パターンに基づくプログラム変換における列変数の導入
Introducing Sequence Variables in Program Transformation based on Templates

千葉 勇輝 † **青戸 等人** † **外山 芳人** †
Yuki Chiba Takahito Aoto Yoshihito Toyama

1. はじめに

Huet と Lang により提案されたパターンに基づくプログラム変換 [7] は，プログラム効率化の有効な手法として現在活発に研究されている [3, 4, 5, 6, 11]．しかし，この手法はラムダ計算に基づいているため，変換の正当性を形式的に検証することが容易ではなく，検証の自動化についてもほとんど研究されていなかった．

著者らは，形式化に項書き換えシステムを用いた，パターンに基づくプログラム変換法を文献 [2] において提案した．ラムダ計算を用いた従来手法と対照的に，この手法では操作的意味論に基づく正当性検証が容易であり，項書き換えシステムの自動定理証明法を応用することにより変換の正当性の自動検証も可能となる．しかし，項書き換えシステムの枠組みでは，変換パターンに出現する関数の引数の数 (次数) が固定されているため，基本的に同じ変換パターンであっても，関数の次数に応じて異なる変換パターンを用意する必要があった．一方，ラムダ計算に基づく変換パターンでは，η 変換を効果的に用いることで，同一の変換パターンを関数の次数に無関係に利用することができる．

本論文では，項書き換えシステムに列変数の概念を導入することで，ラムダ計算の η 変換を用いなくても，関数の次数に依存しない柔軟な変換パターンが実現できることを明らかにする．列変数をもつ (第 1 階) 項の単一化問題の決定可能性はすでに知られており [8]，項書き換えシステムに基づく XML 文書の変換へ列変数の応用が試みられている [9, 10]．しかし，列変数を含む第 2 階変換パターンについてはこれまでほとんど研究されていない．本論文では，列変数を導入した第 2 階変換パターンを形式化するとともに，その照合アルゴリズムを提案する．この照合アルゴリズムを利用することで，より汎用性のあるパターンを用いて項書き換えシステムに基づくプログラム変換を実現することが可能となる．

2. 変換パターンにおける列変数の導入

項書き換えシステムおよび項書き換えシステムに基づくプログラム変換の枠組みにおける記法は文献 [1, 2] に従う．

次の項書き換えシステム $\mathcal{R}_{\mathsf{length}}$ は与えられたリストの長さを計算する length 関数を再帰的に定義する．

$$\mathcal{R}_{\mathsf{length}} = \left\{ \begin{array}{ll} \mathsf{length}([\,]) & \to 0 \\ \mathsf{length}(z{:}zs) & \to \mathsf{s}(\mathsf{length}(zs)) \end{array} \right\}$$

一方，length 関数は次の項書き換えシステム $\mathcal{R}'_{\mathsf{length}}$ により反復的に定義することも出来る．

$$\mathcal{R}'_{\mathsf{length}} = \left\{ \begin{array}{ll} \mathsf{length}(zs) & \to \mathsf{length}_1(zs, 0) \\ \mathsf{length}_1([\,], y) & \to y \\ \mathsf{length}_1(z{:}zs, y) & \to \mathsf{length}_1(zs, \mathsf{s}(y)) \end{array} \right\}$$

反復的なプログラムは再帰的なプログラムより効率的であり，$\mathcal{R}_{\mathsf{length}}$ を $\mathcal{R}_{\mathsf{length}'}$ へ変換する手法は基本的なプログラム変換技術としてよく知られている．

同様のプログラム変換として，3 つの自然数の合計を計算する関数 add3 を定義する項書き換えシステムの変換がある．

$$\mathcal{R}_{\mathsf{add3}} = \left\{ \begin{array}{ll} \mathsf{add3}(0, y, z) & \to +(y, z) \\ \mathsf{add3}(\mathsf{s}(w), y, z) & \to \mathsf{s}(\mathsf{add3}(w, y, z)) \end{array} \right\}$$

$$\mathcal{R}'_{\mathsf{add}} = \left\{ \begin{array}{ll} \mathsf{add3}(w, y, z) & \to \mathsf{add}_1(w, +(y, z)) \\ \mathsf{add}_1(0, y) & \to y \\ \mathsf{add}_1(\mathsf{s}(w), y) & \to \mathsf{add}_1(w, \mathsf{s}(y)) \end{array} \right\}$$

$\mathcal{R}_{\mathsf{length}}$ から $\mathcal{R}'_{\mathsf{length}}$ への変換と $\mathcal{R}_{\mathsf{add}}$ から $\mathcal{R}'_{\mathsf{add}}$ への変換はほぼ同じ構造をしている．パターンに基づくプログラム変換は，このような類似の構造による変換を変換パターンとして一般化し，入力プログラムとの照合によって効率の良いプログラムを出力する．

しかしながら，length と add3 は異なる次数を持つ関数であるため，文献 [2] のプログラム変換の枠組みでは同一の変換パターンで取り扱うことが困難である．そこで，任意個数の変数に対応できる変換パターンを記述するため，列変数 $\overline{x}, \overline{y}, \cdots$ を導入する．

例えば，上記の 2 つの変換例を列変数を用いて一般化すると以下の変換パターンが得られる．

例 1 (列変数を用いた変換パターン)

$$\mathcal{P} = \left\{ \begin{array}{ll} \mathsf{f}(\mathsf{a}, \overline{y}) & \to \mathsf{b}(\overline{y}) \\ \mathsf{f}(\mathsf{c}(x, \overline{x}), \overline{y}) & \to \mathsf{d}(\mathsf{f}(x, \overline{y}), \overline{x}) \end{array} \right\}$$

$$\mathcal{P}' = \left\{ \begin{array}{ll} \mathsf{f}(x, \overline{y}) & \to \mathsf{f}_1(x, \mathsf{b}(\overline{y})) \\ \mathsf{f}_1(\mathsf{a}, y) & \to y \\ \mathsf{f}_1(\mathsf{c}(x, \overline{x}), y) & \to \mathsf{f}_1(x, \mathsf{d}(y, \overline{x})) \end{array} \right\}$$

ここで，$\mathsf{length}([\,])$ と $\mathsf{f}(\mathsf{a}, \overline{y})$ は $\mathsf{f} \mapsto \mathsf{length}$, $\mathsf{a} \mapsto [\,]$, $\overline{y} \mapsto \varepsilon$ で対応付けられ，$\mathsf{add3}(0, y, z)$ と $\mathsf{f}(\mathsf{a}, \overline{y})$ は $\mathsf{f} \mapsto \mathsf{add3}$, $\mathsf{a} \mapsto 0$, $\overline{y} \mapsto y, z$ で対応付けられる．従って，列変数によって次数に依存しない変換パターンが実現できる．

3. 変換パターンの形式化

列変数を導入したパターンを考える前に，第 1 階の項に列変数を導入する場合を考えよう．そのために，通常の関数記号集合 $\mathscr{F}$ および変数記号集合 $\mathscr{V}$ の他に，列変

† 東北大学 電気通信研究所

数記号集合 $\overline{\mathscr{V}}$ を考える．列変数記号には $\overline{x}, \overline{y}, \ldots$ を使用する．次数 n の関数記号 $f \in \mathscr{F}$ に対して，第 1 階項の場合は $f(t_1, \ldots, t_n)$ と引数の個数は n 個になるが，列変数がある場合は $\overline{x}$ が 2 つやそれ以上，あるいは，0 個の引数の代わりとなるため，一般には f は $f(t_1, \ldots, t_m)$ と次数とは関係ない個数の引数を持つことになる．

パターン変数記号を導入して得られる項をパターンとよぶ．パターン変数記号の集合を $\mathscr{P}$ と記す．次数 n のパターン変数記号 $p \in \mathscr{P}$ に対して，パターン $p(t_1, \ldots, t_n)$ は $t_1, \ldots, t_n$ を部分にもつ項に対応する．このとき，それぞれの t_i は項中の独立の場所に現われるので，$t_1, \ldots, t_n$ という列となっているわけではない．従って，列変数を導入した場合でもパターン変数の引数の個数は次数 n のまま変化しない．

このように，列変数が導入された場合，関数記号の (可変長の) 引数列とパターン変数記号の (固定長の) 引数列の取り扱いが異なってくる．これらを区別するために，可変長の引数列を列として明示する記法を採る．つまり，項 (パターン) の結合を表わす結合的な演算子ドット (.) を用いて，可変長の列を表わすことにする．すると，パターン変数記号を根に持つパターンは $p(t_1, \ldots, t_n)$ のままだが，関数記号を根に持つ項パターンは $f(t_1 . \cdots . t_m)$ と表記されることになる．

この記法のもとで，例えば，$x, u, v, w \in \mathscr{V}$ とするとき，パターン $p(x, \overline{y})$ と項 $f(u.v.w)$ と照合を考える．$p := f(\_ \ldots \_)$ が判明したとき，照合アルゴリズムは以下のような変換を行っていくことにより残りの照合解を発見する．

$$\begin{array}{rl} & p(x, \overline{y}) \trianglelefteq f(u.v.w) \\ \Rightarrow & f(x.\overline{y}) \trianglelefteq f(u.v.w) \\ \Rightarrow & x.\overline{y} \trianglelefteq u.v.w \\ \Rightarrow & x \trianglelefteq u,\ \overline{y} \trianglelefteq v.w \end{array}$$

以下は本記法の形式的な定義を与えている．

定義 1 (項/変換パターンにおける列変数の導入) 関数記号，パターン変数記号，局所変数記号，列変数記号の集合をそれぞれ，$\mathscr{F}$, $\mathscr{P}$, $\mathscr{V}$, $\overline{\mathscr{V}}$ とおく．また，関数 arity : $\mathscr{F} \cup \mathscr{P} \to \mathbb{N}$ が与えられているものとする．**項パターン** $T(\mathscr{F} \cup \mathscr{P}, \mathscr{V} \cup \overline{\mathscr{V}})$ と**項パターン列** $T^*(\mathscr{F} \cup \mathscr{P}, \mathscr{V} \cup \overline{\mathscr{V}})$ を以下のように定義する (以下では T, T^* と略す．また，T^* の元を表わすメタ変数として以下では s^*, t^* 等を用いる)．

(1) $\mathscr{V} \cup \overline{\mathscr{V}} \subseteq T$, $\varepsilon \in T^*$;
(2) $f \in \mathscr{F}$, $s^* \in T^* \Rightarrow f(s^*) \in T$;
(3) $p \in \mathscr{P}$, $s_1, \ldots, s_n \in T \Rightarrow p(s_1, \ldots, s_n) \in T$;
　(ただし，$n = \mathrm{arity}(p)$ とする)
(4) $s \in T$, $t^* \in T^* \Rightarrow s.t^* \in T^*$.

ただしここで，ε はドット (.) の単位元である．従って，(1), (4) より $T \subseteq T^*$ となることに注意しておく．項パターン列 t^* に含まれる局所変数集合，列変数集合をそれぞれ $\mathscr{V}(t^*)$, $\overline{\mathscr{V}}(t^*)$ で表す．また，局所変数記号のみを含む項パターン列の集合を $\mathscr{V}^*$ と記す．$|s^*|$ は項列 s^* の長さを表わす．項パターン上の項書き換えシステムを**パターン項書き換えシステム**，パターン項書き換えシステムの対を**変換パターン**とよぶ．

通常の第 1 階項は項パターンの特殊な場合として次のように定式化される．つまり，パターン変数および列変数を含まない項パターン $t \in T(\mathscr{F}, \mathscr{V})$ が**第 1 階項**であるとは，$t \in \mathscr{V}$，または，$\mathrm{arity}(f) = n$ なる $f \in \mathscr{F}$ および第 1 階項 $s_1, \ldots, s_n$ に対して $t = f(s_1 . \cdots . s_n)$ となるときをいう．

4. 項パターンの照合アルゴリズム

変換パターンを項書き換えシステムと照合して項書き換えシステムの変換を実現するには，パターン項書き換えシステムと項書き換えシステムの照合問題を解くことが必要になる．パターン項書き換えシステムや項書き換えシステムは，それぞれ項パターンおよび項の対の集合であるから，この照合問題は項パターンと項の照合問題に帰着される．

照合問題とその解を定式化するために文献 [2] 同様インデックス付き文脈を用いる．インデックス付き文脈とは，特別な定数 $\Box_i$ $(i \geq 1)$ を持つ第 1 階項のことをいう．局所変数を含まないインデックス付き文脈の集合 $T(\mathscr{F} \cup \{\Box_i \mid 1 \leq i \leq n\}, \emptyset)$ を $C_n(\mathscr{F})$ と記す．インデックス付き文脈 $C \in C_n(\mathscr{F})$ に対して，C 中の $\Box_i$ をそれぞれ s_i で置き換えたものを $C\langle s_1, \ldots, s_n \rangle$ と記す．たとえば，$\mathsf{f}(\Box_1.\Box_3.\Box_1) \in C_3(\mathscr{F})$ に対して，$\mathsf{f}(\Box_1.\Box_3.\Box_1)\langle \mathsf{a}, \mathsf{b}, \mathsf{c} \rangle = \mathsf{f}(\mathsf{a}.\mathsf{c}.\mathsf{a})$ となる．また，パターン変数を含む基底文脈の集合 $T(\mathscr{F} \cup \mathscr{P} \cup \{\Box_i \mid 1 \leq i\}, \emptyset)$ を $C(\mathscr{F} \cup \mathscr{P})$ と記す．

定義 2 (項準同型写像) $\mathscr{V} \cup \overline{\mathscr{V}} \cup \mathscr{P}$ から $\mathscr{V}^* \cup \overline{\mathscr{V}} \cup C(\mathscr{F} \cup \mathscr{P})$ への関数 φ が以下の条件を満たすとき φ を**項準同型写像**とよぶ．

(1) $x \in \mathrm{dom}_{\mathscr{V}}(\varphi) \Rightarrow \varphi(x) \in \mathscr{V}$;
(2) $\overline{x} \in \mathrm{dom}_{\overline{\mathscr{V}}}(\varphi) \Rightarrow \varphi(\overline{x}) \in \mathscr{V}^*$;
(3) $p \in \mathrm{dom}_{\mathscr{P}}(\varphi) \Rightarrow \varphi(p) \in C_{\mathrm{arity}(p)}(\mathscr{F})$;
(4) φ は $\mathrm{dom}_{\mathscr{V} \cup \overline{\mathscr{V}}}(\varphi)$ 上で単射，つまり，任意の異なる変数 $\tilde{x}, \tilde{y} \in \mathrm{dom}_{\mathscr{V} \cup \overline{\mathscr{V}}}(\varphi)$ について，$\mathscr{V}(\varphi(\tilde{x})) \cap \mathscr{V}(\varphi(\tilde{y})) = \emptyset$).

ただしここで，定義域 $\mathrm{dom}_{\mathscr{V}}(\varphi) = \{x \in \mathscr{V} \mid \varphi(x) \neq x\}$, $\mathrm{dom}_{\overline{\mathscr{V}}}(\varphi) = \{\overline{x} \in \overline{\mathscr{V}} \mid \varphi(\overline{x}) \neq \overline{x}\}$, $\mathrm{dom}_{\mathscr{P}}(\varphi) = \{p \in \mathscr{P} \mid \varphi(p) \neq p(\Box_1, \ldots, \Box_{\mathrm{arity}(p)})\}$, $\mathrm{dom}_{\mathscr{V} \cup \overline{\mathscr{V}}}(\varphi) = \mathrm{dom}_{\mathscr{V}}(\varphi) \cup \mathrm{dom}_{\overline{\mathscr{V}}}(\varphi)$ とする．また，$\mathrm{range}_{\mathscr{V} \cup \overline{\mathscr{V}}}(\varphi) = \bigcup \{\mathscr{V}(\varphi(\tilde{x})) \mid \tilde{x} \in \mathrm{dom}_{\mathscr{V} \cup \overline{\mathscr{V}}}(\varphi)\}$ とおく．次に，項準同型写像 φ を T^* 上の関数 $\hat{\varphi}$ に拡張する．

$$\hat{\varphi}(s) = \begin{cases} \varphi(s) & (s \in \mathscr{V} \cup \overline{\mathscr{V}} \text{ のとき}) \\ \varepsilon & (s = \varepsilon \text{ のとき}) \\ f(\hat{\varphi}(t^*)) & (s = f(t^*) \text{ のとき}) \\ \varphi(p)\langle \hat{\varphi}(s_1), \ldots, \hat{\varphi}(s_n) \rangle & \\ & (s = p(s_1, \ldots, s_n) \text{ のとき}) \\ \hat{\varphi}(t).\hat{\varphi}(t^*) & (s = t.t^* \text{ のとき}) \end{cases}$$

慣習に従い，以下では $\hat{\varphi}$ と φ を同一視する．

文献 [3, 4, 5, 6, 7, 11] 等のラムダ計算による枠組みにおいて，パターンとプログラムにおける局所変数の対応はラムダ束縛によって 1 対 1 に決定される．本研究では項書き換えシステムに基づき形式化を行っているため，

そのままでは局所変数の対応を一意に決定することが出来ない．そこで項準同形写像に (4) の条件を加えることでラムダ計算の枠組みと同様に局所変数の対応を 1 対 1 にしている．局所変数の対応の一意性を保証することは，文献 [2] のようなプログラム変換の正当性の検証法に不可欠である．

定義 3 (照合問題と照合解) 1. 項パターン列 s^* と第 1 階項 $t_1, \ldots, t_n$ の列の対 $s^* \trianglelefteq t_1 . \cdots . t_n$ を**照合対**とよび，照合対の有限集合を**照合問題**とよぶ．

2. **自明な**照合問題とは全ての要素 $s^* \trianglelefteq t^*$ について $s^* = t^*$ となる照合問題をいう．

3. 項準同型写像 φ と照合問題 S に対して $\varphi(S) = \{\varphi(s^*) \trianglelefteq t^* \mid s^* \trianglelefteq t^* \in S\}$ と定義する．$\varphi(S)$ が自明となるとき，φ を S の**照合解**とよぶ．

次に，照合問題 S に対して，その照合解を求める非決定的手続き **Match** を示す．

定義 4 (手続き Match) 1. 照合問題と項準同型写像の対 $\langle S, \varphi\rangle$ が表 1 中の推論規則 **Bound**，**Split**，**Empty**，**Extract**，**Bound-Seq** のいづれかを適用した結果 $\langle S', \varphi'\rangle$ となるとき，$\langle S, \varphi\rangle \Longrightarrow \langle S', \varphi'\rangle$ と記す．また，$\Longrightarrow$ の反射推移閉包を $\overset{*}{\Longrightarrow}$ で記す．

2. 非決定的手続き **Match**

入力： 照合問題 S
出力： $\langle S, \emptyset\rangle \overset{*}{\Longrightarrow} \langle \emptyset, \varphi\rangle$ となる項準同型写像 φ

例 2 (Match の動作例) 書き換え規則 $\mathsf{f}(\mathsf{c}(x, \overline{x}), \overline{y}) \to \mathsf{d}(\mathsf{f}(x, \overline{y}), \overline{x})$ と $\mathsf{length}(:(z.zs)) \to \mathsf{s}(\mathsf{length}(zs))$ の照合を行うには，$\{\mathsf{f}(\mathsf{c}(x, \overline{x}), \overline{y}) \trianglelefteq \mathsf{length}(:(z.zs)),\ \mathsf{d}(\mathsf{f}(x, \overline{y}), \overline{x}) \trianglelefteq \mathsf{s}(\mathsf{length}(zs))\}$ を **Match** への入力とすればよい．このときの動作例を表 2 に示す．

手続き **Match** の停止性，出力の有限性は手続きの正当性の証明に不可欠であり，また，計算機上に実装する際にも重要な問題である．**Match** の停止性を示すには $\Longrightarrow$ の整礎性を示せばよく，これは文献 [2] と同様に証明される．**Match** が停止することと入力される照合問題が有限集合であることから，**Match** による出力が有限個であることは容易に導かれる．手続き **Match** が停止することから，以降，手続き **Match** をアルゴリズム **Match** とよぶ．

次に，アルゴリズム **Match** の照合問題に対する健全性と完全性を示す．**Match** の健全性と完全性は，[2] と同様に証明される．

定理 1 (Match の健全性) S を照合問題とする．S を入力した **Match** の出力が φ であるとき，φ は S の解である．

次にアルゴリズム **Match** の完全性を示す．完全性の定式化のため完全照合集合の概念を導入する．

定義 5 (完全照合集合) 項準同型写像の集合 Φ が照合問題 S の**完全照合集合**であるとは次のときをいう．

(1) 任意の $\tilde{\varphi} \in \Phi$ は S の解である;
(2) 任意の S の解 φ に対して，$\tilde{\varphi} \subseteq \varphi$ となる $\tilde{\varphi} \in \Phi$ が存在する．

定理 2 (Match の完全性) S を照合問題とする．S を入力とする **Match** のすべての出力解の集合を Φ とするとき，Φ は S の完全照合集合である．

例 3 (パターンに基づくプログラム変換) 例 1 の変換パターン $\mathcal{P}$ と $\mathcal{R}_{\mathsf{length}}$ の照合解を得るには，それぞれの書き換え規則同士の照合問題を解けばよい．**Match** によって $\mathcal{P}$ と $\mathcal{R}_{\mathsf{length}}$ の照合解を求めると以下で示す $\varphi_{\mathsf{length}}$ が求められる．

$$\varphi_{\mathsf{length}} = \left\{ \begin{array}{lcl} \mathsf{f} & \mapsto & \mathsf{length}(\square_1) \\ \mathsf{a} & \mapsto & [\,] \\ \mathsf{c} & \mapsto & :(\square_1.\square_2) \\ \mathsf{b} & \mapsto & 0 \\ \mathsf{d} & \mapsto & \mathsf{s}(\square_1) \\ x & \mapsto & zs \\ \overline{x} & \mapsto & z \end{array} \right\}$$

$\varphi_{\mathsf{length}}(\mathcal{P}') = \mathcal{R}'_{\mathsf{length}}$ となるので，例 1 のパターンに基づくプログラム変換により $\mathcal{R}'_{\mathsf{length}}$ が得られる．

同様に $\mathcal{P}$ と $\mathcal{R}_{\mathsf{add3}}$ を照合すると以下の φ_{add3} が照合解として求められる．

$$\varphi_{\mathsf{add3}} = \left\{ \begin{array}{lcl} \mathsf{f} & \mapsto & \mathsf{add3}(\square_1.\square_2) \\ \mathsf{a} & \mapsto & 0 \\ \overline{y} & \mapsto & y.z \\ \mathsf{b} & \mapsto & +(\square_1) \\ \mathsf{c} & \mapsto & \mathsf{s}(\square_1) \\ \mathsf{d} & \mapsto & \mathsf{s}(\square_1) \\ x & \mapsto & w \end{array} \right\}$$

このとき，プログラム変換の出力として $\varphi_{\mathsf{add3}}(\mathcal{P}') = \mathcal{R}'_{\mathsf{add3}}$ が得られる．

5. 結論

本論文では，列変数の導入を行うことで，項書き換えシステムに基づくプログラム変換の枠組み [2] をより汎用性の高い枠組みへ拡張した．この枠組みを実現するため，列変数を用いた変換パターンの形式化を与えるとともに，列変数をもつ項パターンの照合アルゴリズムを提案し，その正当性を明らかにした．

現在，われわれは [2] の成果に基づき，プログラム変換システム RAPT を実装し，動作確認および実験をすすめている．本研究成果を RAPT に組み込んで有効性を評価することは今後の課題である．

参考文献

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

1. **Bound**

$$\frac{\langle \{x.\, s^* \trianglelefteq y.\, t^*\} \cup S,\ \varphi\rangle}{\langle \{s^* \trianglelefteq t^*\} \cup S,\ \{x \mapsto y\} \cup \varphi\rangle} \qquad \begin{array}{l}\varphi(x) = y,\ \text{または,}\\ x \notin \mathrm{dom}_{\mathscr{V}}(\varphi)\ \text{かつ}\ y \notin \mathrm{range}_{\mathscr{V}\cup\overline{\mathscr{V}}}(\varphi)\end{array}$$

2. **Split**

$$\frac{\langle \{f(s_1^*).\, s^* \trianglelefteq f(t_1^*).\, t^*\} \cup S,\ \varphi\rangle}{\langle \{s_1^* \trianglelefteq t_1^*,\ s^* \trianglelefteq t^*\} \cup S,\ \varphi\rangle} \qquad f \in \mathscr{F}$$

3. **Empty**

$$\frac{\langle \{\varepsilon \trianglelefteq \varepsilon\} \cup S,\ \varphi\rangle}{\langle S,\ \varphi\rangle}$$

4. **Extract**

$$\frac{\langle \{p(s_1, \ldots, s_n).\, s^* \trianglelefteq C\langle t_1^*, \ldots, t_n^*\rangle.\, t^*\} \cup S,\ \varphi\rangle}{\langle \{p \mapsto C\}(\{s_i \trianglelefteq t_i^* \mid \Box_i \in C\} \cup \{s^* \trianglelefteq t^*\} \cup S),\ \{p \mapsto C\} \cup \varphi\rangle} \quad \begin{array}{l} p \in \mathscr{P},\ C \in C_n(\mathscr{F}),\ \forall i \le n. s_i \notin \overline{\mathscr{V}} \Rightarrow |t_i^*| = 1\\ \forall i \le n. \forall x \mapsto y \in \varphi.\\ \quad \Box_i \in C \wedge y \in \mathscr{V}(t_i^*) \Rightarrow x \in \mathscr{V}(s_i),\ \text{かつ,}\\ \forall i \le n. \forall \overline{x} \mapsto y_1.\cdots.y_k \in \varphi.\\ \quad \Box_i \in C \wedge \exists j \le k\, (y_j \in \mathscr{V}(t_i)) \Rightarrow \overline{x} \in \overline{\mathscr{V}}(s_i)\end{array}$$

5. **Bound-Seq**

$$\frac{\langle \{\overline{x}.\, s^* \trianglelefteq x_1.\cdots.x_n.\, t^*\} \cup S,\ \varphi\rangle}{\langle \{s^* \trianglelefteq t^*\} \cup S,\ \{\overline{x} \mapsto x_1.\cdots.x_n\} \cup \varphi\rangle}\ n \ge 0,\ \text{かつ,} \left(\begin{array}{l}\varphi(\overline{x}) = x_1.\cdots.x_n,\ \text{または,}\\ \overline{x} \notin \mathrm{dom}_{\overline{\mathscr{V}}}(\varphi)\ \text{かつ}\ x_1, \ldots, x_n \notin \mathrm{range}_{\mathscr{V}\cup\overline{\mathscr{V}}}(\varphi)\end{array}\right)$$

表 1: 手続き **Match**

$$\begin{array}{rll}
 & \langle \{\mathsf{f}(\mathsf{c}(x,\overline{x}),\overline{y}) \trianglelefteq \mathsf{length}(:(z.zs)),\ \mathsf{d}(\mathsf{f}(x,\overline{y}),\overline{x}) \trianglelefteq \mathsf{s}(\mathsf{length}(zs))\},\ \emptyset\rangle & = \langle S_0, \varphi_0\rangle \\
\underset{\textbf{Extract}}{\Longrightarrow} & \langle \{\mathsf{c}(x,\overline{x}) \trianglelefteq :(z.zs),\ \mathsf{d}(\mathsf{length}(x),\overline{x}) \trianglelefteq \mathsf{s}(\mathsf{length}(zs))\},\ \varphi_0 \cup \{\mathsf{f} \mapsto \mathsf{length}(\Box_1)\}\rangle & = \langle S_1, \varphi_1\rangle \\
\underset{\textbf{Extract}}{\Longrightarrow} & \langle \{x \trianglelefteq zs,\ \overline{x} \trianglelefteq z,\ \mathsf{d}(\mathsf{length}(x),\overline{x}) \trianglelefteq \mathsf{s}(\mathsf{length}(zs))\},\ \varphi_1 \cup \{\mathsf{c} \mapsto :(\Box_2.\Box_1)\}\rangle & = \langle S_2, \varphi_2\rangle \\
\underset{\textbf{Bound}}{\Longrightarrow} & \langle \{\overline{x} \trianglelefteq z,\ \mathsf{d}(\mathsf{length}(x),\overline{x}) \trianglelefteq \mathsf{s}(\mathsf{length}(zs))\},\ \varphi_2 \cup \{x \mapsto zs\}\rangle & = \langle S_3, \varphi_3\rangle \\
\underset{\textbf{Bound-Seq}}{\Longrightarrow} & \langle \{\mathsf{d}(\mathsf{length}(x),\overline{x}) \trianglelefteq \mathsf{s}(\mathsf{length}(zs))\},\ \varphi_3 \cup \{\overline{x} \mapsto z\}\rangle & = \langle S_4, \varphi_4\rangle \\
\underset{\textbf{Extract}}{\Longrightarrow} & \langle \{\mathsf{length}(x) \trianglelefteq \mathsf{length}(zs)\},\ \varphi_4 \cup \{\mathsf{d} \mapsto \mathsf{s}(\Box_1)\}\rangle & = \langle S_5, \varphi_5\rangle \\
\underset{\textbf{Split}}{\Longrightarrow} & \langle \{x \trianglelefteq zs\},\ \varphi_5\rangle & = \langle S_6, \varphi_6\rangle \\
\underset{\textbf{Bound}}{\Longrightarrow} & \langle \emptyset,\ \varphi_6 \cup \{x \mapsto zs\}\rangle &
\end{array}$$

表 2: **Match** の動作例

[2] Y. Chiba, T. Aoto, and Y. Toyama. Program transformation by templates based on term rewriting. In *Proceedings of the 7th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming*, 2005. To appear.

[3] R. Curien, Z. Qian, and H. Shi. Efficient second-order matching. In *Proceedings of 7th International Conference on Rewriting Techniques and Applications*, Vol. 1103 of *LNCS*, pp. 317–331. Springer-Verlag, 1996.

[4] O. de Moor and G. Sittampalam. Higher-order matching for program transformation. *Theoretical Computer Science*, Vol. 269, pp. 135–162, 2001.

[5] M. Harao, S. Yin, K. Yamada, and K. Hirata. Efficiently computable classes of second order predicate schema. In *Proceedings of the 19th International Workshop on Unification*, 2005.

[6] K. Hirata, K. Yamada, and M. Harao. Tractable and intractable second-order matching problems. *Journal of Symbolic Computation*, Vol. 37, No. 5, pp. 611–628, 2004.

[7] G. Huet and B. Lang. Proving and applying program transformations expressed with second order patterns. *Acta Informatica*, Vol. 11, pp. 31–55, 1978.

[8] T. Kutsia. Solving equations involving sequence variables and sequence functions. In *Proceedings of the 7th International Conference on Artificial Intelligence and Symbolic Computation*, Vol. 3249 of *LNAI*, 2005.

[9] T. Kutsia and M. Marin. Can context sequence matching be used for XML querying. In *Proceedings of the 19th International Workshop on Unification*, 2005.

[10] T. Suzuki and S. Okui. A rewrite system with incomplete regular expression type for transformation of XML documents. 第 53 回プログラミング研究会 (PRO-2004-5), 2004.

[11] T. Yokoyama, Z. Hu, and M. Takeichi. Deterministic second-order patterns. *Information Processing Letters*, Vol. 89, No. 6, pp. 309–314, 2004.