則を追加することで，(3) の判定は行わなくてよいようにする．

潜在帰納法に基づいて帰納的定理を証明する際には，正規形の集合が変化してはいけない．すなわち，$(E, R) \vdash_{\mathscr{C}} (E', R')$ のとき，$NF_R(\mathcal{F}) = NF_{R'}(\mathcal{F})$ を満たさなければならない．反駁証明法と組み合わせた手法[14] では Expansion の後に毎回 $NF_R(\mathcal{F}) = NF_{R'}(\mathcal{F})$ を判定する．本手法では，制約付き TRS を扱っているので，正規形の集合が一般には計算可能ではない．しかし，本論文で扱う手続き型プログラムは十分完全性を持つので，関数の十分完全性を利用することで，$NF_R(\mathcal{F}) = NF_{R'}(\mathcal{F})$ が常に成り立つことを示す．これは，真偽安定性を保つために Expansion に付加した制約 $\mathtt{root}(s) \notin \mathcal{F}_{\mathcal{P}}$ の影響である．

まずは，制約付き TRS の十分完全性を TRS と同様に定義する．

定義 7.2 R を制約付き TRS とする．すべての基底項 $s \in \mathcal{T}(\mathcal{F})$ に対して $s \xrightarrow{*}_R t \in \mathcal{T}(\mathcal{C}_R)$ を満たす t が存在するとき，R は**十分完全性**を持つという． □

命題 7.3 制約付き TRS R が十分完全性を持つとき，かつそのときに限り，$NF_R(\mathcal{F}) = \mathcal{T}(\mathcal{C}_R)$． □

補題 7.4 $(E, R) \vdash_{\mathscr{C},\mathscr{C}'} (E', R')$ とする．このとき，以下のすべてが成り立つ．

- $NF_R(\mathcal{F}) = NF_{R'}(\mathcal{F})$．
- R が十分完全性を持つならば，R' も十分完全性を持つ．

［証明］ Expansion のときのみ示せばよい．命題 7.3 より，$NF_R(\mathcal{F}) = \mathcal{T}(\mathcal{C}_R)$．$\mathtt{root}(s) \notin \mathcal{F}_{\mathcal{P}}$ より，$\mathcal{C}_R = \mathcal{C}_{R'}$．$\mathcal{T}(\mathcal{C}_{R'})$ の項は正規形であるので，$\mathcal{T}(\mathcal{C}_{R'}) \subseteq NF_{R'}(\mathcal{F})$．一方，$\xrightarrow{+}_R \subseteq \xrightarrow{+}_{R'}$ より，$NF_R(\mathcal{F}) \supseteq NF_{R'}(\mathcal{F})$．よって，$NF_{R'}(\mathcal{F}) = \mathcal{T}(\mathcal{C}_{R'})$．命題 7.3 より，$R'$ は十分完全性を持つ． □

次に，変換で得られる P 文付き TRS 停止性を持つときには，その P 文付き TRS は十分完全性を持つことを示す．

命題 7.5 P を手続き型プログラムとする．このとき，TRS $\mathbb{TR}(P)$ は左線形である．さらに，$\mathbb{TR}(P) \cup R_c$ が弱停止性を持つならば，$\mathbb{TR}(P) \cup R_c$ は十分完全性を持つ．

［証明］ 付録 A.6 を参照． □

よって，変換 $\mathbb{TR}$ で得られる P 文付き TRS が停止性を持つときは十分完全性も持つので，完備化手続きが成功すれば検証が成功したことになる．

なお，TRS での検証では，異なる 2 つの構成子項からなる等式が出現した時点で最初に与えた等式は帰納的定理ではないことが保証できる（反駁証明）．提案した完備化においても同様のことが言える．例えば，P 文付き TRS での完備化の途中に等式 $\mathsf{s}(x) \approx \mathsf{0} \Leftarrow x \mathsf{<=} \mathsf{0}$ などが出現したとする．最初に与えられた等式が帰納的定理であることを示すには，この等式が成り立たないといけない．しかし，これは自然数の意味を考えると成り立たない．よって，最初に与えた等式が帰納的定理ではないことが示される．

7.3 手続き型プログラムの検証手続き

本論文で提案する検証手続きは図 9 のようにまとめられる．図 9（4）(a)–(k) は制約付き書換え上の完備化の戦略の一つである．これは，TRS 上の KB 完備化手続きの自然な拡張である．（3）の補題の等式は，（4）の繰り返しの途中 (a) の直前で必要に応じて追加しても構わない．（4）では等式 $f(x_1, \ldots, x_n) = f'(x_1, \ldots, x_n)$ が $R_p \cup R_s$ 上で帰納的定理であることを潜在帰納法により証明している．

定理 7.6 R を基底項に関して完備性を持つ P 文付き TRS，E を等式集合，$\succ$ を $\rightarrow_R \subseteq \succ$ を満たす簡約化順序とする．(E, R) から $\succ$ の下で，図 9（4）の完備化手続きが成功したならば，E のすべての等式は R の帰納的定理である．

［証明］ 付録 A.7 を参照． □

定理 7.6 より，本手法で検証が成功した際に，手続き型プログラムが仕様を満たすことが保証される．

系 7.7 n 引数関数 $\mathtt{f}$ を定義する手続き型プログラム P，仕様の pdTRS（関数 $g(x_1, \ldots, x_n)$ を定義）を R_s とし，$R_s \supseteq R_c$ とする．さらに，$\mathbb{TR}(P) \cup R_c \cup R_s$ は完備性を持つとする[⋆1]．図 9 が成功したとき，任意の自然数 $v_1, \ldots, v_n$ について，手続き型関数 $f(v_1, \ldots, v_n)$ が返す値と項 $g(\mathsf{s}^{v_1}(\mathsf{0}), \ldots, \mathsf{s}^{v_n}(\mathsf{0}))$ の R_s での正規形は（$\mathbb{E}$ を法として）等価である． □

なお，手続き型プログラム P_1 と P_2 の等価性につい

⋆1 $\mathbb{TR}(P) \cup R_c \cup R_s$ は停止性を持ち，R_s は合流性を持ち，$\mathbb{TR}(P)$ と R_s の被定義記号に重複はない状況を仮定する．

n 引数関数 f を定義する手続き型プログラムを P，P 文付き TRS（関数 $g(x_1,\ldots,x_n)$ を定義）を R_s とする．
(1) $R_p = \mathbb{TR}(P)$ を求め，P 文付き TRS R_c を定める．
(2) $\rightarrow_{R_p \cup R_c \cup R_s} \subseteq \succ$ を満たす簡約化順序 $\succ$ を定める．
(3) $E = \{f(x_1,\ldots,x_n) \approx g(x_1,\ldots,x_n)\} \cup \{$ 補題の等式 $\}$ とする．
(4) $R_0 := R_p$，$E_0 := E$，$i := 0$ として，以下の完備化手続きを $E_i = \emptyset$ となるまで繰り返す．
  (a) Expansion を適用．$E_i \neq \emptyset$ で順序の付く等式が存在しないとき失敗で終了．
  (b) Composition で R_i のすべての規則の右辺を正規化．
  (c) Simplification で E_i のすべての等式の両辺を正規化．
  (d) MGU-Deletion をすべての等式に適用．
  (e) 制約が変化した等式の両辺を Simplification でを正規化．
  (f) Deletion で E_i の自明か不能な等式をすべて除去．
  (g) 各等式について 1 回ずつ E-Decomposition を適用．
  (h) Simplification で E_i のすべての等式の両辺を正規化．
  (i) MGU-Deletion をすべての等式に適用．
  (j) 制約が変化した等式の両辺を Simplification で正規化．
  (k) Deletion で E_i の自明か不能な等式をすべて除去．
  (l) $R_{i+1} := R_i$，$E_{i+1} := E_i$，$i := i+1$．

図 9 潜在帰納法に基づいた検証手続き
Fig. 9 Verification procedure based on inductionless induction.

ても $\mathbb{TR}$ で得られる P 文付き TRS の u 記号の重複がないように $\mathbb{TR}$ を適用すれば，上記と同様に行える．

7.4 検 証 例

本小節では，手続き型で書かれた関数 sum1（図 4）と仕様として P 文付き TRS で書いた関数 sum（図 10 の R_{sum}）を与え，それらの等価性を検証する．手続き型プログラム sum1 を P 文付き TRS に変換したものが例 4.1 の $R_{\mathtt{sum1}}$ である．$R_{\mathtt{sum1}}$ に Composition を施して簡単化した P 文付き TRS $R'_{\mathtt{sum1}}$ を図 10 に示す．$R_0 = R'_{\mathtt{sum1}} \cup R_{\mathsf{sum}} \cup R_c$ 上で等式 $\mathtt{sum1}(x) \approx \mathsf{sum}(x)$ が帰納的定理であることは図 9 の手続きにより図 10 のように証明できる．$\Rightarrow_{\mathrm{X}}$ で図 9（4）の適用の様子を X にどの段階までを適用したかの情報を付加して表示する．P 文の判定の際には，どの変数についても 0 以上であるとみなして真偽判定を行った．なお，検証には補題等式 $\mathtt{u}_2(\mathsf{s}(n),i,z) \approx \mathtt{u}_2(n,i,z) + \mathsf{s}(n) \Leftarrow i \mathtt{<=} n$ を導入することが必要であったため，補題等式から規則化し証明した．この補題等式は，$R_0 \cup \{(7)\}$ の危険対を R_0 のみで Simplification して導かれる等式 $\mathtt{u}_2(n,\mathsf{s}(\mathsf{0}),\mathsf{0}) + \mathsf{s}(n) \approx \mathtt{u}_2(\mathsf{s}(n),\mathsf{s}(\mathsf{0}),\mathsf{0})$ から，定数項を変数で置き換えて補題等式を生成することにより得た．また，よく使われるインクリメント演算子（++）に対して項が簡潔になるように，$x+1$, $1+y$ を計算する規則も用いた．完備化が成功して終了したので，関数 sum1 と関数 sum の等価性が保証され，プログラム sum1 が P 文付き TRS で与えられた仕様 R_{sum} を満たすことが示された．

P 文付き TRS と等式の制約部を場合分けする E-Decomposition を導入したことで，完備化の流れが，sum1 と sum の等価性を人間が机上で証明する様子に非常に類似したことが図 10 からわかる．

8. 関 連 研 究

9. お わ り に

本論文では，TRS 上の危険対定理および完備化手続きを制約付き TRS 上へ拡張することで，制約付き TRS 上での帰納的定理の証明法を提案した．そして，自然数上の手続き型プログラムから P 文付き TRS への等価変換を与えることで，手続き型プログラムの検証法の枠組を提案した．本手法で対象とする手続き型言語は配列、ポインタが扱えないなど制限が強いが，数値を計算するための関数などは記述できる．そのような場合には，本手法で検証を行える．

一方，本論文で示した検証例では，完備化手続きで必要な簡約化順序は人間が適当に方向付けを行うことで代用した．さらに，E-Decomposition の適用に関しても本手法では決定的ではない．よって，本手法は現在のところ，人間が対話的に推論規則の適用を選択しなければならない．これを自動化するために，以下の手法を開発することが今後の課題である．

- 制約付き等式の方向付け，すなわち，制約付き

規則集合

$$R'_{\mathsf{sum1}} = \left\{ \begin{array}{rcll} \mathsf{sum1}(n) & \to & \mathsf{u}_2(n, \mathsf{s}(\mathsf{0}), \mathsf{0}) & (1) \\ \mathsf{u}_2(n,i,z) & \to & \mathsf{u}_2(n, \mathsf{s}(i), z+i) \Leftarrow i \mathsf{<=} n & (2) \\ \mathsf{u}_2(n,i,z) & \to & z \Leftarrow \neg(i \mathsf{<=} n) & (3) \end{array} \right. \quad R_{\mathsf{sum}} = \left\{ \begin{array}{rcll} \mathsf{sum}(\mathsf{0}) & \to & \mathsf{0} & (4) \\ \mathsf{sum}(\mathsf{s}(n)) & \to & \mathsf{sum}(n) + \mathsf{s}(n) & (5) \end{array} \right.$$

$$R_c = \left\{ \begin{array}{ll} \mathsf{0} + y \to y, & \mathsf{s}(x) + y \to \mathsf{s}(x+y), \\ \mathsf{s}(\mathsf{0}) + y \to \mathsf{s}(y), & x + \mathsf{s}(\mathsf{0}) \to \mathsf{s}(x) \end{array} \right\} \quad R_0 = R'_{\mathsf{sum1}} \cup R_{\mathsf{sum}} \cup R_c$$

等式集合

$$\left\{ \begin{array}{rcll} \mathsf{u}_2(\mathsf{s}(n), i, z) & \approx & \mathsf{u}_2(n,i,z) + \mathsf{s}(n) \Leftarrow i \mathsf{<=} n+1 & (6) \\ \mathsf{sum}(n) & \approx & \mathsf{sum1}(n) & (7) \end{array} \right.$$

完備化の様子

$(\{\ (6),\ (7)\ \},\ R_0)$

$\Rightarrow_{(a)}$

$$(\left\{ \begin{array}{rl} & (7), \\ (8)\ \mathsf{u}_2(\mathsf{s}(n), \mathsf{s}(i), z+i) \approx & \mathsf{u}_2(n,i,z) + \mathsf{s}(n) \Leftarrow i \mathsf{<=} n+1 \wedge i \mathsf{<=} n+1, \\ (9) \quad z \approx & \mathsf{u}_2(n,i,z) + \mathsf{s}(n) \Leftarrow i \mathsf{<=} n+1 \wedge \neg(i \mathsf{<=} n+1) \end{array} \right\},\ R_0 \cup \{(6)\ \mathsf{u}_2(\mathsf{s}(n),i,z) \to \mathsf{u}_2(n,i,z) + \mathsf{s}(n) \Leftarrow i \mathsf{<=} n+1\})$$

$\Rightarrow_{(b)-(f)}$

$$(\{\ (7')\ \mathsf{sum}(n) \approx \mathsf{u}_2(n, \mathsf{s}(\mathsf{0}), \mathsf{0}),\ (8)\ \},\ R_0 \cup \{\ (6)\ \})$$

$\Rightarrow_{(g)}$

$$(\left\{ \begin{array}{rl} (7'\text{-}1) & \mathsf{sum}(n) \approx \mathsf{u}_2(n, \mathsf{s}^2(\mathsf{0}), \mathsf{0} + \mathsf{s}(\mathsf{0})) \Leftarrow \mathsf{1} \mathsf{<=} n, \\ (7'\text{-}2) & \mathsf{sum}(n) \approx \mathsf{u}_2(n, \mathsf{s}(\mathsf{0}), \mathsf{0}) \Leftarrow \neg(\mathsf{1} \mathsf{<=} n), \\ (8\text{-}1) & \mathsf{u}_2(\mathsf{s}(n), \mathsf{s}(i), z+i) \approx \mathsf{u}_2(n,i,z) + \mathsf{s}(n) \Leftarrow (i \mathsf{<=} n+1 \wedge i \mathsf{<=} n+1) \wedge \neg(i+1 \mathsf{<=} n+1), \\ (8\text{-}2) & \mathsf{u}_2(n, \mathsf{s}(i), z+i) + \mathsf{s}(n) \approx \mathsf{u}_2(n,i,z) + \mathsf{s}(n) \Leftarrow (i \mathsf{<=} n+1 \wedge i \mathsf{<=} n+1) \wedge i+1 \mathsf{<=} n+1 \end{array} \right\},\ R_0 \cup \{\ (6)\ \})$$

$\Rightarrow_{(h)}$

$$(\left\{ \begin{array}{rl} (7'\text{-}1') & \mathsf{sum}(n) \approx \mathsf{u}_2(n, \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) \Leftarrow \mathsf{1} \mathsf{<=} n, \\ (7'\text{-}2') & \mathsf{sum}(n) \approx \mathsf{0} \Leftarrow \neg(\mathsf{1} \mathsf{<=} n), \\ (8\text{-}1') & z+i \approx z + \mathsf{s}(n) \Leftarrow (i \mathsf{<=} n+1 \wedge i \mathsf{<=} n+1) \wedge \neg(i+1 \mathsf{<=} n+1), \\ (8\text{-}2') & \mathsf{u}_2(n, \mathsf{s}(i), z+i) + \mathsf{s}(n) \approx \mathsf{u}_2(n, \mathsf{s}(i), z+i) + \mathsf{s}(n) \Leftarrow (i \mathsf{<=} n+1 \wedge i \mathsf{<=} n+1) \wedge i+1 \mathsf{<=} n+1 \end{array} \right\},\ R_0 \cup \{\ (6)\ \})$$

$\Rightarrow_{(i)}$

$$(\left\{ \begin{array}{rl} & (7'\text{-}1'),\quad (7'\text{-}2'),\quad (8\text{-}2') \\ (8\text{-}1'')\ z+i \approx & z + \mathsf{s}(n) \Leftarrow ((i \mathsf{<=} n+1 \wedge i \mathsf{<=} n+1) \wedge \neg(i+1 \mathsf{<=} n+1)) \wedge \neg(i \mathsf{==} n) \end{array} \right\},\ R_0 \cup \{\ (6)\ \})$$

$\Rightarrow_{(j)-(l)}$

$$(\{\ (7'\text{-}1'),\ (7'\text{-}2')\ \},\ R_0 \cup \{\ (6)\ \})$$

$\Rightarrow_{(a)}$

$$(\left\{ \begin{array}{rl} & (7'\text{-}2'), \\ (10) & \mathsf{0} \approx \mathsf{u}_2(\mathsf{0}, \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) \Leftarrow \mathsf{1} \mathsf{<=} \mathsf{0}, \\ (11)\ \mathsf{sum}(n) + \mathsf{s}(n) & \approx \mathsf{u}_2(\mathsf{s}(n), \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) \Leftarrow \mathsf{1} \mathsf{<=} n+1 \end{array} \right\},\ R_0 \cup \left\{(6),\ (7'\text{-}1')\ \mathsf{sum}(n) \to \mathsf{u}_2(n, \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) \Leftarrow \mathsf{1} \mathsf{<=} n\right\})$$

$\Rightarrow_{(b)-(f)}$

$$(\{\ (7'\text{-}2'),\ (11)\ \},\ R_0 \cup \{\ (6),\ (7'\text{-}1')\ \})$$

$\Rightarrow_{(g)}$

$$(\left\{ \begin{array}{rl} & (7'\text{-}2'), \\ (11\text{-}1)\ \mathsf{u}_2(n, \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) + \mathsf{s}(n) & \approx \mathsf{u}_2(\mathsf{s}(n), \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) \Leftarrow \mathsf{1} \mathsf{<=} n+1 \wedge \mathsf{1} \mathsf{<=} n, \\ (11\text{-}2) & \mathsf{sum}(n) + \mathsf{s}(n) \approx \mathsf{u}_2(\mathsf{s}(n), \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) \Leftarrow \mathsf{1} \mathsf{<=} n+1 \wedge \neg(\mathsf{1} \mathsf{<=} n) \end{array} \right\},\ R_0 \cup \{\ (6),\ (7'\text{-}1')\ \})$$

$\Rightarrow_{(h)}$

$$(\left\{ \begin{array}{rl} & (7'\text{-}2'), \\ (11\text{-}1')\ \mathsf{u}_2(n, \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) + \mathsf{s}(n) & \approx \mathsf{u}_2(n, \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) + \mathsf{s}(n) \Leftarrow \mathsf{1} \mathsf{<=} n+1 \wedge \mathsf{1} \mathsf{<=} n, \\ (11\text{-}2') & \mathsf{sum}(n) + \mathsf{s}(n) \approx \mathsf{s}(\mathsf{0}) \Leftarrow \mathsf{1} \mathsf{<=} n+1 \wedge \neg(\mathsf{1} \mathsf{<=} n) \end{array} \right\},\ R_0 \cup \{\ (6),\ (7'\text{-}1')\ \})$$

$\Rightarrow_{(i)-(l)}$

$$(\{\ (7'\text{-}2'),\ (11\text{-}2')\ \},\ R_0 \cup \{\ (6),\ (7'\text{-}1')\ \})$$

$\Rightarrow_{(a)}$ $(\{\ (11\text{-}2')\ \},\ R_0 \cup \{\ (6),\ (7'\text{-}1'),\ (7'\text{-}2')\ \mathsf{sum}(n) \to \mathsf{0} \Leftarrow \neg(\mathsf{1} \mathsf{<=} n)\ \})$

$\Rightarrow_{(b)}$ $(\{\ (11\text{-}2'),\ (12)\ \mathsf{0} \approx \mathsf{0} \Leftarrow \neg(\mathsf{1} \mathsf{<=} \mathsf{0}),\ (13)\ \mathsf{0} \approx \mathsf{sum}(n) + \mathsf{s}(n) \Leftarrow \neg(\mathsf{1} \mathsf{<=} n+1)\ \},\ R_0 \cup \{\ (6),\ (7'\text{-}1'),\ (7'\text{-}2')\ \})$

$\Rightarrow_{(c)}$ $(\{\ (11\text{-}2'')\ \mathsf{s}(n) \approx \mathsf{s}(\mathsf{0}) \Leftarrow \mathsf{1} \mathsf{<=} n+1 \wedge \neg(\mathsf{0} \mathsf{<=} n),\ (12),\ (13)\ \},\ R_0 \cup \{\ (6),\ (7'\text{-}1'),\ (7'\text{-}2')\ \})$

$\Rightarrow_{(d-f)}$ $(\{\ \},\ R_0 \cup \{\ (6),\ (7'\text{-}1'),\ (7'\text{-}2')\ \})$

図 10 $\mathsf{sum1}(x) \approx \mathsf{sum}(x)$ を証明する完備化の様子
Fig. 10 Completion steps for proving $\mathsf{sum1}(x) \approx \mathsf{sum}(x)$.

TRS の停止性を保証する簡約化順序.

- 図 9 (g) における E-Decomposition の決定的な適用のための戦略.

完備化には簡約化順序 $\succ$ が必要であるが，今回の sum 関数の検証では人間の直観にたよった順序に基づいて完備化を行った．規則の順序付けを経路順序で自動的に行うと図 10 の規則 (2) には順序が付かない．よって，制約付き等式の方向付けは手続きの自動化のため

の大きな課題である．また，本論文では，変換で得られたP文付きTRSの停止性を仮定して検証を行った．これを自動で保証するために，P文付きTRSの停止性証明法を開発することも今後の課題である．

`sum1`の検証例では補題となる等式を導入した．本手法ではループ不変式や事前条件・事後条件を与える必要がない一方で，検証を成功させるために補題を与える必要がある場合も多い．そのような補題の発見は定理自動証明の分野での古くからの課題である．本論文の変換でプログラムから得られるP文付きTRSでは，uの記号が状態を表す構造を取り制御文の特徴を反映するので，一般のTRSの場合での補題発見よりも補題の生成が容易であると期待している．手続き型プログラム検証に特化した補題の発見の自動化も今後の課題である．今回は自然数のみを扱ったが，整数の取り扱いは今後も今後の課題である．

今回は潜在帰納法に基づいた手続きを提案したが，書換え帰納法に基づいた手法など，他の検証法もP文付きTRSを用いた検証に拡張できると予想される．

制約付きTRSの枠組は文献[4),5)]の制約付きCTRSの枠組をP文を制約として扱うために拡張した書換え系であり，その違いは限量子の扱いである．本論文の手法では，制約付きTRSの危険対定理を与え，それを成り立たせる完備化の条件が十分完全性を保たせるので，完備化手続きがそのまま定理自動証明に利用できる．よって，文献[4)]の定理自動証明のための推論規則よりも完備化に特化した手法を与え，さらに部分的には単純である制約付きTRSの枠組で議論することで，問題点を明記した議論を展開した．また，十分完全性に注目した定理自動証明も文献[4),5)]で扱われており，本手法との比較は今後の課題である．

謝　辞　本研究は一部，文部科学省科学研究費#16650005，#17700009，#18500011の補助を受けている．

参 考 文 献

1) Baader, F. and Nipkow, T.: *Term Rewriting and All That*, Cambridge University Press (1998).
2) Bachmair, L.: *Canonical Equational Proofs*, Birkhauser (1991).
3) Bouhoula, A.: Automated Theorem Proving by Test Set Induction, *Journal of Symbolic Computation*, Vol.23, No.1, pp.47–77 (1997).
4) Bouhoula, A. and Jacquemard, F.: Automated Induction for Complex Data Structures, Research Report LSV-05-11, Laboratoire Spécification et Vérification, ENS Cachan, France (2005). 24 pages.
5) Bouhoula, A. and Jacquemard, F.: Automating Sufficient Completeness Check for Conditional and Constrained TRS, *Proceedings of the 20th International Workshop on Unification*, pp.108–128 (2006).
6) Boyer, R.S. and Moore, J.S.: *A Computational Logic*, Academic Press (1979).
7) Clarke, E. M. and Emerson, E. A.: Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic, *Proceedings of Logic and Programs Workshop*, Lecture Notes in Computer Science, Vol.131, Springer, pp.52–71 (1981).
8) Cooper, D.: Theorem Proving in Arithmetic without Multiplication, *Proceedings of the Seventh Annual Machine Intelligence Workshop*, Machine Intelligence, No.7, Edinburgh University Press, pp.91–99 (1972).
9) Dijkstra, E.W.: A Discipline of Programming, Prentice-Hall (1976).
10) Ganzinger, H.: A Completion Procedure for Conditional Equations, *Journal of Symbolic Computation*, Vol.11, No.1/2, pp.51–81 (1991).
11) Hoare, C.A.R.: An Axiomatic Basis for Computer Programming, *Communications of the ACM*, Vol.12, No.10, pp.576–580 (1969).
12) Huet, G.P. and Hullot, J.-M.: Proofs by Induction in Equational Theories with Constructors, *Journal of Computer and System Sciences*, Vol.25, No.2, pp.239–266 (1982).
13) Huth, M. and Ryan, M.: *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press (2000).
14) Kapur, D., Narendran, P. and Zhang, H.: Automating Inductionless Induction Using Test Sets, *Journal of Symbolic Computation*, Vol.11, No.1/2, pp.81–111 (1991).
15) Musser, D.R.: On Proving Inductive Properties of Abstract Data Types, *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages*, pp.154–162 (1980).
16) Ohlebusch, E.: *Advanced Topics in Term Rewriting*, Springer-Verlag (2002).
17) Presburger, M.: Uber die Vollstandigkeit eines gewissen Systems der Arithmetic ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt, Compte-Rendus dei Congres des Mathematiciens des Pays Slavs (1929).
18) Queille, J.-P. and Sifakis, J.: Specification and Verification of Concurrent Systems in CESAR, *Proceedings of the 5th International Sympo-*

sium on Programming, Lecture Notes in Computer Science, Vol. 137, Springer, pp. 337–351 (1982).
19) Reddy, U.S.: Term Rewriting Induction, *Proceedings of the 10th International Conference on Automated Deduction*, Lecture Notes in Computer Science, Vol.449, Springer, pp.162–177 (1990).
20) Reynolds, J. C.: *Theories of Programming Languages*, Cambridge University Press (1998).
21) Toyama, Y.: How to Prove Equivalence of Term Rewriting Systems Without Induction, *Theoretical Computer Science*, Vol. 90, No. 2, pp.369–390 (1991).
22) 伊理 正夫, 野崎 明弘, 野下 浩平（編）：計算の効率とその限界, 入門現代の数学 [13], 日本評論社 (1980).
23) 高橋 宣孝, 酒井 正彦, 外山 芳人：条件付き項書換え系の合流性について, 電子情報通信学会論文誌, Vol.J79-DI, No.11, pp.1–6（1996）.
24) 東野 輝夫, 北道 淳司, 谷口 健一：整数上の線形制約の処理と応用, コンピュータソフトウェア, Vol.9, No.6, pp.31–39 (1992).
25) 古市 祐樹, 西田 直樹, 酒井 正彦, 草刈 圭一朗, 坂部 俊樹：手続き型プログラムから書換え系への変換に基づくソフトウェア検証の試み, 信学技報 SS2006-41, Vol.106, No.324, pp.7–12 (2006).
26) M. ヘネシー（著）/ 荒木 啓二郎, 程京徳（共著）：プログラミング言語の意味論入門, Information & Computing, 第 76 巻, サイエンス社 (1993).

付　　録

A.1 定理 4.2 の略証

5 節で示すように, 生成した P 文付き TRS が停止性を持つ場合は合流性を持つ. 合流性を持つ場合はどのように計算を行っても結果は一致するので, 本論文では, P 文付き TRS の書換えは最左最内で評価されることとする.

次の補題はステートメント系列と変換してられた P 文付き TRS で同様の動作をすることを示す. なお, $\rightarrow^{p}$, $\rightarrow^{p<}$ はそれぞれ p の位置, $p < q$ を満たす位置 q での書換えを表す.

補題 A.1.1 ss をステートメントの系列, $x_1, \ldots, x_n$ を ss に現れる自由変数, $\mathbb{T}(\mathrm{u}_j(x_1, \ldots, x_n),\ \mathrm{ss},\ j+1) = (t, R, k)$ とする. このとき, 変数 $y_1, \ldots, y_m$ が存在して, $t \equiv \mathrm{u}_{k-1}(x_1, \ldots, x_n, y_1, \ldots, y_m)$. さらに, $(\mathrm{ss}, \mathrm{M}) \stackrel{*}{\Rightarrow}_C (\epsilon, \mathrm{M}')$ のとき, かつこのときに限り, $\mathrm{u}_j(x_1, \ldots, x_n)\theta_{\mathrm{M}} \stackrel{*}{\rightarrow}_{R \cup R_c} t\theta_{\mathrm{M}'}$.

［略証］ t の構造については定義より明らか. 後者の題意の右向きは, ステートメント系列 ss の構造と $\Rightarrow_C$ の推論木の高さの辞書式順序に関する帰納法により証明できる. 左向きは, ステートメント系列 ss の構造と $\rightarrow_R \stackrel{*}{\rightarrow}^{!}_{R_c}$ のステップ数 n の辞書式順序に関する帰納法により証明できる. ここで, $\stackrel{*}{\rightarrow}^{!}_{R_c} = \{ (s,t) \mid s \stackrel{*}{\rightarrow}_{R_c} t \in NF_{R_c} \}$ とする. □

上記の補題により, 変換の正当性を保証する定理 4.2 が系として得られる.

A.2 命題 5.2 の証明

まずは, 真偽安定性の性質を示す.

補題 A.2.1 真偽安定性は ¬, ∧, ∀, ∨, ∃, および代入に閉じる.

［証明］ 代入に閉じることは定義より明らかである.

∨, ∃ を含む制約は ¬, ∧, ∀ のみから成る制約で表現できるので, ¬, ∧, ∀ のみ証明する.

- ¬ の場合 $(\neg c)\sigma$ が真（偽）であると仮定すると, $c\sigma$ は偽（真）である. c の真偽安定性より $c\sigma_{\rightarrow}$ は偽（真）であるので, $(\neg c)\sigma_{\rightarrow}$ は真（偽）である.
- ∧ の場合 $(c_1 \wedge c_2)\sigma$ が真であると仮定すると, $c_1\sigma$ と $c_2\sigma$ は真である. c_1 と c_2 の真偽安定性より $c_1\sigma_{\rightarrow}$ と $c_2\sigma_{\rightarrow}$ は真であるので, $(c_1 \wedge c_2)\sigma_{\rightarrow}$ は真である. 一方, $(c_1 \wedge c_2)\sigma$ が偽であると仮定すると, $c_1\sigma$ が偽としても一般性を失わない. c_1 の真偽安定性より $c_1\sigma_{\rightarrow}$ は偽であるので, $(c_1 \wedge c_2)\sigma_{\rightarrow}$ は偽である.
- $\forall x.$ の場合 このとき, $x \notin \mathcal{D}om(\sigma)$ とする. $(\forall x.c)\sigma$ が真であると仮定する. $\sigma' = \sigma \cup \{x \mapsto t\}$ とする. $(\forall x.c)\sigma$ が真なので, $c\sigma'$ は真である. c の真偽安定性より $c\sigma'_{\rightarrow}$ は真である. 任意の t について $c\sigma'_{\rightarrow}$ が真であるので, $(\forall x.c)\sigma_{\rightarrow}$ は真である. 一方, $(\forall x.c)\sigma$ が偽であると仮定する. このとき, $c\sigma'$ が偽となるような代入 $\sigma' = \sigma \cup \{x \mapsto t\}$ が存在する. c の真偽安定性より, $c\sigma'_{\rightarrow}$ は偽である. $c\sigma'_{\rightarrow}$ が偽となる t が存在するので, $(\forall x.c)\sigma_{\rightarrow}$ は偽である. □

次に, 真偽安定性を保つような書換え規則の追加のための条件を示す.

補題 A.2.2 $l \rightarrow r \Leftarrow c$ を書換え規則, 制約 $c, d \in \mathcal{P}$ を $\stackrel{*}{\rightarrow}_R$ に関して真偽安定性を持つとする. $\mathtt{root}(l) \notin$

$\mathcal{F}_\mathcal{P}$ ならば，d は $\xrightarrow{*}_{R\cup\{l\to r\Leftarrow c\}}$ に関して真偽安定性を持つ．

［証明］ c, d の真偽が決定する際に代入される項は $\mathcal{F}_\mathcal{P}$ 上の項であり，$l \to r \Leftarrow c$ は $\mathcal{T}(\mathcal{F}_\mathcal{P}, \mathcal{X})$ の項を書き換えない．よって，題意は成り立つ． □

P 文付き TRS では次の補題が P 文の性質から成り立つ．

補題 A.2.3 c を P 文，$R_{+,-}$ を自然数上の加減算を行う $(\mathcal{F}_{\mathcal{P}bgr}, \mathcal{P}bgr)$ 上の P 文付き TRS とする．このとき，c は $\xrightarrow{*}_{R_{+,-}}$ に関して真偽安定性を持つ．また，$R_c \supseteq R_{+,-}$ かつ $R_c \setminus R_{+,-}$ は $+, -$ の規則を持たないとする．このとき，c は $\xrightarrow{*}_{R_c}$ に関して真偽安定性を持つ．

［証明］ 前者は自然数の加減算の性質から成り立つことは明らか．後者は補題 A.2.2 より成り立つ． □

補題 A.2.3 より命題 5.2 が導かれる．

A.3 補題 5.3 の略証

［略証］ t_0, t_1 がそれぞれ規則 $l_i \to r_i \Leftarrow c_i \in R$ により位置 $p_i \in \mathcal{O}(s)$ で書き換えられたとすると，$s|_{p_i} \equiv l_i\sigma_i$，$t_i \equiv s[r_i\sigma_i]_{p_i}$ とおける（$i = 0, 1$）．p_0 と p_1 が並列な位置である場合，および 2 つの規則が重なっている場合は TRS の場合の危険対補題と同様に題意を証明できる．ここでは，上下の関係にあり重なりもない場合，すなわち，$p_1 = p_0p$ となるような p が存在し，$s \equiv (l_0[x]_q)\sigma_0$ かつ $\sigma(x) \equiv C[l_1\sigma_1]_{q'}$ かつ $p = qq'$ の場合を考える．このとき，$t_0 \equiv s[r_0\sigma_0]_{p_0}$，$t_1 \equiv s[(l_0\sigma_0)[r_1\sigma_1]_p]_{p_0}$ である．$\sigma = \sigma_0|_{\mathcal{D}om(\sigma_0)\setminus\{x\}} \cup \{x \mapsto C[r_1\sigma_1]\}$ とすると，$t_0 \equiv s[r_0\sigma_0]_{p_0} \xrightarrow{*}_{l_1\to r_1\Leftarrow c_1} s[r_0\sigma]_{p_0}$．一方，$t_1 \equiv s[(l_0\sigma_0)[r_1\sigma_1]_p]_{p_0} \xrightarrow{*}_{l_1\to r_1\Leftarrow c_1} s[l_0\sigma]_{p_0}$．ここで，$c_0\sigma_0$ が真，$x\sigma_0 \to_{l_1\to r_1\Leftarrow c_1} x\sigma$，$R$ が制約部安定性を持つことから，$c_0\sigma$ は真である．よって，$s[l_0\sigma]_{p_0} \to_{l_0\to r_0\Leftarrow c_0} s[r_0\sigma]_{p_0}$．ゆえに，$t_0 \downarrow_R t_1$ である． □

A.4 命題 6.1 の証明

まずは，制約の下での書換え系列に対して，その制約を満たす代入について，対応した書換え系列が存在することを示す．

補題 A.4.1 $s_0 \underset{c}{\to}_R s_1 \underset{c}{\to}_R \cdots \underset{c}{\to}_R s_n$ ならば，$c\theta$ が真である任意の代入 θ について $s_0\theta \to_R s_1\theta \to_R \cdots \to_R s_n\theta$．

［略証］ $s_i \underset{c}{\to}_R s_{i+1}$ を考える．このとき，$l \to r \Leftarrow d \in R$, $C[\,]$, σ が存在し，$s_i \equiv C[l\sigma]$ かつ $s_{i+1} \equiv C[r\sigma]$ であり，さらに，$c \models d\sigma$．よって，$c\theta$ が真ならば，$d\sigma\theta$ も真である．したがって，$s_i\theta \equiv C\theta[l\sigma\theta] \underset{c}{\to}_R C\theta[r\sigma\theta] \equiv s_{i+1}\theta$． □

命題 6.1 は以下のように証明される．

［証明］ (1)⇒(2) (2) でないとする．このとき，規則 $l \to r \Leftarrow c \in R$ と $c\theta$ が真である代入 θ が存在し，$l\theta \not\succ r\theta$ である．このとき，$l\theta \to_R r\theta$ であるので，(1) より，$l\theta \succ r\theta$ となり，(2) でないことに矛盾する．

(2)⇒(3) (3) でないとする．このとき，制約 c と項 s, t と $c\theta$ が真である代入 θ が存在し，$s \underset{c}{\to}_R t$ かつ $s\theta \not\succ t\theta$ である．$s \underset{c}{\to}_R t$ より，$s \equiv C[l\sigma]$ かつ $t \equiv C[r\sigma]$ かつ $c \models d\sigma$ を満たす規則 $l \to r \Leftarrow d \in R$ と文脈 $C[\,]$ と代入 σ が存在する．$c \models d\sigma$ と $c\theta$ が真であることから，$d\sigma\theta$ も真である．よって，$l\sigma\theta \to_R r\sigma\theta$ であり，$s\theta \to_R t\theta$ が成り立つ．ここで，(2) より，$s\theta \succ t\theta$ が得られ，これは (3) でないことに矛盾する．

(3)⇒(1) (1) でないとする．このとき，$s \to_R t$ かつ $s \not\succ t$ を満たす項 s と t が存在する．$s \to_R t$ より，$s \equiv C[l\sigma]$ かつ $t \equiv C[r\sigma]$ かつ $c\sigma$ が真である規則 $l \to r \Leftarrow c \in R$ と文脈 $C[\,]$ と代入 σ が存在する．$c \models c$ は常に成り立つことは明らかである．よって，$l \underset{c}{\to}_R r$ である．ここで，(3) より，$l\sigma \succ r\sigma$ である．$\succ$ は代入と文脈に閉じるので $s \succ t$ となり，これは (1) でないことに矛盾する． □

A.5 定理 6.4 の証明

本論文では，文献[1] にまとめられている TRS の完備化手続きの正当性の証明を元に，制約付き TRS の完備化の推論規則の正当性を示す．

次の命題で，完備化手続きの正当性を証明するための図 7 の推論規則の性質を示す．

命題 A.5.1 $(E, R) \vdash_\mathscr{C} (E', R')$ とする．このとき，以下のすべてが成り立つ．

(A) $\to_R \subseteq \succ$ ならば，$\to_{R'} \subseteq \succ$．

(B) $\overset{*}{\leftrightarrow}_{E\cup R} = \overset{*}{\leftrightarrow}_{E'\cup R'}$．

(C) E のすべての等式の制約部と R のすべての規則の制約部が $\xrightarrow{*}_R$ に関して真偽安定性を持つならば，E' のすべての等式の制約部と R' のすべての規則の制約部は $\xrightarrow{*}_{R'}$ に関して真偽安定

性を持つ．

(D) $\mathcal{C}_R = \mathcal{C}_{R'}$．

［証明］ (A) Expansion，Simplification，Deletion については明らか．Expansion で加えられる規則で順序が付くのは明らか．Composition のときは，$\to_{s\to u\Leftarrow c} \subseteq \to_{s\to t\Leftarrow c\in R} \circ \to_R \subseteq \succ \circ \succ$ であるので，$\succ$ の推移性より，題意は成り立つ．(B)，(D) は明らかに成り立つ．(C) Expansion のとき，危険対の定義より，E' と R' に現れる制約は，E や R に現れる制約そのものか，$\wedge$ で組み合わせて生成した制約である．よって，補題 A.2.1 より，E' のすべての等式の制約部と R' のすべての規則の制約部は $\xrightarrow{*}_R$ に関して真偽安定性を持つ．$\mathtt{root}(s) \notin \mathcal{F}_{\mathcal{P}}$ なので，補題 A.2.2 より，E' のすべての等式の制約部と R' のすべての規則の制約部は $\xrightarrow{*}_{R'}$ に関して真偽安定性を持つ．Composition，Simplification，Deletion のときは E' と R' の制約の集合は E と R の制約の集合と等しく，さらに $\xrightarrow{*}_{R'} \subseteq \xrightarrow{*}_R$ であるので，題意は成り立つ． □

図 8 の推論規則は等式の制約部分を変形するため，変数を含む 2 つの項では適用後に等価関係が成り立たなくなる場合がある．すなわち，命題 A.5.1（B）が成り立たない．しかし，基底項上の等価関係では制約部が分解されたことは影響がない．例えば，項 $\mathsf{f}(x,y)$ と y は等式 $\mathsf{f}(x,y) \approx y$ で関係付けられるが，分解された等式 $\mathsf{f}(x,y) \to y \Leftarrow x \mathtt{==} 0$ と $\mathsf{f}(x,y) \to y \Leftarrow x \mathtt{>} 0$ では関係付けられない．しかし，変数 x, y に基底項が代入されれば分解された等式でも関係付けられる．よって，基底項上では，等式や書換え規則の制約が分解されても，基底項上の関係は維持される．

命題 A.5.2 $(E,R) \vdash_{\mathscr{C}'} (E',R')$ とする．このとき，以下が成り立つ．

(A) $\to_R \subseteq \succ$ ならば，$\to_{R'} \subseteq \succ$．

(B) 基底項上で $\overset{*}{\leftrightarrow}_{E\cup R} = \overset{*}{\leftrightarrow}_{E'\cup R'}$．

(C) E のすべての等式の制約部と R のすべての規則の制約部が $\xrightarrow{*}_R$ に関して真偽安定性を持つならば，E' のすべての等式の制約部と R' のすべての規則の制約部は $\xrightarrow{*}_{R'}$ に関して真偽安定性を持つ．

(D) $\mathcal{C}_R = \mathcal{C}_{R'}$．

［略証］ E-Decomposition と MGU-Deletion のそれぞれについて，後述の命題 A.5.3 と命題 A.5.4 から (B) が成り立つといえる．

命題 A.5.3 基底項上で $\leftrightarrow_{\{s\approx t\Leftarrow c_1\wedge c_2, s\approx t\Leftarrow c_1\wedge\neg c_2\}} = \leftrightarrow_{\{s\approx t\Leftarrow c_1\}}$． □

命題 A.5.4 $c \in \mathcal{P}bgr$，s と t を単一化可能な項，σ を $\mathcal{D}om(\sigma) \subseteq \mathrm{fv}(c)$ かつ $\mathcal{R}an(\sigma) \subseteq \mathcal{T}(\mathcal{F}_{\mathcal{P}bgr}, \mathrm{fv}(c))$ を満たす s と t の最汎単一化子，$d \in \mathcal{P}bgr$ を $\bigwedge_{x\in\mathcal{D}om(\sigma)} x \mathtt{==} \mathbb{E}^{-1}(x\sigma)$ とする．このとき，基底項上で $\leftrightarrow_{\{s\approx t\Leftarrow c\}} = (\downarrow_{R_c} \cup \leftrightarrow_{\{s\approx t\Leftarrow c\wedge\neg d\}})$．

［略証］ $c\theta$ が真になる基底代入 θ について $s\theta \downarrow_{R_c} t\theta$ または $\neg d\theta$ が真であることは，項の構造に関する帰納法により証明できる．よって，題意は成り立つ． □

$R_\infty = \bigcup_{i=0}^{n} R_i$，$E_\infty = \bigcup_{i=0}^{n} E_i$ とする．R_∞ の危険対は以下の性質を持つ．

補題 A.5.5 E_0 を等式の集合，R_0 を $\to_{R_0} \subseteq \succ$ を満たす完備な制約付き TRS とし，E_0 のすべての等式の制約部と R_0 のすべての規則の制約部が $\xrightarrow{*}_{R_0}$ の下で真偽安定，$(E_0,R_0) \vdash_{\mathscr{C}} \cdots \vdash_{\mathscr{C}} (E_n,R_n)$ を得たとする．このとき，任意の重なりのある 2 つ規則 $l_1 \to r_1 \Leftarrow c_1 \in R_\infty$ と $l_2 \to r_2 \Leftarrow c_2 \in R_\infty$ の危険対 $\langle s,t\rangle \Leftarrow c$ について，以下のどちらかが成り立つ．

- s と t が c の下で R_∞ に関して会同関係にある．
- ある i（$1 \le \le n$）で，同様の位置で重なる 2 つの規則 $l_1 \to r_1' \Leftarrow c_1 \in R_i$ と $l_2 \to r_2' \Leftarrow c_2 \in R_i$ の危険対 $\langle s',t'\rangle \Leftarrow c$ が E_i に存在し，$s' \xrightarrow[c]{*}_{R_\infty} s$ と $t' \xrightarrow[c]{*}_{R_\infty} t$ を満たす．

［略証］ n に関する帰納法で証明できる． □

$\leftrightarrow_{E_\infty\cup R_\infty}$ で関係付けられる項の列が R_n で会同関係にあることを示すための，証明系列の概念とその重みを定義する．

定義 A.5.6 E を等式集合，R を制約付き TRS とする．$s_0 \leftrightarrow_{E\cup R} s_1 \leftrightarrow_{E\cup R} \cdots \leftrightarrow_{E\cup R} s_n$ のとき，$(s_0, s_1, \cdots, s_n)$ を**証明系列**という．$s \leftrightarrow_{E\cup R} t$ のコスト $\mathcal{C}ost(s,t)$ を，項の 2 つの多重集合と，代入と制約の対の三つ組として，以下のように定義する．

- $s \equiv C[s'\sigma] \leftrightarrow_{\{s'\simeq t'\Leftarrow c\}} C[t'\sigma] \equiv t$ のとき，$\mathcal{C}ost(s,t) = (\{s,t\}, \{s',t'\}, (\sigma,c))$．
- $s \to_R t$ のとき，$\mathcal{C}ost(s,t) = (\{s\},\{t\},-)$．
- $s \leftarrow_R t$ のとき，$\mathcal{C}ost(s,t) = (\{t\},\{s\},-)$．

ここで，$-$ は任意の対を表す． □

$\mathcal{C}ost$ の 1 番目と 2 番目の要素である項の多重集合は，項上の順序 $\succ$ で定められる項の多重集合上の

順序 $\succ^{Mul}$ で比較される．3 番目の要素 (σ, c) は E-Decomposition と MGU-Deletion が適用された際の制約を順序付けるために用いられる．σ は基底代入のみを考える．制約と代入の対 (σ, c)（ただし，$\mathrm{fv}(c) \subseteq \mathcal{D}om(\sigma)$）の重みは以下の全てを満たす基底代入の個数とする．

- $\mathcal{D}om(\theta) = \mathcal{D}om(\sigma)$,
- $c\theta$ が真,
- $t \in \mathcal{R}an(\theta)$ の項の高さは $\mathcal{R}an(\sigma)$ に現れる項の高さの最大値を越えない．

関数記号が有限であるので，高さの最大値を決められた下での基底項の個数は有限である．よって，θ の個数は有限であり，重みの比較が可能である．(σ, c) が (σ, c') より上記を満たす代入の個数が多いとき $(\sigma, c) > (\sigma, c')$ と記す．このような重みについて以下の性質が成り立つ．

命題 A.5.7 σ を $\mathcal{D}om(\sigma) \neq \emptyset$ を満たす基底代入，c を $\mathrm{fv}(c) \subseteq \mathcal{D}om(\sigma)$ を満たす制約，d を $\mathrm{fv}(d) \subseteq \mathcal{D}om(\sigma)$ を満たす制約とする．このとき，$c \models d$ ならば，$(\sigma, c) > (\sigma, c \wedge \neg d)$． □

定義より，制約上の重みは停止性を持つ．

コスト上の順序 $\succ_{\mathcal{C}ost}$ は $\succ^{Mul}$ と $\succ^{Mul}$ と $>$ の辞書式順序とする．証明系列 $(s_0, s_1, \cdots, s_n)$ のコスト $\mathcal{C}ost((s_0, s_1, \cdots, s_n))$ を各 s_i, s_{i+1} $(0 \leq i < n)$ のコストの多重集合とする．$\succ_{\mathcal{C}ost}$ で定まるコストの多重集合上の順序を $\succ^{Mul}_{\mathcal{C}ost}$ で表す．$\succ^{Mul}_{\mathcal{C}ost}$ は帰納法に利用できる整礎な順序である．

命題 A.5.8 $\succ$ が簡約化順序ならば，$\succ^{Mul}_{\mathcal{C}ost}$ は停止性を持つ． □

次に，2 つの証明系列とその重みの関係を示す．

補題 A.5.9 E を等式集合，R を制約付き TRS，$\succ$ を簡約化順序，$p1$ と $p2$ を証明系列とする．$\rightarrow_R \subseteq \succ$ のとき，$p1$ と $p2$ が以下のどれかを満たすならば，$\mathcal{C}ost(p1) \succ^{Mul}_{\mathcal{C}ost} \mathcal{C}ost(p2)$ である．

(1) $p1$ が $s \leftrightarrow_E t$ であり，$p2$ が $s \rightarrow_R t$.
(2) $p1$ が $s \leftrightarrow_E t$ であり，$p2$ が $s \leftrightarrow_E u \leftarrow_R t$.
(3) $p1$ が $s \rightarrow_R t$ であり，$p2$ が $s \rightarrow_R u \leftarrow_R t$.
(4) $p1$ が $s \rightarrow_R t$ であり，$p2$ が $t \rightarrow_R u$.
(5) $p1$ が $s \rightarrow_R t$ であり，$p2$ が $t \rightarrow_R t_1 \rightarrow_R \cdots \rightarrow_R t_n$ $(n > 0)$.
(6) $p1$ が $s \rightarrow_R t \rightarrow_R u$，$p2$ が $s \leftarrow_R \cdots \leftarrow_R t_m \leftrightarrow_E u_n \rightarrow_R \cdots \rightarrow_R u_1 \rightarrow_R u$ であり，$t_m \overset{+}{\leftarrow}_R t \overset{+}{\rightarrow}_R u_n$ を満たす．
(7) s と t を基底項，c と d を $\mathrm{fv}(d) \subseteq \mathrm{fv}(c)$ かつ $c \wedge d$ が充足可能である制約とする．$p1$ が $s \leftrightarrow_{\{s' \simeq t' \Leftarrow c\}} t$ であり，$p2$ が $s \leftrightarrow_{\{s' \simeq t' \Leftarrow c \wedge \neg d\}} t$.

［証明］ $\mathcal{C}ost$ と $\succ^{Mul}_{\mathcal{C}ost}$ の定義，仮定 $\rightarrow_R \subseteq \succ$ より明らか． □

次の補題で，$s_0 \leftrightarrow_{E_\infty \cup R_\infty} s_1 \leftrightarrow_{E_\infty \cup R_\infty} \cdots \leftrightarrow_{E_\infty \cup R_\infty} s_m$ について，s_0 と s_m が R_n で会同関係にあることを示す．

補題 A.5.10 E_0 を等式の集合，R_0 を $\rightarrow_{R_0} \subseteq \succ$ を満たす完備な制約付き TRS とし，E_0 のすべての等式の制約部と R_0 のすべての等式の制約部が $\overset{*}{\rightarrow}_{R_0}$ の下で真偽安定，$(E_0, R_0) \vdash_{\mathscr{C}} \cdots \vdash_{\mathscr{C}} (E_n, R_n) = (\emptyset, R_n)$ を得たとする．$s_0 \leftrightarrow_{E_\infty \cup R_\infty} s_1 \leftrightarrow_{E_\infty \cup R_\infty} \cdots \leftrightarrow_{E_\infty \cup R_\infty} s_m$ $(m \geq 0)$ を証明系列とする．このとき，$s \downarrow_{R_n} t$.

［略証］ p を証明系列 $s_0 \leftrightarrow_{E_\infty \cup R_\infty} s_1 \leftrightarrow_{E_\infty \cup R_\infty} \cdots \leftrightarrow_{E_\infty \cup R_\infty} s_m$ $(m \geq 0)$ として，$\succ^{Mul}_{\mathcal{C}ost}$ に関する帰納法によって示す．以下の証明では，E-Decomposition と MGU-Deletion の議論の際には，証明系列は基底項上であると仮定する．

$R_n \subseteq R_\infty$ より，証明系列の各 $\leftrightarrow_{E_\infty \cup R_\infty}$ は，$\leftrightarrow_{E_\infty}$，$\leftrightarrow_{R_\infty \setminus R_n}$，$\leftrightarrow_{R_n}$ のいずれかである．証明系列が $s_0 \rightarrow_{R_n} \cdots \rightarrow_{R_n} s_i \leftarrow_{R_n} \cdots \leftarrow_{R_n} s_m$ のとき，もしくは $n = 0$ のときは，$s_0 \downarrow_{R_n} s_n$．そうでないときは，以下のいずれかが証明中に存在する．

(1) $s_i \leftrightarrow_{E_\infty} s_{i+1}$.
(2) $s_i \leftrightarrow_{E_\infty} s_{i+1}$ かつ $s_i \equiv s_{i+1}$ かつ $n > 0$ かつ $0 \leq i < n$.
(3) $s_i \rightarrow_{R_\infty \setminus R_n} s_{i+1}$ または $s_i \leftarrow_{R_\infty \setminus R_n} s_{i+1}$.
(4) $s_{i-1} \leftarrow_{R_n} s_i \rightarrow_{R_n} s_{i+1}$.

推論規則の定義と補題 A.5.5 より，これらの部分はそれぞれ以下の系列に置き換えられる．

(1) (a) $s_i \rightarrow_{R_\infty} s_{i+1}$ または $s_i \leftarrow_{R_\infty} s_{i+1}$ (Expansion)．
(b) $s_i \leftrightarrow_{\{s \simeq t \Leftarrow c\}} s_{i+1}$ とすると，$s_i \leftrightarrow_{E_\infty} u \leftarrow_{R_\infty} s_{i+1}$ または $s_i \rightarrow_{R_\infty} u \leftrightarrow_{E_\infty} s_{i+1}$ または $s_i \leftrightarrow_{\{s \simeq t \Leftarrow c \wedge \neg d\}} s_{i+1}$（ただし，$\mathrm{fv}(d) \subseteq \mathrm{fv}(c)$ かつ $c \wedge d$ は充足可能）(E-Decomposition)．
(c) $s_i \leftrightarrow_{\{s \simeq t \Leftarrow c\}} s_{i+1}$ かつ s と t が単一化可能であるとすると，制約 d が存在して，$s_i \downarrow_{R_\infty} s_{i+1}$ または $s_i \leftrightarrow_{\{s \simeq t \Leftarrow c \wedge \neg d\}}$

s_{i+1}（ただし，$\mathrm{fv}(d) \subseteq \mathrm{fv}(c)$）を満たす（MGU-Deletion）．

(2) s_{i+1} を除去（Deletion）．

(3) $s_i \leftrightarrow_{E_\infty} t \leftarrow_{R_\infty} s_{i+1}$ または $s_i \rightarrow_{R_\infty} t \leftrightarrow_{E_\infty} s_{i+1}$（Composition）．

(4) $s_i \equiv t_0 \rightarrow_{R_\infty} \cdots \rightarrow_{R_\infty} t_j \leftarrow_{R_\infty} \cdots \leftarrow_{R_\infty} t_k \equiv s_{i+1}$（重なりがない場合の危険対補題 5.3）または $s_i \equiv t_0 \rightarrow_{R_\infty} \cdots \rightarrow_{R_\infty} t_j \leftrightarrow_{E_\infty} t_{j+1} \leftarrow_{R_\infty} \cdots \leftarrow_{R_\infty} t_k \equiv s_{i+1}$（補題 A.5.5）．

p 中の 1 つの部分を上記のように置き換えて得られる証明系列 p' を $s'_0 \leftrightarrow_{E_\infty \cup R_\infty} s'_1 \leftrightarrow_{E_\infty \cup R_\infty} \cdots \leftrightarrow_{E_\infty \cup R_\infty} s'_{m'}$ とすると，$s'_0 \equiv s_0$ かつ $s'_{m'} \equiv s_m$．また，補題 A.5.9 より，$\mathcal{C}ost(p) \succ^{Mul}_{Cost} \mathcal{C}ost(p')$．よって，帰納法の仮定より，$s'_0 \downarrow_{R_n} s'_{m'}$．ゆえに，$s_0 \downarrow_{R_n} s_m$． □

最後に，定理 6.4 の証明を以下に与える．

［証明］ $\overset{*}{\leftrightarrow}_{E_0 \cup R_0} = \overset{*}{\leftrightarrow}_{R_n}$，$R_n$ が制約部安定性と停止性を持つことは命題 A.5.1 より明らか．

次に，合流性を持つことを示す．$t_1 \overset{*}{\leftarrow}_{R_n} s \overset{*}{\rightarrow}_{R_n} t_2$ とする．このとき，$R_n \subseteq R_\infty$ より，$t_1 \overset{*}{\leftrightarrow}_{E_\infty \cup R_\infty} t_2$ である．よって，補題 A.5.10 より，$t_1 \downarrow_{R_n} t_2$．ゆえに，R_n は合流性を持つ． □

A.6 命題 7.5 の証明

TRS が弱停止性と擬簡約性を持つとき，十分完全性を持つことが知られている．よって，制約付き TRS についても同様の手順で十分完全性を示す．

定義 A.6.1 R を制約付き TRS，$f \in \mathcal{D}_R$ とする．任意の $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}_R)$ について $f(t_1, \ldots, t_n)$ が R の正規形でないとき，f は R において**疑似簡約性**を持つという． □

補題 A.6.2 R を左線形制約付き TRS，$f \in \mathcal{D}_R$ とする．すべての代入 $\theta = \{x_i \mapsto t_i \mid 1 \le i \le n, t_i \in \mathcal{T}(\mathcal{C}_R)\}$ に対して論理式 $\left(\bigvee_{f(x_1,\cdots,x_n) \to u_i \Leftarrow c_i \in R} c_i\right)\theta$ が真であるとする．このとき，f は R において疑似簡約性を持つ．

［証明］ 条件より，どんな $t_i \in \mathcal{T}(\mathcal{C}_R)$ を与えても少なくとも一つ以上の規則 $f(x_1, \cdots, x_n) \to u_i \Leftarrow c_i$ の制約部が真となるから $f(t_1, \cdots, t_n) \rightarrow_R u$．よって，$f$ は疑似簡約性を持つ． □

補題 A.6.3 疑似簡約性と弱停止性をもつ制約付き TRS R は十分完全性を持つ．

［証明］ R が弱停止性を持つことからどの $t \in \mathcal{T}(\mathcal{F})$ についても $t \overset{*}{\rightarrow}_R u \in NF_R(\mathcal{F})$ となる u が存在する．このとき，$u \notin \mathcal{T}(\mathcal{C}_R)$ を仮定すると，$f \in \mathcal{D}_R$ かつ $t_i \in \mathcal{T}(\mathcal{C}_R)$ を満たす u の部分項 $f(t_1, \cdots, t_n)$ が存在する．R は疑似簡約性を持つことからこの部分項は正規形ではない．これは u が正規形であることに矛盾する．よって，R は十分完全性を持つ． □

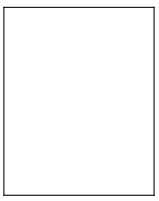
命題 7.5 は以下のように証明される．

［証明］ 左線形性は作り方より明らか．$\mathbb{TR}$ で生成された規則集合が補題 A.6.2 を満たすことは作り方より明らか．また，R_c は十分完全性を持つように与えられている．よって，弱停止性を持つならば，補題 A.6.3 より十分完全性を持つ． □

A.7 定理 7.6 の証明

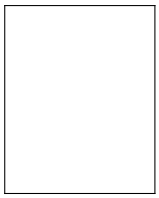
［証明］ $R_1 = R$，$R_2 = R'$，関係 $\rightarrow_i$ を $\rightarrow_{R_i}$ と定義，$S_i = (\mathcal{T}(\mathcal{F}), \rightarrow_i)$ とする．このとき，R が停止性を持つことから，S_1 は停止性を持つ．また，完備化の性質（定理 6.4）より，$\overset{*}{\leftrightarrow}_{E \cup R} \subseteq \overset{*}{\leftrightarrow}_{R'}$ かつ $NF_{\rightarrow_R}(\mathcal{F}) = NF_{\rightarrow_{R'}}(\mathcal{F})$ であり，R' は基底項に関して完備性を持つ．よって，$\overset{*}{\leftrightarrow}_1 \subseteq \overset{*}{\leftrightarrow}_2$ かつ $NF_{S_1} = NF_{S_2}$ であり，S_2 は合流性を持つ．ゆえに，S_1 と S_2 は潜在帰納法の条件を満たすので，定理 7.1 より，$\overset{*}{\leftrightarrow}_1 = \overset{*}{\leftrightarrow}_2$ である．したがって，$\overset{*}{\leftrightarrow}_{R_1} = \overset{*}{\leftrightarrow}_{R_2}$ である．ここで，$s \approx t \Leftarrow c \in E$ と $c\theta_g$ を真にする基底代入 θ_g を考える．$s\theta_g \leftrightarrow_{\{s \approx t \Leftarrow c\}} t\theta_g$ より，$s\theta_g \leftrightarrow_E t\theta_g$ である．R' の性質より $s\theta_g \overset{*}{\leftrightarrow}_{R'} t\theta_g$ である．したがって，$s\theta_g \overset{*}{\leftrightarrow}_R t\theta_g$ であるので，$s \approx t \Leftarrow c$ は R の帰納的定理である． □

(平成 20 年 1 月 17 日受付)
(平成 20 年 5 月 1 日採録)



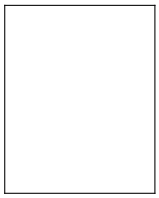
古市 祐樹

2005 名古屋大学工学部電気電子・情報工学科卒，2007 同大学大学院情報科学研究科修士課程了．項書換え系，プログラム変換に関する研究に従事．



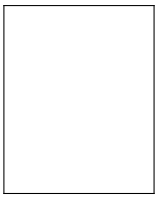
西田　直樹

2000 名大・工・電気電子・情報工卒，2002 同大大学院工学研究科計算理工学博士前期課程了，2004 同大学院工学研究科情報工学専攻博士後期課程了，2004 同大学院情報科学研究科助手，2007 より同助教，現在に至る．項書換え系，プログラム変換に関する研究に従事．工博．電子情報通信学会，日本ソフトウェア科学会各会員．



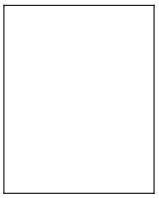
酒井　正彦

1989 名古屋大学大学院博士課程満了．同年同大工学部助手．1993 北陸先端科学技術大学院大学助教授．1997 名古屋大学工学研究科助教授．2002 同教授．2003 同大大学院情報科学研究科教授，現在に至る．この間，1996 年 3 月〜8 月ニューヨーク州立大学スートーニーブルック校客員研究教授．項書換え系などのソフトウェア基礎理論に関する研究に従事．工学博士．平成 3 年度電子情報通信学会論文賞受賞．電子情報通信学会，日本ソフトウェア科学会各会員．



草刈圭一朗（正会員）

1994 東京工業大学理学部生命理学科卒．1996 北陸先端科学技術大学院大学情報科学研究科博士前期課程了．2000 同大学博士後期課程にて博士 (情報科学) 取得．同年東北大学電気通信研究所助手．2003 名古屋大学大学院情報科学研究科情報システム学専攻講師．2006 より同大学院研究科計算機数理科学専攻助教授，2007 より同准教授．項書換え系・プログラム理論・定理自動証明の研究に従事．電子情報通信学会・日本ソフトウェア科学会会員．



坂部　俊樹（正会員）

1972 名古屋大学工学部電気学科卒．1977 同大大学院博士課程了．名古屋大学助手，三重大学助教授，名古屋大学助教授を経て，1993 より名古屋大学教授．ソフトウェアの基礎理論全般に興味を持つ．抽象データ型の理論，プログラミング言語の形式的意味論，自動プログラミング，書換え型計算モデル，並行プロセスの理論などの研究に従事．工学博士．電子情報通信学会，人工知能学会，日本ソフトウェア科学会，EATCS 各会員．