非決定性 Streaming String Transducer と Parikh オートマトンを用いた文字列制約の充足可能性判定

釜野 雅基　福田 大我　南出 靖彦

正規表現や文字列操作関数などを含む論理式は文字列制約と呼ばれ，文字列を扱うプログラムの分析に現れる．本研究は構文的に制限された文字列制約の充足可能性を，トランスデューサを用いて判定する．Streaming String Transducer (SST) は文字列を別の文字列に変換するトランスデューサの一種であり，文字列を保持できる定数個の変数を使用して出力を計算する．また Parikh オートマトン (PA) は入力文字列から自然数のベクトルを計算し，それが予め決められた論理式を満たすかで受理を判定する．本研究は非決定性 SST による PA の逆像が計算できることを示す．さらにこれを応用し，正規表現とキャプチャグループを使う文字列置換や，文字列と整数を入力とする関数 substring を含む文字列制約の充足可能性が決定可能であることを示す．本研究はこの手法を文字列理論の SMT ソルバとして実装し，いくつかの制約を解く実験によって既存のソルバと比較した．

String constraints are logical formulas with regular expressions and string manipulating functions, and emerge in analysis of string manipulating programs. We develop a procedure to check satisfiability of syntactically-restricted string constraints with transducers. A streaming string transducer (SST) is a variant of a string-to-string transducer that computes an output with a fixed number of string variables. We also employ a Parikh automaton (PA), a string acceptor which computes a vector of natural numbers for an input and accepts if the vector satisfies a given arithmetic formula. We first show construction of the pre-image of a PA under an SST. Then we apply the result to show decidability of string constraints that may contain string replace functions using regular expressions with capture groups, and substring functions. The decision procedure is implemented as an SMT solver for string theory, and compared with existing solvers through an experiment on several constriants.

1　はじめに

本研究は文字列操作関数（トランスダクション）や正規言語への所属制約を含む論理式（文字列制約）の充足可能性判定を考える．文字列制約はプログラムの性質を考える上で役に立つことがある．例えば次の文字列制約は，2 つの文字列置換関数の合成が恒等関数であるとき，またそのときに限り充足不可能である：

$$x_1 = x_0.\mathit{replaceAll}(/(\mathtt{a}^+)_1(\mathtt{b}^+)_2/, \backslash 2 \backslash 1)$$
$$x_2 = x_1.\mathit{replaceAll}(/(\mathtt{b}^+)_1(\mathtt{a}^+)_2/, \backslash 2 \backslash 1)$$
$$x_0 \neq x_2$$

例えば一つ目の等式は x_0 に replaceAll 関数を適用した結果が x_1 に等しいことを意味する．この replaceAll は x_0 の値を左から走査し，正規表現 $(\mathtt{a}^+)_1(\mathtt{b}^+)_2$ にマッチする部分文字列を別の文字列へと置き換える．ここで括弧の下付き文字はキャプチャグループの名前であり，replaceAll はグループにマッチした文字列を使って置換文字列を構成する．制約は充足解 $x_0 = \mathtt{bab}$, $x_1 = \mathtt{bba}$, $x_2 = \mathtt{abb}$ を持つため，合成は恒等関数でないことが分かる．

算術については線形算術が決定可能であるのと対照的に，文字列制約のクラスは容易に決定不能となることが知られている．Z3-str [18] や CVC4 [15] など

Solving String Constraints with Nondeterministic Streaming String Transducers and Parikh Automata

Masaki Kamano, Taiga Fukuda, Yasuhiko Minamide, 東京工業大学情報理工学院, School of Computing, Tokyo Institute of Technology.

コンピュータソフトウェア, Vol.99, No.0 (2082), pp.1–20.
[研究論文] 2999 年 12 月 31 日受付.

は広い範囲の文字列制約をヒューリスティクスにより解く試みを行っている．これらは多くの制約を現実的な時間で決定できるものの，充足不可能な場合や正規表現を含む場合に必ずしも停止しないなど，完全性は保証されていない．

決定可能かつ十分に実用的な文字列制約のクラスを考えるための構文的な制限として直線性が知られている [14]．ある文字列制約が直線文字列制約であるためには，文字列の等式を代入文と考えたとき文字列変数の依存関係に循環が存在しないことが必要である．例えば前述の制約は直線性を満たす．直線文字列制約を対象としたソルバの一つとして Chen らによる OSTRICH がある [9]．OSTRICH はトランスダクションに関する正規言語の逆像を繰り返し計算して制約を判定する．また Zhu らは等式や正規表現を決定性 streaming string transducer (SST) と呼ばれる計算モデルで表すことで制約を解く方法を示した [19]．SST はそれらの逐次合成の空性が制約の充足不可能性と同値であるように構成される．この方法では含まれるトランスダクションが全て決定性 SST で表現できるような制約を解ける．

本研究は Zhu らの方法をより高い表現力を持つ制約のクラスへ拡張する．具体的にはトランスダクションを表現するための計算モデルとして，非決定性 SST [3] を採用する．また正規言語に加え，文字列から整数への関数を Parikh オートマトン [6] の変種を用いて表す．Parikh オートマトンは入力文字列から自然数のベクトルを計算するトランスデューサと線形算術論理式の組であり，計算したベクトルが論理式を満たすかで受理を判定する．さらに本研究はこれらを統合した計算モデル Parikh SST を導入する．Parikh SST は SST の文字列変換と，その出力の一部の Parikh 像をとる操作を同時に行うモデルと考えることができる．この仕組みは，非決定性と組み合わせることで，整数引数をとる操作を模倣するために利用される．Parikh SST は例えば文字列の連接に加え，部分文字列をとる操作 $substring\colon A^* \times \mathbb{Z}^2 \to A^*$ や，キャプチャグループを含む正規表現 r とグループ変数をもつ文字列 α につき r にマッチした文字列を α に置換する操作 $replaceAll_{r,\alpha}\colon A^* \to A^*$ を表現できる．この拡張のためには Parikh SST に関する Parikh オートマトンの逆像を Parikh オートマトンとして構成する必要がある．本稿はこの方法を詳細に定式化する．

本研究はこの手法を文字列理論の SMT ソルバとして実装した．ソルバは `https://github.com/minamide-group/expresso` で入手できる．また，このソルバでいくつかの制約を解く実験をおこない，OSTRICH や Zhu らのソルバと性能及び表現力を比較した．

ベクトルを出力するトランスデューサを使って文字列と整数を関連付けるアイデアは OSTRICH を拡張する研究 [8] で既に現れていた．本研究は先行研究がサポートするトランスダクションを表現できる計算モデルを見出し，この計算モデルの性質を利用して理解しやすい決定手続きへ整理したと言える．より詳細な比較は 7 節で議論する．

本稿はまず 2 節で文字列制約を定義する．3 節では Parikh オートマトンと Parikh SST を導入・例示し，4 節でその閉包性と決定可能問題を議論する．これらの性質を使って文字列制約を解く方法を 5 節で述べ，ソルバへの実装とその評価について 6 節に記す．

2 直線文字列制約

この節は整数制約で拡張された直線文字列制約を一般的な形で定義する．また，本研究の提案手法で扱える具体的なトランスダクションの例を示す．

A をアルファベット，$S = \{x_0, x_1, \cdots, x_{m-1}\}$ を添字で順序付けられた文字列変数の集合，I を整数変数の有限集合とする．y で文字列変数を，i で整数変数を，c で整数定数を表す．$f\colon A^* \times \mathbb{Z}^n \to \mathbb{Z}$ を整数値関数のメタ変数とするとき，線形算術項 t を $t ::= c \mid i \mid c \times t \mid t + t \mid f(y, t, \cdots, t)$ で定義し，t 上の線形算術論理式 φ_{int} を $\varphi_{int} ::= t \geq 0 \mid \neg\varphi_{int} \mid \varphi_{int} \wedge \varphi_{int}$ と定める．また，$T\colon (A^*)^{n_1} \times \mathbb{Z}^{n_2} \to 2^{A^*}$ を非決定性文字列値トランスダクションのメタ変数とするとき，文字列式 e は $e ::= T(y, \cdots, y, t, \cdots, t)$ で定義される．

本稿が対象とする直線文字列制約は $\varphi \equiv \varphi_{sl} \wedge \varphi_{reg} \wedge \varphi_{int}$ の形であり，各節は次の通りである．

- φ_{sl} は基礎直線制約 $\left(\bigwedge_{i=k}^{m-1} x_i{=}e_i\right)$，ただし $k \leq m$ かつ，各節の e_i は x_i 以上の文字列変数を含まない．これは $x_0, \cdots, x_{k-1}$ を引数とし，$x_k, \cdots, x_{m-1}$ の値を順に定義していくプログラムを表す．
- φ_{reg} は各文字列変数の正規言語への所属制約 $\left(\bigwedge_{i=0}^{m-1} x_i \in L_i\right)$
- φ_{int} は整数値項 t 上の線形算術論理式

附値 $\eta_S \colon S \to A^*$ と $\eta_I \colon I \to \mathbb{Z}$ のもとで論理式 ϕ が成り立つことを $\eta_S, \eta_I \models \phi$ と記す．ただし T に非決定性を許しているため，文字列等式の成立 $\eta_S, \eta_I \models x_i = e$ は η_S と η_I の下で x_i が e の表す集合に属することで定義する．

直線性は文字列制約を構文的に制限する．本研究はメタ変数 T や f で表した関数を，後の節で述べる Parikh SST や Parikh オートマトンと呼ぶ計算モデルのインスタンスに変換して扱うため，実際には扱える関数に対する意味論的な制限も持つ．表 1 にそのような変換が可能なトランスダクションとして主要なものを示した．表において x, y は文字列変数であってもよいが，w は文字列定数に限る．また，本稿は文字列を引数とするトランスダクションの多くを，文字列に定義されたメソッドのように記すことに注意されたい．例えば関数 $substr$ は型 $A^* \times \mathbb{Z} \times \mathbb{Z} \to A^*$ をもつ関数であるが，その適用は文字列変数 x と整数変数 i, l に対し $x.substr(i, l)$ のように記す．

最後に等号否定を含む制約への拡張について述べる．文字列変数間の等号否定を含む論理式 φ_{neg} を $\varphi_{neg} ::= y \neq y \mid \varphi_{neg} \wedge \varphi_{neg} \mid \varphi_{neg} \vee \varphi_{neg}$ で定義する．先行研究でも示されているように，この等号否定は整数を使う制約へとエンコードできる [8]．本研究は制約 $y_1 \neq y_2$ を整数制約 $y_1.codeAt(i) \neq y_2.codeAt(i)$ に変形する．ただし i は新しい整数変数である．この変換は充足可能性を変えないため，本研究の手法は $\varphi \wedge \varphi_{neg}$ に対しても使うことができる．

3 Parikh オートマトンと Parikh SST

この節は非決定性 SST と Parikh オートマトンの定義を例とともに確認し，さらに Parikh SST を導入する．そのためにまずモノイド出力トランスデューサを定義する．

3.1 モノイド出力トランスデューサ

集合 M に単位的結合的な二項演算 $\cdot$ が定義されているとき，$(M, \cdot)$ をモノイドと呼ぶ．演算が明らかなとき単に M と記す．また，モノイド M の単位元を 1_M と書く．

例 3.1 (自由モノイド)**.** 集合 A の要素を有限個並べた列の全体は連結を演算としてモノイドとなる．このモノイドを A^* で表し，その単位元を ε と記す．$w \in A^*$ に含まれる $a \in A$ の個数を $|w|_a$ で表す． □

例 3.2 (ベクトルモノイド)**.** 本稿におけるベクトルとは有限集合 L から可換モノイド M への関数である．例えば $L = \{l_1, l_2\}$ とするとき，型 $\mathbb{Z}^L$ を持つ関数はベクトルである．ベクトル M^L 全体は成分ごとの演算によって可換モノイドを成す． □

A を有限のアルファベット，$(M, \cdot)$ をモノイドとする．モノイド出力トランスデューサ $\mathcal{T}$ は A の元を読むごとに M の元を出力するトランスデューサであり，形式的には 4 つ組 (Q, Δ, q_0, q_f) で指定される．ここで Q は状態の有限集合，$\Delta \subseteq Q \times A_\$ \times M \times Q$ は遷移の有限集合，$q_0 \in Q$ は始状態, $q_f \in Q$ は終状態である．ただし $A_\$ = A \cup \{\$\}$, \$ は入力の終わりを表す記号で $\$ \notin A$ とする．遷移について $(q, a, m, r) \in \Delta$ を $q \xrightarrow[\Delta]{a/m} r$ と書く．これを $q \xrightarrow[\Delta]{\varepsilon/1_M} q$ と，$q \xrightarrow[\Delta]{w/m_1} q'$ かつ $q' \xrightarrow[\Delta]{a/m_2} q''$ ならば $q \xrightarrow[\Delta]{wa/m_1 \cdot m_2} q''$ によって $A_\$^*$ へと拡張する．$\mathcal{T}$ の意味は関数 $A^* \to 2^M$ であり，$m \in \mathcal{T}(w) \iff q_0 \xrightarrow[\Delta]{w\$/m} q_f$ とする．これは自然に A^*, M 上の関係（$A^* \times M$ の部分集合）とみなすこともできる．

以下では一般性を失わずに，q_f への遷移は \$ のみであり，q_f からの遷移は存在しないことを仮定する．この仮定を満たさないときは新たな状態 q_f' を加え，q_f への \$ による遷移を q_f' への遷移で置き換えればよい．また，\$ の存在により，始状態を複数の状態から非決定的に選ぶことを許しても表現力は変わらない．いま，トランスデューサ $\mathcal{T} := (Q, \Delta, Q_0, q_f)$ ただし $Q_0 \subseteq Q$ で $m \in \mathcal{T}(w) \iff \exists q_0 \in Q_0 \text{ s.t. } q_0 \xrightarrow[\Delta]{w\$/m} q_f$ としたものを考える．このとき新たに一意な始状態 q_0 を

表 1: 本研究のソルバでサポートされる代表的なトランスダクション．*substr* と *indexOf* の仕様は SMT-LIB [16] によるものである．

トランスダクション	説明	構成法
$x \cdot y$	x と y の連接.	例 3.4
$x.replaceAll_{r,\alpha}$	x 中の正規表現 r にマッチした文字列を キャプチャグループ変数を含む文字列 α で置換したもの.	文献 [12]
$x.reverse$	x の文字の並びを逆にしたもの.	例 3.3
$x.substr(i,l)$	x の i 文字目から最大で l 文字とったもの．$i<0$ なら空列.	例 3.6
$\|x\|$	x の長さ.	例 3.5
$x.indexOf_w(i)$	x の i 文字目以降で最初に現れる w の位置. w が現れないまたは $i<0$ なら -1.	附録 A. 1
$x.codeAt(i)$	x の i 番目の文字の文字コード．$i<0$ または $i \geq \|x\|$ なら -1.	

とり，Q_0 の元からの遷移を q_0 からの遷移として加えたモノイド出力トランスデューサは，$\mathcal{T}$ と同じトランスダクションを表す．以降（特に 4.2 節）では，この事実を前提として始状態が集合である機械を構成する場合がある．

3.2 Streaming String Transducer

Streaming String Transducer (SST) は Alur らによって提案された有限状態機械で，文字列から別の文字列への変換 (トランスダクション) を表現する [1] [2] [3]．機械は入力を先頭から末尾へと一方向的に読み，一文字読むごとに機械内部の有限な文字列変数を更新する．そして入力を読み終わった時点で，予め決めておいた変数 x_{out} の値を出力する．

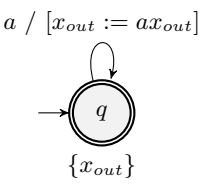
SST を定義するために，変数更新を定める．変数集合 X とアルファベット B に対して変数更新の集合を $\mathrm{Update}(X,B) = X \to (X \cup B)^*$ とする．$\alpha \in \mathrm{Update}(X,B)$ について $\alpha(x) = w$ とは x の内容を w で上書きすることを表す．$\mathrm{Update}(X,B)$ の元は自然に関数 $(X \cup B)^* \to (X \cup B)^*$ へと拡張できるため，$\mathrm{Update}(X,B)$ 上の演算 $\cdot$ を関数合成により $\alpha_1 \cdot \alpha_2 = \alpha_1 \circ \alpha_2$ と定義できる．このとき $(\mathrm{Update}(X,B), \cdot)$ は恒等関数を単位元とするモノイドを成す．

入力アルファベット A, 出力アルファベット B 上の非決定性 SST $\mathcal{S}$ を組 $(Q, X, \Delta, q_0, q_f, x_{out})$ と定義する．ここで X は変数集合，$x_{out} \in X$ であり，(Q, Δ, q_0, q_f) は $\mathrm{Update}(X,B)$ を出力するトランスデューサを成す．トランスデューサとしての意味 $A^* \to 2^{\mathrm{Update}(X,B)}$ に関数 $\mathrm{Update}(X,B) \ni \alpha \mapsto \varepsilon_X(\alpha(x_{out})) \in B^*$ を合成して得られる関数 $A^* \to 2^{B^*}$ を $\mathcal{S}$ の意味と定義する．ただし $\varepsilon_X : (X \cup B)^* \to B^*$ は変数を取り除く関数である．

変数更新 $\alpha \in \mathrm{Update}(X,B)$ が任意の $x \in X$ について $\sum_{y \in X} |\alpha(y)|_x \leq 1$ を満たすとき，α は copyless であるという．copyless な変数更新の全体は $\cdot$ で閉じている．任意の遷移の変数更新が copyless であるとき，その SST は copyless であるという．本稿では非決定性 copyless SST を考え，単に SST と書いたときはこれを意味する †1.

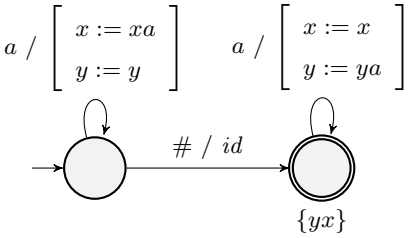
例 3.3. 文字列 $w = a_1 a_2 \cdots a_{n-1} a_n$ に対し $w.reverse = a_n a_{n-1} \cdots a_2 a_1$ とする．関数 *reverse* は次の SST が表現する．

†1 Alur が初めて SST を導入したとき，copyless であることは SST の本質的な条件として定義の中に含まれていた．一方，その後の研究で bounded-copy SST や copyful SST などの，copyless 性を仮定しない SST のクラスも提案されている [4] [11]．本稿の手法は SST が copyless であることに依存している (4.2 節)．



変数更新は [] で囲まれた部分であり，記法 $x := w$ は変数 x を w に更新することを表す．また q_f は図に記さず，\$ により q_f へ遷移するときの出力を状態下の {} 内に示す．SST においては x_{out} の値のみが最後の出力に影響するため，図で $\{w\}$ と書いたら $q \xrightarrow[\Delta]{\$/\alpha} q_f$, $\alpha(x_{out}) = w$ を意味する． □

例 3.4. A をアルファベットとし，$\# \notin A$ とする．次の SST はトランスダクション $A^*\#A^* \ni w_1\#w_2 \mapsto w_2w_1 \in A^*$ を表現する．



□

3.3 Parikh オートマトン

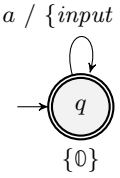
Parikh オートマトン [6] は文字と自然数ベクトルの組を読むオートマトンであり，実行で得られたベクトル部分の半線形集合への所属性によって受理する文字列を制限する．本稿ではその変種として，半線形集合（これは線形算術論理式で表現する）が整数パラメータを含むようなものを考える．

入力アルファベット A, 整数変数集合 I 上の Parikh オートマトン $\mathcal{A}$ とは組 $(Q, L, \Delta, q_0, q_f, \varphi)$ であり，$\mathcal{T} := (Q, \Delta, q_0, q_f)$ が自然数ベクトル $\mathbb{N}^L$ の加法モノイドを出力するトランスデューサを成す．φ は L, I を自由変数とする線形算術論理式であり，充足する附値の集合は関係 $R_\varphi \subseteq \mathbb{N}^L \times \mathbb{Z}^I$ を定める．モノイド出力トランスデューサとしての意味 $R_\mathcal{T} \subseteq A^* \times \mathbb{N}^L$ と R_φ を合成することにより得られる関係 $R_\mathcal{A} \subseteq A^* \times \mathbb{Z}^I$ を $\mathcal{A}$ の意味とする．より操作的には，$\mathcal{A}\colon \mathbb{Z}^I \to 2^{A^*}$ を任意の $\eta \in \mathbb{Z}^I$ に対し $w \in \mathcal{A}(\eta) \iff \exists v \in \mathbb{N}^L$ s.t. $v \in \mathcal{T}(w)$ かつ $\eta, v \models \varphi$ と定める．

例 3.5. I を整数変数の集合とし，$i \in I$ とする．φ_{int} の等式 $|x| = i$ からの変換を意図して，関係 $\mathrm{LENGTH}_i \subseteq A^* \times \mathbb{Z}^I$ を

$$\mathrm{LENGTH}_i = \{(w, \eta) \in A^* \times \mathbb{Z}^I \mid |w| = \eta(i)\}$$

と定義する．LENGTH_i は次の Parikh オートマトンが表現する．



$$L = \{input\},$$
$$\varphi = (i = input)$$

これは入力文字を読むごとに $input$ に 1 を加え，入力の終わりで $input$ が i に等しいか確認する． □

3.4 Parikh SST

本研究は SST と Parikh オートマトンを自然に統合したモデルとして Parikh SST を導入する．Parikh SST は SST の変数更新と自然数ベクトルを計算し，得られるベクトルと整数変数の間に課せられた論理式を満たす場合にのみ文字列を出力する．この仕組みによって，整数変数の値に応じて要求されうる出力を非決定的に計算し，その中から仕様を満たすものを選び出すことができる．つまり Parikh SST は整数変数の値に依存するトランスダクションを表現できる．

入力アルファベット A, 出力アルファベット B, 整数変数集合 I 上の Parikh SST $\mathcal{P}$ を 8 つ組 $(Q, X, L, \Delta, q_0, q_f, x_{out}, \varphi)$ とする．ここで X は変数集合，$x_{out} \in X$ であり，$\mathcal{T} := (Q, \Delta, q_0, q_f)$ は直積モノイド $\mathrm{Update}(X, B) \times \mathbb{N}^L$ を出力するトランスデューサを成す．トランスデューサとして定める関係 $R_\mathcal{T} \subseteq A^* \times \mathrm{Update}(X, B) \times \mathbb{N}^L$ に関数 $\mathrm{Update}(X, B) \ni \alpha \mapsto \varepsilon_X(\alpha(x_{out})) \in B^*$ を合成して関係 $R'_\mathcal{T} \subseteq A^* \times B^* \times \mathbb{N}^L$ を得る．さらに論理式の定める関係 $R_\varphi \subseteq \mathbb{N}^L \times \mathbb{Z}^I$ を合成して得られる $R_\mathcal{P} \subseteq A^* \times B^* \times \mathbb{Z}^I$ を $\mathcal{P}$ の意味とし，自然に型 $\mathbb{Z}^I \to A^* \to 2^{B^*}$ なるトランスダクションと同一視

する．本稿では Parikh SST についても任意の文字列変数更新が copyless であることを仮定する．

例 3.6. I を整数変数の集合とし，$i,l \in I$ とする．φ_{sl} の等式 $y = x.substr(i,l)$ からの変換を意図して，トランスダクション $\mathrm{SUBSTR}_{i,l}\colon \mathbb{Z}^I \to A^* \to A^*$ を次のように定義する．$w = a_0 \cdots a_{n-1} \in A^*$, $\eta \in \mathbb{Z}^I$ とするとき，$0 \le \eta(i) < \eta(i) + \eta(l) \le n$ なら

$$\mathrm{SUBSTR}_{i,l}(\eta)(w) = a_{\eta(i)} \cdots a_{\eta(i)+\eta(l)-1},$$

$0 \le \eta(i) < n < \eta(i) + \eta(l)$ なら

$$\mathrm{SUBSTR}_{i,l}(\eta)(w) = a_{\eta(i)} \cdots a_{n-1},$$

それ以外のとき $\mathrm{SUBSTR}_{i,l}(\eta)(w) = \varepsilon$. $\mathrm{SUBSTR}_{i,l}$ は図 1 の Parikh SST で表現される． □

4 Parikh オートマトンの性質

この節では文字列制約を解くにあたって必要となる Parikh オートマトンの性質について述べる．具体的には言語の共通部分や SST による逆像を定義し，これらの操作に関して Parikh オートマトンが閉じていることを構成によって示す．本研究の Parikh オートマトンは先行研究と異なり整数パラメータ I を持つため，言語の共通部分やトランスダクションに関する逆像の概念も I が関わるように定義される．SST による逆像の構成法は本研究の主たる成果の一つである．そのほか，言語の空性判定が線形算術論理式に帰着できることも説明する．共通部分の構成法や空性判定の帰着法は先行研究 [13] に見られるものと本質的に同じであるが，整数パラメータの存在による細かな差異があるため本稿でも改めて説明する．

4.1 共通部分

3 つの Parikh オートマトン

$$\mathcal{A} = (Q_1, L_1, \Delta_1, q_0^1, q_f^1, \varphi_1)\colon \mathbb{Z}^I \to 2^{A^*},$$
$$\mathcal{B} = (Q_2, L_2, \Delta_2, q_0^2, q_f^2, \varphi_2)\colon \mathbb{Z}^I \to 2^{A^*},$$
$$\mathcal{C} = (Q, L, \Delta, q_0, q_f, \varphi)\colon \mathbb{Z}^I \to 2^{A^*}$$

について，$\mathcal{C}$ が $\mathcal{A}$ と $\mathcal{B}$ の共通部分を表すとは，任意の $\eta \in \mathbb{Z}^I$ に対して $\mathcal{C}(\eta) = \mathcal{A}(\eta) \cap \mathcal{B}(\eta)$ を満たすことをいう．$\mathcal{A}, \mathcal{B}$ が与えられたとき，この $\mathcal{C}$ を次のように直積構成できる：

- $Q = Q_1 \times Q_2$, $q_0 = (q_0^1, q_0^2)$, $q_f = (q_f^1, q_f^2)$,
- $L = L_1 \cup L_2$,
- $q_1 \xrightarrow[\Delta_1]{a/v_1} r_1$ かつ $q_2 \xrightarrow[\Delta_2]{a/v_2} r_2$ ならば $(q_1, q_2) \xrightarrow[\Delta]{a/v_1 \oplus v_2} (r_1, r_2)$, ただし $(v_1 \oplus v_2)(l) = \mathit{if}\ l \in L_1\ \mathit{then}\ v_1(l)\ \mathit{else}\ v_2(l)$
- $\varphi = \varphi_1 \wedge \varphi_2$.

4.2 SST による逆像

SST $\mathcal{S}\colon A^* \to 2^{B^*}$ と Parikh オートマトン $\mathcal{A}\colon \mathbb{Z}^I \to 2^{B^*}$ に対して，$\mathcal{S}$ による $\mathcal{A}$ の逆像 $\mathcal{S}^{-1}(\mathcal{A})\colon \mathbb{Z}^I \to 2^{A^*}$ とは任意の $\eta \in \mathbb{Z}^I$ と $w \in A^*$ に対して “$w \in \mathcal{S}^{-1}(\mathcal{A})(\eta) \iff \exists w'$ s.t. $w' \in \mathcal{S}(w)$ かつ $w' \in \mathcal{A}(\eta)$” を満たすものをいう．この節では $\mathcal{S}^{-1}(\mathcal{A})$ が Parikh オートマトンで表せることを示す．そのためには $\mathcal{A}$ を $\mathbb{N}^L$ 出力トランスデューサとみなしたときの合成 $\mathcal{A} \circ \mathcal{S}$ がまた $\mathbb{N}^L$ 出力トランスデューサであればよい．実際，この合成に $\mathcal{A}$ の論理式を加えることで $\mathcal{S}^{-1}(\mathcal{A})$ を表す Parikh オートマトンになる．

以下 $(M, \cdot)$ を可換モノイドとし，より一般的な問題を考える．$\mathcal{T}$ を入力アルファベット B 上の M 出力トランスデューサとする．いま，次を満たす M 出力トランスデューサ $\mathcal{U}\colon A^* \to 2^M$ を構成する：

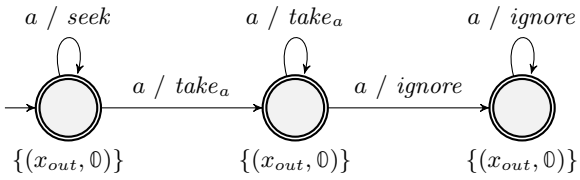
命題 4.1. 任意の $w \in A^*$ に対して $\mathcal{U}(w) = \mathcal{T}(\mathcal{S}(w))$. ただし $\mathcal{T}(\mathcal{S}(w))$ は $\bigcup_{w' \in \mathcal{S}(w)} \mathcal{T}(w')$ を意味する．

構成は次の二段階に分けられる．

1. 合成と等価な M 出力 SST (後述) $\mathcal{M}$ を構成する
2. $\mathcal{M}$ を等価な M 出力トランスデューサ $\mathcal{U}$ へと変換する

第一段階は Alur [1] による，決定性 SST の逐次合成の構成法のアイデアを非決定性 SST に適用したものである．また第二段階は，加賀江ら [12] による SST の出力文字列長の差をカウントする 1 カウンタ機械の構成法や，Zhu [19] らによる SST の Parikh 像を出力するトランスデューサの構成法の一般化になっている．附録 B には各段階の合成例を示した．

なお各段階の方法はそれぞれ独立に用いることができる．すなわち，第一段階は任意の copyless SST と M 出力トランスデューサから，それらの合成と等価な M 出力 (copyless) SST を構成する．ここで，



$$X = \{x_{out}\},\ L = \{input, index, output\},\ a \in A$$
$$seek = ([x_{out} := x_{out}], \{input : 1, index : 1, output : 0\})$$
$$take_a = ([x_{out} := x_{out}a], \{input : 1, index : 0, output : 1\})$$
$$ignore = ([x_{out} := x_{out}], \{input : 1, index : 0, output : 0\})$$
$$\varphi_{\mathrm{f}} = \bigwedge \begin{cases} i < 0 \vee i \geq input \vee l \leq 0 \Rightarrow output = 0, \\ 0 \leq i < input \wedge l \leq input - i \Rightarrow index = i \wedge output = l, \\ 0 \leq i < input \wedge l > input - i \Rightarrow index = i \wedge output = input - i \end{cases}$$

図 1: $\mathrm{SUBSTR}_{i,l}$ を表現する Parikh SST. *input* には入力文字列長を，*index* には読み飛ばした文字列の長さを，*output* には出力として変数 x_{out} に格納した文字列の長さを記録する．論理式は，指定されたインデックス i が有効であるとき，読み飛ばした文字列長が i に，出力の長さが指定された長さと残りの文字列の長さとの最小値にそれぞれ一致することを確認する．また，指定されたインデックスまたは長さが有効でなければ，出力は空列であることを保証する．

M は非可換なモノイドでも良い．また第二段階は任意の可換モノイド M に対して，与えられた M 出力 copyless SST と等価な M 出力トランスデューサを構成する方法になっている．

4.2.1 第一段階

M 出力 SST は文字列の代わりにモノイド M の要素を出力する SST であり，組 $(Q, X, \Delta, q_0, q_f, x_{out})$ と定義される．ここで組 (Q, Δ, q_0, q_f) は $\mathrm{Update}(X, M)$ 出力トランスデューサを成す．すなわち $\Delta \subseteq Q \times A_{\$} \times \mathrm{Update}(X, M) \times Q$ を満たす．$\mathrm{Update}(X, M) \ni \alpha \mapsto \tau(\alpha(x_{out})) \in M$ によって意味を $A^* \to 2^M$ に修正する．ただし $\tau\colon (X \cup M)^* \to M$ は変数を無視して M の積を計算する関数である．M 出力 SST に対しても，copyless 性が SST と同様に定義される．

Copyless SST $\mathcal{S} = (Q_1, X, \Delta_1, q_0^1, q_f^1, x_{out})$ と M 出力トランスデューサ $\mathcal{T} = (Q_2, \Delta_2, q_0^2, q_f^2)$ が与えられたとする．まず合成と等価な M 出力 copyless SST $\mathcal{M}\colon A^* \to 2^M$ を構成する．

状態集合 Q, 始状態集合 Q_0, 終状態 q_f は

- $Q = Q_1 \times (X \rightharpoonup Q_2 \times Q_2)$
- $Q_0 = \{(q_0^1, f_0) \mid \forall x, q, r. f_0(x) = (q, r) \Rightarrow q = r\}$
- $q_f = (q_f^1, f_{\$})$ ただし $dom(f_{\$}) = \{x_{out}\}$ かつ $f_{\$}(x_{out}) = (q_0^2, q_f^2)$

とする．ここで $X \rightharpoonup Q_2 \times Q_2$ は X から $Q_2 \times Q_2$ への部分関数を意味する．$(q_1, f) \in Q$ の内，$q_1 \in Q_1$ は直積構成と同様に $\mathcal{S}$ を模倣する．$f\colon X \rightharpoonup Q_2 \times Q_2$ について，$f(x) = (q_2, r_2) \in Q_2 \times Q_2$ は，$\mathcal{M}$ が模倣している $\mathcal{S}$ の計算状況 (q_1, η) における x の値 $\eta(x)$ が $\mathcal{S}$ の出力文字列で使用され，その値によって $\mathcal{T}$ の状態が q_2 から r_2 に遷移することを表す．ただし SST の計算状況とは実行 $q_0^1 \xrightarrow[\Delta_1]{w/\alpha} q_1$ に対してペア $(q_1, \varepsilon_X \circ \alpha)$ のことを指す．$\mathcal{S}$ の初期計算状況 (q_0^1, ε_X) において各変数 $x \in X$ には $\varepsilon_X(x) = \varepsilon$ が入っており，これは $\mathcal{T}$ の状態を変化させない．$\mathcal{M}$ の始状態の $f_0\colon X \rightharpoonup Q_2 \times Q_2$ に課せられる条件にこのことが反映されている．

より詳しく，入力文字列 $w = w'w'' \in A^*$ に対して $\mathcal{S}$ と $\mathcal{T}$ がそれぞれ $q_0^1 \xrightarrow[\Delta_1]{w'/\alpha_1} q_1 \xrightarrow[\Delta_1]{w''\$/\alpha_2} q_f^1$, $q_0^2 \xrightarrow[\Delta_2]{(\varepsilon_X \circ \alpha_1)(\alpha_2(x_{out}))\$/m} q_f^2$ のように遷移する場合を

考える．$\mathcal{M}$ は w を一方向的に読みながらこれら $\mathcal{S}$ と $\mathcal{T}$ の動作を同時に模倣したい．w' まで読んだ時の $\mathcal{S}$ の状態は q_1 で変数 x には $(\varepsilon_X \circ \alpha_1)(x)$ が保存されているが，これらについては $\mathcal{M}$ も容易に模倣できる．しかし各 $(\varepsilon_X \circ \alpha_1)(x)$ は出力文字列を構成するための断片であり，w 全体を読み終えるまでこれら断片がどのように組み合わせられるか $(\alpha_2(x_{out}))$ が確定しない．そこで $\mathcal{M}$ は f と非決定性を使って，$\mathcal{S}$ のどの変数がどう $\mathcal{T}$ によって読まれるかを推測し，各断片に対して $\mathcal{T}$ が出力する M の元を変数に保持する．$\mathcal{T}$ が読む文字列を保存している変数の集合は $\mathrm{Var}(\alpha_2(x_{out}))$ （ただし $\mathrm{Var}\colon (X \cup B)^* \to 2^X$ は文字列に出現する変数の集合を返す関数）であるが，これを $dom(f)$ によって推測する．いま $\alpha_2(x_{out}) = w_0 x_1 w_1 \cdots x_k w_k$, $w_i \in B^*$, $x_i \in X$ とおくと，各 x_i の値 $v_i := (\varepsilon_X \circ \alpha_1)(x_i)$ によって $\mathcal{T}$ の遷移は $q_0^2 = r_2^0 \xrightarrow[\Delta_2]{w_0} q_2^1 \xrightarrow[\Delta_2]{v_1} r_2^1 \xrightarrow[\Delta_2]{w_1} \cdots \xrightarrow[\Delta_2]{w_k\$} q_2^{k+1} = q_f^2$ と分解できる（出力する M の元は省略）．このように v_i が $\mathcal{T}$ の状態を q_2^i から r_2^i に遷移させる，ということを $f(x_i) = (q_2^i, r_2^i)$ で推測する．

注意 4.2. Alur らによる決定性 SST 同士の合成の定式化[1] では，決定性のものが構成される．決定性では $dom(f)$ すなわち「出力文字列の構成で使われる変数の集合」を推測できないため，彼らは本研究よりも「大きい」情報を管理することで決定性の機械を構成した．構成される状態は関数 $f\colon Q_2 \times X \to Q_2$ を持ち，これが任意の $q \in Q_2$ と $x \in X$ に対して「$\mathcal{T}$ が x に格納された文字列を状態 q で読んだ時の遷移先の状態 $f(q, x)$」を記憶する．本研究は非決定性を許すことで管理する情報のサイズを削減しているため，$\mathcal{S}$ と $\mathcal{T}$ の両方が決定性であったとしても $\mathcal{M}$ は一般に非決定性となる． □

遷移集合は以上の直観を反映して，次のようにする．まず，$f\colon X \rightharpoonup Q_2 \times Q_2$ に対して Δ_2 を $\Delta_2^{(f)} \subseteq Q_2 \times (X \cup B_\$) \times (X \cup M) \times Q_2$ へと拡張する:

$$q_2 \xrightarrow[\Delta_2]{a/m} q_2' \text{ なら } q_2 \xrightarrow[\Delta_2^{(f)}]{a/m} q_2',$$
$$f(x) = (q_2, q_2') \text{ なら } q_2 \xrightarrow[\Delta_2^{(f)}]{x/x} q_2'.$$

その上で，$q_1, q_1' \in Q_1$, $f, f'\colon X \rightharpoonup Q_2 \times Q_2$, $a \in A_\$$, $\alpha' \in \mathrm{Update}(X, M)$ について，$(q_1, f) \xrightarrow[\Delta]{a/\alpha'} (q_1', f')$ が次と同値であるように Δ を定義する：ある $\alpha \in \mathrm{Update}(X, B)$ が存在して，$q_1 \xrightarrow[\Delta_1]{a/\alpha} q_1'$ かつ次の 3 つを全て満たす．

- $dom(f) = \cup_{x \in dom(f')} \mathrm{Var}(\alpha(x))$,
- 任意の $x \in dom(f')$ について，$f'(x) = (q_2, q_2')$ とするとき $q_2 \xrightarrow[\Delta_2^{(f)}]{\alpha(x)\$_a/\alpha'(x)} q_2'$, ただし $\$_a = \mathit{if}\ (a = \$)\ \mathit{then}\ \$\ \mathit{else}\ \varepsilon$,
- 任意の $x \notin dom(f')$ について $\alpha'(x) = \varepsilon$.

これらの条件を説明するために，入力文字列 $w = w'aw'' \in A^*$ $(a \in A)$ に対する $\mathcal{S}$ の状態遷移が $q_0^1 \xrightarrow[\Delta_1]{w'/\alpha_1} q_1 \xrightarrow[\Delta_1]{a/\alpha} q_1' \xrightarrow[\Delta_1]{w''\$/\alpha_2} q_f^1$ である場合を考える．これ（と $\mathcal{S}$ の出力文字列に対する $\mathcal{T}$ の動作）を模倣する $\mathcal{M}$ の実行が $(q_0^1, f_0) \xrightarrow[\Delta_1]{w'/\alpha_1'} (q_1, f) \xrightarrow[\Delta_1]{a/\alpha'} (q_1', f') \xrightarrow[\Delta_1]{w''\$/\alpha_2'} (q_f^1, f_\$)$ であるとし，このうち入力文字 a による遷移に注目する．f と f' の定義域が $\mathrm{Var}(\alpha(\alpha_2(x_{out})))$ と $\mathrm{Var}(\alpha_2(x_{out}))$ をそれぞれ推測するのであったから，第一の条件が成り立つべきだとわかる．また x に対し $f'(x)$ が定義されているとき，これは $w'a$ を読んだ後の $\mathcal{S}$ における x の値 $(\varepsilon_X \circ \alpha_1)(\alpha(x))$ が引き起こす $\mathcal{T}$ の状態遷移を表すのだった．いま $\alpha(x) = v_0 y_1 v_1 \cdots y_k v_k$, $v_i \in B^*$, $y_i \in X$ とおくと $(\varepsilon_X \circ \alpha_1)(\alpha(x)) = v_0 w_1' v_1 \cdots w_k' v_k$, $w_i' = (\varepsilon_X \circ \alpha_1)(y_i)$ であり，各 w_i' は w' まで読んだ時点での y_i の値である．w_i' が引き起こす $\mathcal{T}$ の状態遷移は $f(y_i)$ によって推測されているので，$f'(x)$ の値はこれと矛盾しない必要がある．第二の条件はこれらの推測の整合性を表している．

第二の条件において，入力文字 $a = \$$ の場合は次の理由から特別扱いしている．$\mathcal{S}$ の出力は $\mathcal{S}$ の x_{out} に格納されるが，これを $\mathcal{T}$ に入力するときは後ろに $\$$ を加える．$\mathcal{M}$ はこれに対する $\mathcal{T}$ の動作を模倣しなければならない．そこで仮想的に，$\mathcal{S}$ の x_{out} には本来の $\alpha(x_{out})$ の代わりに $\alpha(x_{out})\$$ が代入されると考えている．これは $f_\$$ の定義にも反映されている．

$\mathcal{M}$ の任意の変数更新は $\mathcal{S}$ の対応する更新を元にしているため，copyless 性が保存される．また，第二・第三の条件により遷移集合が有限になる．

構成される $\mathcal{M}$ は次を満たす．

命題 4.3. 任意の $w \in A^*$ に対して $\mathcal{M}(w) = \mathcal{T}(\mathcal{S}(w))$.

注意 4.4. ここまで述べた構成法は M が非可換の場合にも適用可能であり，やはり命題 4.3 が成り立つ．実際，この命題や後述の補題の証明において，M が可換であることは使っていない． □

証明には以下の 2 つの補題を用いる．いずれも，u が長さ 1 である場合を示せば，長さに関する帰納法で一般の $u \in A^*$ に対して示せる．命題と補題の証明は附録 C に記す．

補題 4.5. $u \in A^*$ について $(q_1, f) \xrightarrow[\Delta]{u/\alpha'} (q_1', f')$ であるとき，$\alpha \in \mathrm{Update}(X, B)$ が存在して，$q_1 \xrightarrow[\Delta_1]{u/\alpha} q_1'$ 及び次の条件を満たす：任意の $q_2, q_2' \in Q_2$, $w \in (X \cup B)^*$, $w' \in (X \cup M)^*$ に対して

$$q_2 \xrightarrow[\Delta_2^{(f')}]{w/w'} q_2' \text{ ならば } q_2 \xrightarrow[\Delta_2^{(f)}]{\alpha(w)/\alpha'(w')} q_2'$$

が成り立つ．

補題 4.6. $w \in (X \cup B)^*$ は重複する変数の出現を持たないとする．$u \in A^*$ について $q_1 \xrightarrow[\Delta_1]{u/\alpha} q_1'$ かつ $q_2 \xrightarrow[\Delta_2^{(f)}]{\alpha(w)/w''} q_2'$ かつ $dom(f) = \mathrm{Var}(\alpha(w))$ であるとき，ある $\alpha' \in \mathrm{Update}(X, M)$, $f' \colon X \rightharpoonup (Q_2 \times Q_2)$, $w' \in (X \cup M)^*$ が存在して，$(q_1, f) \xrightarrow[\Delta]{u/\alpha'} (q_1', f')$ かつ $q_2 \xrightarrow[\Delta_2^{(f')}]{w/w'} q_2'$ かつ $\alpha'(w') = w''$ かつ $dom(f') = \mathrm{Var}(w)$ が成り立つ．

4.2.2 第二段階

M 出力 SST $\mathcal{M}$ を等価な M 出力トランスデューサ $\mathcal{U}$ に変換することで合成を完了する．この変換は M の可換性を利用したものになっている．そのため，以下では M の演算を加法とする．

変数集合 X 上の全順序を任意に固定し，この順序で i 番目に小さい変数を x_i と記す．$\Phi(\alpha)$ を $\mathbb{N}^{|X|\times|X|}$ の行列で，i 行 j 列の成分が $\alpha(x_j)$ に現れる x_i の個数であるようなものと定義する．すなわち，$\Phi(\alpha)$ の第 i 行は x_i の情報がどの変数に何回コピーされるかを表す．また $\tau \colon (X \cup M)^* \to M$ は変数を無視して M の積を計算する関数であった．これを $\alpha \mapsto (x \mapsto \tau(\alpha(x)))$ によって $\mathrm{Update}(X, M) \to M^X$ へと拡張する．$u \in \mathbb{N}^X$, $\tau(\alpha)$ をそれぞれ $\mathbb{N}^{|X|}$ の縦ベクトル，$M^{|X|}$ の横ベクトルと自然に同一視し，$\tau(\alpha) \cdot u \in M$ を内積で定義する．

$\mathcal{M} = (Q, X, \Delta, q_0, q_f, x_{out})$ を M 出力 SST とする．等価な M 出力トランスデューサ $\mathcal{U} = (Q', \Delta', Q_0, q_f')$ の状態集合は $Q' = Q \times 2^X$，終状態は $q_f' = (q_f, \{x_{out}\})$, 始状態集合は $Q_0 = \{(q_0, u) \mid u \subseteq X\}$ と構成する．ここで Q' の第二成分 $u \in 2^X$ は出力において各変数の値が使用されるかどうかを表す．また $u(x) = 1 \iff x \in u$ によって $u \colon X \to \{0, 1\}$ とみなす．遷移集合 $\Delta' \subseteq Q' \times A_\$ \times M \times Q'$ は，$q \xrightarrow[\Delta]{a/\alpha} r$, $a \in A_\$$ であるとき任意の $u \in 2^X$ $(\subseteq \mathbb{N}^X)$ について $(q, \Phi(\alpha) \cdot u) \xrightarrow[\Delta']{a/\tau(\alpha)\cdot u} (r, u)$ とする．$\mathcal{M}$ が copyless であるとき，$\Phi(\alpha) \cdot u$ は再び 2^X の元である．

遷移の構成における $\tau(\alpha) \cdot u$ に注目しよう．これは各 x について，α によって x に加えられるベクトル $\tau(\alpha(x))$ を，x が実際に出力で使われる回数 $u(x)$ だけ出力することを言っている．また，状態の第二成分が表す推測も，遷移の前後において整合している．

命題 4.7. 任意の $w \in A^*$ に対して $\mathcal{U}(w) = \mathcal{M}(w)$.

注意 4.8. 変換の対象となった $\mathcal{M}$ は前節で構成されたものでなくても良い．M が可換モノイドであるような M 出力 copyless SST $\mathcal{M}$ であれば，この節の方法で構成された $\mathcal{U}$ は命題 4.7 を満たす． □

この命題は下記の補題 4.10 から直ちに証明される．なお，同様の補題が Zhu ら [19] によっても証明されている．

補題 4.9. 任意の $\alpha_1, \alpha_2 \in \mathrm{Update}(X, M)$ に対して

- $\Phi(\alpha_1 \circ \alpha_2) = \Phi(\alpha_1) \cdot \Phi(\alpha_2)$,
- $\tau(\alpha_1 \circ \alpha_2) = \tau(\alpha_1) \cdot \Phi(\alpha_2) + \tau(\alpha_2)$

が成り立つ．

Proof. 定義より明らか． □

補題 4.10. $q \xrightarrow[\Delta]{w/\alpha} r$ ならば任意の $u \in 2^X$ に対して $(q, \Phi(\alpha) \cdot u) \xrightarrow[\Delta']{w/\tau(\alpha)\cdot u} (r, u)$ が成り立つ．逆に，$(q, v) \xrightarrow[\Delta']{w/m} (r, u)$ ならばある $\alpha \in \mathrm{Update}(X, M)$ が存在して $q \xrightarrow[\Delta]{w/\alpha} r$ かつ $v = \Phi(\alpha) \cdot u$, $m = \tau(\alpha) \cdot u$ が成り立つ．

Proof. $\mathcal{U}$ の構成と補題 4.9 から，w の長さに関する

帰納法で容易に示される．□

注意 4.11. M 出力 SST の意味では x_{out} の値のみを出力と考えた．代わりに全ての変数の値が出力だと考えれば，SST はベクトル M^X を出力するトランスダクションを表現する．このときもすべての変数更新が copyless であれば，この節の方法を拡張することで等価な M^X 出力トランスデューサへ変換できる．具体的には，$u \in 2^X$ としていた部分を行列 $U \in 2^{X\times X}$ とし，(i,j) 成分で x_j に x_i がコピーされるかどうかを表すように変更すればよい．

$M = \mathbb{N}$ のとき SST の変数更新 α は Affine 変換 $\mathbb{N}^X \ni v \mapsto v \cdot \Phi(\alpha) + \tau(\alpha) \in \mathbb{N}^X$ と本質的に同じ意味を持つ (v は横ベクトル)．このように遷移ごとの Affine 変換によってベクトルを計算し，Parikh オートマトンと同様に線形算術論理式で受理を判定する計算モデルは Affine Parikh オートマトン (APA) [6] と呼ばれる．以上の議論は copyless な APA が Parikh オートマトンへと変換できることを含意する．□

4.3 Parikh SST による逆像

Parikh SST $\mathcal{P}\colon \mathbb{Z}^I \to A^* \to 2^{B^*}$ と Parikh オートマトン $\mathcal{A}\colon \mathbb{Z}^I \to 2^{B^*}$ が与えられたとき，$\mathcal{P}$ による $\mathcal{A}$ の逆像 $\mathcal{P}^{-1}(\mathcal{A})\colon \mathbb{Z}^I \to 2^{A^*}$ とは任意の $\eta \in \mathbb{Z}^I$ と $w \in A^*$ について "$w \in (\mathcal{P}^{-1}(\mathcal{A}))(\eta) \iff \exists w'$ s.t. $w' \in \mathcal{P}(\eta)(w)$ かつ $w' \in \mathcal{A}(\eta)$" を満たすものをいう．$\mathcal{P}^{-1}(\mathcal{A})$ は Parikh オートマトンとして構成できる．

構成は SST による逆像 (4.2 節) と同様である．ただし $\mathcal{P}$ は Parikh オートマトンのようにベクトルを計算し論理式を評価するため，構成される $\mathcal{P}^{-1}(\mathcal{A})$ は Parikh オートマトンの共通部分 (4.1 節) のようにそれぞれが出力するベクトルの直和を計算する必要がある．構成の詳細は附録 D に記す．

4.4 空性検査

Parikh オートマトン $\mathcal{A}\colon \mathbb{Z}^I \to 2^{A^*}$ が空であるとは，任意の $\eta \in \mathbb{Z}^I$ に対して $\mathcal{A}(\eta) = \varnothing$ であることを言う．Parikh オートマトンの空性は線形算術論理式へと帰着できるため決定可能である．

命題 4.12. Parikh オートマトン $\mathcal{A} = (Q, L, \Delta, q_0, q_f, \varphi)\colon \mathbb{Z}^I \to 2^{A^*}$ に対して，$L \cup I$ を自由変数とする線形算術論理式 ψ を任意の $\eta \in \mathbb{Z}^I$ に対して $\exists v \in \mathbb{N}^L$ s.t. $v, \eta \models \psi \iff \mathcal{A}(\eta) \neq \varnothing$ であるように構成できる．より強く，ψ は $v, \eta \models \psi \iff \exists w \in A^*$ s.t. $q_0 \xrightarrow[\Delta]{w\$/v} q_f$ かつ $v, \eta \models \varphi$ を満たす．また，ψ は $\mathcal{A}$ に対して線形サイズである．

Proof. 次の補題の ϕ を用いて $\psi = \phi \wedge \varphi$ とすれば良い．□

補題 4.13. $\mathbb{N}^L$ 出力トランスデューサ $\mathcal{T} = (Q, \Delta, q_0, q_f)\colon A^* \to 2^{\mathbb{N}^L}$ に対して，L を自由変数とする線形算術論理式 ϕ を次のように構成できる：任意の $v \in \mathbb{N}^L$ について $v \models \phi \iff \exists w \in A^*$ s.t. $v \in \mathcal{T}(w)$ が成り立つ．

Proof. 有限オートマトンを右線形文法に変換するのと同様の方法で，$\mathcal{T}$ を $\mathbb{N}^L$ をアルファベットとする文法に変換できる．この文法に対して，Verma ら [17] による，文脈自由言語の Parikh 像を捉えた論理式の構成法を（簡単な修正を加えることで）利用できる．□

5 文字列制約の決定手続き

この節では 2 節で定義した文字列制約の充足可能性を，4 節で述べた性質を使って判定する方法を示す．

直線文字列制約 $\varphi = \varphi_{sl} \wedge \varphi_{reg} \wedge \varphi_{int}$ は整数変数と等式を追加することにより次の仮定を満たすように変形できる．以下ではこの形のみを考える．

- 文字列値トランスダクションを適用する式 $T(y_1, \cdots, y_{n_1}, t_1, \cdots, t_{n_2})$ において，各 t_i は整数変数．
- φ_{int} は $\varphi^1_{int} \wedge \varphi^2_{int}$ の形をしている．ここで φ^1_{int} は $j = f(x, i_1, \cdots, i_n)$ の形をした等式の連言で，$j, i_1, \cdots, i_n \in I$．また φ^2_{int} は整数値関数を含まない論理式である．

例 5.1. φ が次のように与えられたとする．

$$x_1 = x_0.substr(x_0.indexOf_{\mathtt{aab}}(0), |x_0|)$$
$$x_1 \notin \mathtt{aab}.^*$$

このとき新たな整数変数 i, l をとり，φ^1_{int} を

$i = x_0.indexOf_{\mathtt{aab}}(0) \wedge l = |x_0|$ とすることで φ_{sl} を $x_1 = x_0.substr(i,l)$ にすることができる．なお，φ_{reg} は $x_1 \notin \mathtt{aab}.^*$ である． □

$S_{<i} = \{x_j \mid j < i\}$ とすると $S = S_{<m}$ である．$S_{<i}$ の附値 $\eta\colon S_{<i} \to A^*$ に対し $\gamma(\eta) = \eta(x_0)\#\cdots\eta(x_{i-1})\# \in (A^*\#)^i$ を定義する．これは $S_{<i} \to A^*$ と $(A^*\#)^i$ の同型を導く．本節はこれを利用し，アルファベット $A \cup \{\#\}$ 上の文字列で $S_{<i}$ の附値を表す．

この節で述べる手続きは以下の仮定を満たす制約を解くことができる：

- 等式 $j = f(x_i, i_1, \cdots, i_n)$ を φ^1_{int} の s 番目の等式とする．これに対して関係 $R_s \subseteq A^* \times \mathbb{Z}^I$ を
  $(w,\eta) \in R_s \iff \eta(j) = f(w, \eta(i_1), \cdots, \eta(i_n))$
  で定める．任意の R_s は Parikh オートマトンで表現できる．
- φ_{sl} の等式 $x_i = T(x_{l_1}, \cdots, x_{l_{n_1}}, i_1, \cdots, i_{n_2})$ に対して関数 $g_i\colon \mathbb{Z}^I \to (A^*\#)^i \to 2^{A^*}$ を
  $g_i(\eta)(w_0\#\cdots w_{i-1}\#) =$
  $T(w_{l_1}, \cdots, w_{l_s}, \eta(i_1), \cdots, \eta(i_n))$
  と定める．任意の g_i に対して $A \cup \{\#\}$ をアルファベットとする Parikh SST $\mathcal{S}_i$ が構成できて，
  $$\mathcal{S}_i(\eta)(w) = \begin{cases} g_i(\eta)(w) & \text{if } w \in (A\#)^i \\ \varnothing & \text{otherwise} \end{cases}$$
  が成り立つ．

以下では制約 φ の充足可能性判定手続きを議論するが，これは次のようにまとめられる：

1. φ_{int} と φ_{reg} を充足する附値全体を捉えた Parikh オートマトン $\mathcal{A} \subseteq (A^*\#)^m \times \mathbb{Z}^I$ を構成する．
2. φ_{sl} を Parikh SST $\mathcal{T}_k, \cdots, \mathcal{T}_{m-1}$ ただし $\mathcal{T}_i\colon \mathbb{Z}^I \to (A^*\#)^i \to 2^{(A^*\#)^{i+1}}$ に変換する．
3. $\mathcal{B} := \mathcal{T}_k^{-1}(\cdots\mathcal{T}_{m-1}^{-1}(\mathcal{A})\cdots)\colon \mathbb{Z}^I \to 2^{(A^*\#)^k}$ を計算し，その空性を判定する．

Step 1. $\varphi_{reg} \wedge \varphi_{int}$ を捉えた Parikh オートマトンの構成

φ_{reg} すなわち $\bigwedge_{i=0}^{m-1} x_i \in L_i$ から，$x_0, \cdots, x_{m-1}$ の満たすべき関係を表す言語 $L_0\#\cdots L_{m-1}\# = \{w_0\#\cdots w_{m-1}\# \mid w_i \in L_i\}$ が定義できる．これは容易に有限オートマトンで表現できるため，$L_0\#\cdots L_{m-1}\# \times \mathbb{Z}^I$ を受理する Parikh オートマトン $\mathcal{A}_{reg}$ を構成する．

φ^1_{int} の s 番目の等式 $j = f(x_i, i_1, \cdots, i_n)$ をとる．仮定より $R_s \subseteq A^* \times \mathbb{Z}^I$ は適当な Parikh オートマトン $\mathcal{A}_s$ で表現できる．さらにこれを $(w_0\#\cdots w_{m-1}\#, \eta) \in \mathcal{A}'_s \iff (w_i, \eta) \in \mathcal{A}_s$ を満たす Parikh オートマトン $\mathcal{A}'_s$ へと変換する．$\mathcal{A}'_s$ 全体の共通部分をとり，その結果の論理式に φ^2_{int} を加えたものを $\mathcal{A}_{int}$ とする．さらに $\mathcal{A}$ を $\mathcal{A}_{reg}$ と $\mathcal{A}_{int}$ の共通部分とする．このとき

命題 5.2. 任意の $\eta_S\colon S \to A^*$, $\eta_I\colon I \to \mathbb{Z}$ について $\eta_S, \eta_I \models \varphi_{reg} \wedge \varphi_{int} \iff (\gamma(\eta_S), \eta_I) \in \mathcal{A}$.

例 5.1 (Continued)**.** 二つ目の等式 $l = |x_0|$ からは $R_2 = \mathrm{LENGTH}_l$ (例 3.5) が得られ，これは Parikh オートマトンで表せる． □

Step 2. φ_{sl} を捉えた Parikh SST の構成

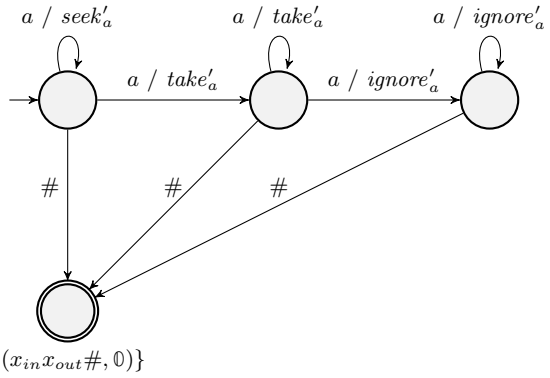
φ_{sl} の各等式 $x_i = T(x_{l_1}, \cdots, x_{l_{n_1}}, i_1, \cdots, i_{n_2})$ に対して定義される g_i を Parikh SST $\mathcal{S}_i$ で表現する．さらに $\mathcal{S}_i$ に入力をそのまま保存して出力に使う変数を追加することで，以下の関数 $h_i\colon \mathbb{Z}^I \to (A^*\#)^i \to 2^{(A^*\#)^{i+1}}$ を計算する Parikh SST $\mathcal{T}_i$ を構成する：$h_i(\eta)(w) = \{ww'\# \mid w' \in g_i(\eta)(w)\}$, $w \in (A^*\#)^i$. これら $(h_i)_i$ は次の性質を保証する．

命題 5.3. $(h_i)_i$ の合成を $h(\eta) = h_{m-1}(\eta)\circ\cdots\circ h_k(\eta)$ で定義するとき，任意の $\eta_S\colon S \to A^*$, $\eta_I\colon I \to \mathbb{Z}$ について
$\eta_I, \eta_S \models \varphi_{sl} \iff \gamma(\eta_S) \in h(\eta_I)(\gamma(\eta_S \restriction_{S_{<k}}))$.

例 5.1 (Continued)**.** $x_1 = x_0.substr(i,l)$ からは関数 $\mathrm{SUBSTR}_{i,l}\colon \mathbb{Z}^I \to A^* \to A^*$ が得られる (例 3.6)．これを表現する Parikh SST (図 1) は $h_1\colon \mathbb{Z}^I \to A^*\# \to A^*\#A^*\#$ と等価なものへと図 2 のように変換できる． □

Step 3. 逆像計算と充足可能性判定

4.3 節の方法を繰り返し使うことで逆像 $h^{-1}(\mathcal{A}) = \mathcal{T}_k^{-1}(\cdots(\mathcal{T}_{m-1}^{-1}(\mathcal{A}))\cdots)\colon \mathbb{Z}^I \to 2^{(A^*\#)^k}$ を計算し，その空性を判定する．空であるとき，かつそのときに限り制約は充足不可能であると判定する．以上の手続



$X = \{x_{in}, x_{out}\}$, $L = \{input, index, output\}$, $a \in A$

図 2: $h_1(\eta)(w_0\#) = w_0\#\mathrm{SUBSTR}_{i,l}(\eta)(w_0)\#$ を計算する Parikh SST. 図 1 からの変形であり，$seek'_a$ などはそれぞれプライムがついていないものの文字列変数更新を，$x_{in} := x_{in}a$ を含むように修正したものである．元の \$ による遷移は # による遷移に変換され，$x_{in} := x_{in}\#$ を含む．

きの健全性と完全性は以下の性質から保証される．

命題 5.4. $\eta_I : I \to \mathbb{Z}$ とする．$h^{-1}(\mathcal{A})(\eta_I) \subseteq (A^*\#)^k$ が非空であるとき，かつそのときに限りある $\eta_S : S \to A^*$ が存在して $\eta_S, \eta_I \models \varphi_{sl} \wedge \varphi_{reg} \wedge \varphi_{int}$ が成り立つ．

Proof. $h^{-1}(\mathcal{A})(\eta_I) \neq \varnothing$ を仮定すると，ある $w \in (A^*\#)^k$ が存在して $w \in h^{-1}(\mathcal{A})(\eta_I)$ が成り立つ．逆像の定義よりある $w' \in (A^*\#)^m$ が存在して $w' \in h(\eta_I)(w) \cap \mathcal{A}(\eta_I)$ である．命題 5.2, 5.3 より $\eta_S = \gamma^{-1}(w')$ が $\eta_S, \eta_I \models \varphi_{sl} \wedge \varphi_{reg} \wedge \varphi_{int}$ を満たす．逆向きも同様である． □

なお，充足可能な場合に具体的な解（モデル）を生成することもできる．$h^{-1}(\mathcal{A})$ が計算するベクトルの型を $\mathbb{Z}^L$ とするとき，空性判定で構成される線形算術論理式 ψ（命題 4.12）の充足可能性判定によりベクトル $v \in \mathbb{Z}^L$ と $\eta_I \in \mathbb{Z}^I$ が得られる．また $h^{-1}(\mathcal{A})$ が v を出力するような文字列 $w \in (A^*\#)^k$ は幅優先探索で見つけられる．$h(\eta_I)(w)$ は有限集合であるから，その要素 $w_0\# \cdots \#w_{m-1}\#$ の中から $\eta_S \equiv x_i \mapsto w_i$ $(0 \leq i < m)$ と η_I がモデルになるものを選べる．

6 実装と評価

本研究は 5 節の決定手続きを文字列理論の SMT ソルバとして実装した．まず実装についていくつか述べる．5 節の手続きは線形算術論理式の充足可能性判定を含む．本研究はこれを解くために Z3 SMT ソルバ [10] を使用した．またソルバの入出力フォーマットや挙動は SMT ソルバの標準 SMT-LIB [5] を参考にした．

Parikh オートマトンの逆像計算の実装については以下のようにした．構成の第一段階 (4. 2. 1 節) では M 出力 SST の状態集合を定義したが，その要素を全て列挙すると非常に大きくなってしまう．そこで始状態から到達可能かつ終状態へと到達可能な状態のみを構成するよう試みる．このとき状態集合 Q の列挙には次の 2 通りの戦略が考えられる：1. $Q := Q_0$ とし，$q \in Q$ から到達できる状態を付け加えることで探索する，または 2. $Q := \{q_f\}$ とし，$q \in Q$ へと到達できる状態を付け加えることで探索する．戦略 2 は 4. 2. 1 節で説明した $f : X \rightharpoonup Q_2 \times Q_2$ が「SST の出力文字列によってトランスデューサが始状態から終状態へと遷移する」ことを推測している状態 $q_f = (q_f^1, f_\$)$ からの逆算で構成する．一方で Q_0 は推測に課せられる条件により $X \rightharpoonup Q_2$ と同型であるが，これはトランスデューサが各変数をどの状態で読むかの推測であり，戦略 1 はその全ての可能性を考慮する．二つの戦略の内，戦略 1 は簡単な例でも現実的な計算資源で停止しない場合があった．このことから，現実の例において Q_0 は不要な推測を多く含むことが示唆された．そのため本研究では戦略 2 を採用した．

以下ではソルバの評価について述べる．本研究は実装したソルバでいくつかの制約を解く実験をおこない，性能及び表現力を Zhu らのソルバや OSTRICH と比較した．本節で述べる比較は執筆時点での各ソルバの GitHub 上における最新バージョンに基づいており，それぞれコミットハッシュは本研究のソルバが fa58f80, Zhu らのソルバが 2b2af7c, OSTRICH が 824cec8 である．実験はすべて Intel Core i7-8550U @ 1.80 GHz と 16 GiB RAM を搭載する計算機上で実行した．結果の概要を表 2 に示す．なお OS-

TRICH+ は公開されていないため，これで制約例を解く実験はできなかった．

以下で実験に用いた制約を詳しく説明する．全ての制約について，アルファベット A は制約に現れる文字集合に $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ を加えたものとした．

まず全てのソルバで解ける制約で実行速度について簡単に比較する．

例 6.1. 次の制約は Zhu らのソルバや OSTRICH でも解くことができる．

$$x_2 = x_0.replaceAll(/\texttt{<script>}/, \varepsilon)$$
$$x_3 = x_1.replaceAll(/\texttt{<script>}/, \varepsilon)$$
$$x_4 = x_2 \cdot x_3$$
$$x_4 \in \texttt{<script>}$$

この制約は充足可能であり，本研究のソルバは次のモデルを出力した．

$$x_0 = x_2 = \texttt{<scr}$$
$$x_1 = x_3 = \texttt{ipt>}$$
$$x_4 = \texttt{<script>}$$

Zhu らのソルバはこの制約を解くのに 4 分以上かかった一方で，本研究のソルバは約 18 秒で解くことができた．また OSTRICH はこの制約を 1 秒未満で解くことができた．

本研究のソルバが Zhu らのものより速い要因は，基礎制約 $x_i = T(x_j)$ から構成する SST の変数の個数の違いにある．$x_i = T(x_j)$ について T が n 個の変数を使用するとき，Zhu らのソルバは $n+i+1$ 個の変数を使用していた．ここで $i+1$ 個の変数とは $\{x_0, x_1, \cdots, x_i\}$ であり，最終状態では $x_0\#x_1\#\cdots x_i\#$ を出力していた．しかし，x_i すなわち $T(x_j)$ の値を構成する部分以外は単に入力をそのまま出力しているだけであるから，5 節図 2 のように $\{x_0, x_1, \cdots, x_i\}$ の代わりに適当な 1 つの変数 x_{in} を使うだけで足りる．この最適化を Zhu らのプログラムに適用したとき，実行時間は約 40 秒へと短縮された．

OSTRICH と本研究の比較については第 7 節における議論を参照されたい． □

続いて 3 つの制約例によって本研究のソルバが扱える制約の表現力を示す．これらは，Zhu らのソルバや OSTRICH で解けないが本研究のソルバが扱える制約の例になっている．OSTRICH+ は例 6.2 と 6.3 を扱えると思われるが，例 6.4 には対応していないと考えられる．

Zhu らによるソルバや OSTRICH では，部分文字列をとる演算 *substr* は限定的な形でしかサポートされていなかった．本研究のソルバでは任意の線形算術式を整数引数に書くことができる．

例 6.2. 次の制約は充足可能である．

$$i = x_0.indexOf_{\texttt{aab}}(0)$$
$$x_1 = x_0.substr(i, |x_0| - i)$$
$$x_1 \notin \texttt{aab}.^*$$

ソルバは次のモデルを出力した．

$$x_0 = \texttt{bbbbb}$$
$$x_1 = \varepsilon$$
$$i = -1$$

また，制約 $x_0 \in .^*\texttt{aab}.^*$ を追加したとき正しく充足不可能と判定した．

Zhu らのソルバは *substr* の整数引数が定数の場合しかサポートしない．また OSTRICH は x_0 が複数回左辺に現れる制約に変形するため解くことができない．一方で OSTRICH+ は substr や indexOf に対応しているため解けると考えられる． □

等号否定を使うと文字列操作プログラムの等価性を表現できる．

例 6.3. 次の操作は等価である．

- $x.substr(i, l)$
- $x.substr(i, |x| - i).substr(0, l)$

この等価性は次の制約が充足不可能であることと同値である．

$$x_1 = x_0.substr(i, l)$$
$$x_2 = x_0.substr(i, |x_0| - i)$$
$$x_3 = x_2.substr(0, l)$$
$$x_1 \neq x_3$$

本研究のソルバはこの制約を充足不可能と判定した．

Zhu らのソルバや OSTRICH は等号否定に対応していないため，この制約を解くことができない．一方で OSTRICH+ はサポートしているため解けると考えられる． □

例 6.4. 等号否定の別の応用として，プログラムの可逆性やべき等性の検証がある．この例は 2 つの文字

表 2: 実験結果の概要．各段階の実行時間 (ミリ秒) と構成される Parikh オートマトンの大きさを示している．「逆像」は制約から機械を構成し逆像を計算するのに要した時間,「SAT」は Parikh オートマトンから論理式への変換とその充足可能性判定に要した時間,「モデル」は充足可能だった場合にモデル探索に要した時間．また，$|Q|$ と $|\Delta|$ は逆像計算で得られた Parikh オートマトンの状態と遷移集合の要素の個数．条件の追加で充足不可能になることを述べた例については，追加前 (sat) と追加後 (unsat) を区別している．

制約		逆像	SAT	モデル	$\|Q\|$	$\|\Delta\|$
例 6.1		17129	384	27	731	870
例 6.2	(sat)	293	127	31	30	144
例 6.2	(unsat)	252	79	-	19	53
例 6.3		470	752	-	87	1092
例 6.4		561	1949	870	157	846

列置換が互いに逆操作となっているか検証する：文字集合を $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ とするとき，

$$x_1 = x_0.\mathit{replaceAll}(/(\mathtt{a}^+)_1(\mathtt{b}^+)_2/, \backslash 2 \backslash 1)$$
$$x_2 = x_1.\mathit{replaceAll}(/(\mathtt{b}^+)_1(\mathtt{a}^+)_2/, \backslash 2 \backslash 1)$$
$$x_0 \neq x_2$$

この制約は充足可能であり，よって 2 つの操作は逆操作ではない．本研究のソルバはこれを約 3 秒で解いた．

この例から構成される線形算術論理式に対し，Z3 は Parikh オートマトンが出力するベクトルとして $(32, 32, 5, 5, 98, 97)$ を生成した．ここで 32 は x_0 と x_2 の長さを，5 は異なる文字になる位置を，そして 98, 97 は b, a の文字コードを表す．ソルバは次のようなモデルを生成した．

$$x_0 = \mathtt{bbbcbacb}^{25}$$
$$x_1 = \mathtt{bbbcbacb}^{25}$$
$$x_2 = \mathtt{bbbcabcb}^{25}$$

一方で以下の変更を加えた制約は 10 分以内に判定できなかった．(1) $A = \{0, 1, 2\}$ とし，‘a’ を ‘0’, ‘b’ を ‘2’ に変えた場合．Z3 が出力するベクトルの，異なる文字になる位置を示す成分が大きくなり，モデル生成のための幅優先探索が終わらなかった．(2) 文字集合 A を ASCII printable の 94 文字に増やした場合．このとき線形算術論理式の判定が終わらなかった．

この制約には等号否定が現れるため，やはり Zhu らのソルバや OSTRICH では解くことができない．また OSTRICH+ も解くことができないと考えられる．OSTRICH+ は，等号否定や，マッチした文字列を定数文字列に置き換える replaceAll 関数は扱えるとされているものの，キャプチャグループを使う replaceAll 関数が扱えるとは言及されていない．実際，このような replaceAll 関数と等号否定を同時にサポートするために，本研究は 4.2 節で示した構成を必要としたのである． □

7 まとめと関連研究

本稿は非決定性 SST による Parikh オートマトンの逆像を Parikh オートマトンとして構成する方法を示した．また同じ構成法が適用できる Parikh SST と呼ぶ拡張を導入した．さらにこれらの機械を使うことで，文字列から整数への関数 indexOf, 引数に整数を含む文字列のトランスダクション substring, 正規表現にマッチした文字列を置換文字列の構成で使える文字列置換 replaceAll を含む直線文字列制約が解けることを示した．

本研究に類似する成果として Chen らによる研究 [8] がある．彼らのソルバ OSTRICH+ もまた indexOf や substring を扱うことができる．OSTRICH+ も言語の逆像をとる操作に基づいているが，解空間を cost-enriched recognisable relation (CERR) と線形算術論理式の組によって捉える．ここで CERR とは関係 $\mathcal{R} \subseteq (A^* \times \mathbb{Z}^{k_1}) \times \cdots \times (A^* \times \mathbb{Z}^{k_l})$ であり，ベクトル出力トランスデューサ（彼らの言葉では cost-enriched finite automaton: CEFA）の集合

$\{\mathcal{T}_{i,j}\}$ によって $\mathcal{R} = \bigcup_{i=1}^{n} \mathcal{T}_{i,1} \times \cdots \times \mathcal{T}_{i,l}$ のように表現する．彼らは CEFA が表す言語 $L \subseteq A^* \times \mathbb{Z}^{k_0}$ と関数 $f\colon (A^* \times \mathbb{Z}^{k_1}) \times \cdots \times (A^* \times \mathbb{Z}^{k_l}) \to A^*$ に対して，f による L の逆像を CERR $\mathcal{R} \subseteq (A^* \times \mathbb{Z}^{k_1+k_0}) \times \cdots \times (A^* \times \mathbb{Z}^{k_l+k_0})$ と線形算術項の k_0 次元ベクトル $\mathbf{t} = (t_1, \cdots, t_{k_0})$ の組 $(\mathcal{R}, \mathbf{t})$ で，特定の条件を満たすものと定義した．

OSTRICH+ の決定手続きは以下の 2 点によりオートマトン (CEFA) のサイズを小さく抑えている：

- 各文字列変数ごとに独立したオートマトンで解の集合を表す．
- 逆像を計算するたびに得られる CERR は選言標準形 (DNF) とみなすことができるが，その中から一つの節 (すなわち CEFA 集合 $\{\mathcal{T}_{i,1}, \cdots, \mathcal{T}_{i,l}\}$) を非決定的に選択して手続きを継続する．

そのため，OSTRICH+ は効率面で優れていると考えられる．公開されていないため直接の比較はできなかったが，解空間を Parikh オートマトンの集合と追加的な整数制約の組で表した素朴な実装を試したところ，特に replaceAll を含むような制約では本稿に基づく実装と比べて高速であった．

一方で CEFA の逆像の定義は複雑であった．本研究の貢献の一つは，ベクトル出力トランスダクションの逆像と呼ばれる集合を，Parikh オートマトンを用いて自然な形に定式化した（4.3 節）ことにある．これによって計算モデルの詳細が文字列制約の決定手続きにおいて適切に隠蔽され，決定手続きが満たしている命題 5.2 などの性質を記述し健全性と完全性を確認することが容易となった．

文字列制約における replaceAll 関数には，本研究で触れたものとこれまでに研究されてきたものを合わせて次の 3 つの変種がある：r を正規表現，w を文字列定数，α をキャプチャグループ変数を含む文字列，β を制約の文字列変数を含む文字列とするとき，(1)$replaceAll_{r,w}$，(2)$replaceAll_{r,\alpha}$，(3)$replaceAll_{r,\beta}$ である．例えば $x.replaceAll_{r,\beta}$ において $r = \mathtt{a}$，$\beta = y$ とした式は，x の値に現れる文字 $\mathtt{a}$ を全て文字列変数 y の値で置き換えたものを表す．OSTRICH+ では (1) のみが扱えた．また，(3) の関数と文字列長制約を同時に含むような直線文字列制約は，一般に決定不能であることが知られていた [7]．本研究が導入した Parikh SST は，OSTRICH+ を開発した文献 [8] で取り上げられている全ての文字列値関数に加え，(2) の replaceAll 関数を模倣できる．さらに，OSTRICH+ では reverse や substring のように文字の入れ替えや整数引数を伴うトランスダクションについて，逆像の計算方法を個別に実装する必要があったが，本稿の手法では Parikh SST としての表現から逆像の計算法が自動的に導出される．このように，Parikh オートマトンや CEFA の逆像をとれる計算モデルで極大な表現力を持つものを見出したことが本研究の別の貢献である．

最後に表現力について付け加える．Chen ら [8] が提案した決定手続きは CEFA の逆像が取れる全ての文字列操作に対応できるフレームワークとなっていた．4.3 節の構成法は CEFA の逆像の構成法に変換できるため，この意味で本研究の表現力は彼らの方法の表現力に内包される．ただし OSTRICH+が明示的にサポートしている全ての整数値関数は Parikh オートマトンで，文字列値関数は Parikh SST でそれぞれ模倣できるため，実装が扱える制約のクラスという意味では本研究の方が真に大きいと考えられる．

謝辞 本研究の一部は JSPS 科研費 19K11899, 20H04162 の助成を受けたものである．

参 考 文 献

[1] Alur, R. and Černỳ, P.: Expressiveness of streaming string transducers, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, Vol. 8, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 1–12.

[2] Alur, R. and Černỳ, P.: Streaming transducers for algorithmic verification of single-pass list-processing programs, *Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 2011, pp. 599–610.

[3] Alur, R. and Deshmukh, J. V.: Nondeterministic streaming string transducers, *International Colloquium on Automata, Languages, and Programming*, Springer, 2011, pp. 1–20.

[4] Alur, R., Filiot, E., and Trivedi, A.: Regular Transformations of Infinite Strings, *27th Annual IEEE Symposium on Logic in Computer Science (LICS)*, June 2012, pp. 65–74.

[5] Barrett, C., Stump, A., and Tinelli, C.: The

SMT-LIB Standard: Version 2.0, *Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, UK)*, 2010.
[6] Cadilhac, M., Finkel, A., and McKenzie, P.: On the Expressiveness of Parikh Automata and Related Models, *Third Workshop on Non-Classical Models for Automata and Applications*, Austrian Computer Society, 2011, pp. 103–119.
[7] Chen, T., Chen, Y., Hague, M., Lin, A. W., and Wu, Z.: What is Decidable about String Constraints with the ReplaceAll Function, *Proc. ACM Program. Lang.*, Vol. 2, No. POPL(2017).
[8] Chen, T., Hague, M., He, J., Hu, D., Lin, A. W., Rümmer, P., and Wu, Z.: A Decision Procedure for Path Feasibility of String Manipulating Programs with Integer Data Type, *Automated Technology for Verification and Analysis*, Springer International Publishing, 2020, pp. 325–342.
[9] Chen, T., Hague, M., Lin, A. W., Rümmer, P., and Wu, Z.: Decision Procedures for Path Feasibility of String-Manipulating Programs with Complex Operations, *Proc. ACM Program. Lang.*, Vol. 3, No. POPL(2019).
[10] De Moura, L. and Bjørner, N.: Z3: An efficient SMT solver, *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2008, pp. 337–340.
[11] Filiot, E. and Reynier, P.-A.: Copyful Streaming String Transducers, *International Workshop on Reachability Problems*, Springer, 2017, pp. 75–86.
[12] 加賀江優幸, 南出靖彦: Streaming String Transducer の等価性判定と正規表現による文字列置換への応用, 情報処理学会論文誌プログラミング *(PRO)*, Vol. 8, No. 3(2015), pp. 1–10.
[13] Klaedtke, F. and Rueß, H.: Parikh Automata and Monadic Second-Order Logics with Linear Cardinality Constraints, Technical report, Universität Freiburg, 2002.
[14] Lin, A. W. and Barceló, P.: String Solving with Word Equations and Transducers: Towards a Logic for Analysing Mutation XSS, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '16, Association for Computing Machinery, 2016, pp. 123–136.
[15] Reynolds, A., Woo, M., Barrett, C., Brumley, D., Liang, T., and Tinelli, C.: Scaling Up DPLL(T) String Solvers Using Context-Dependent Simplification, *Computer Aided Verification*, Springer International Publishing, 2017, pp. 453–474.
[16] Tinelli, C., Barrett, C., and Fontaine, P.: SMT-LIB The Satisfiability Modulo Theories Library – Unicode Strings, `http://smtlib.cs.uiowa.edu/theories-UnicodeStrings.shtml`. 2021 年 8 月 2 日最終閲覧.
[17] Verma, K. N., Seidl, H., and Schwentick, T.: On the Complexity of Equational Horn Clauses, *Automated Deduction – CADE-20*, Springer, 2005, pp. 337–352.
[18] Zheng, Y., Zhang, X., and Ganesh, V.: Z3-Str: A Z3-Based String Solver for Web Application Analysis, *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, Association for Computing Machinery, 2013, pp. 114–124.
[19] Zhu, Q., Akama, H., and Minamide, Y.: Solving String Constraints with Streaming String Transducers, *Journal of Information Processing*, Vol. 27(2019), pp. 810–821.

A indexOf 関数と replaceAll 関数

ここでは 2 節表 1 の関数の内，機械への変換法を述べなかったものを説明する．

A.1 indexOf 関数

indexOf 関数は，2 つの文字列 w, w_2 を受け取って，w における w_2 の最初の出現位置を返す関数である．Parikh オートマトンは indexOf 関数を表現することができる．より正確に，まず $w_2 \in A^*$ につき関数 $\mathit{indexOf}_{w_2} : A^* \times \mathbb{Z} \to \mathbb{Z}$ を

$$w.\mathit{indexOf}_{w_2}(i) = \begin{cases} \mathit{min}\{j \mid w = w_1 w_2 w_3,\ i \le j = |w_1|\} & \text{if } i \ge 0 \\ -1 & \text{otherwise} \end{cases}$$

と定める．ただし $\mathit{min}\varnothing = -1$ とする．文字列 w を固定するとき，制約 $j = x.\mathit{indexOf}_w(i)$ が定める関係として $\mathrm{INDEXOF}_{w,i,j} \subseteq A^* \times \mathbb{Z}^I$ を考える．すなわち $\mathrm{INDEXOF}_{w,i,j}$ は，$\eta_S \colon S \to A^*$ と $\eta_I \colon I \to \mathbb{Z}$ に対して $(\eta_S(x), \eta_I) \in \mathrm{INDEXOF}_{w,i,j} \iff \eta_I(j) = \eta_S(x).\mathit{indexOf}_w(\eta_I(i))$ を満たす．例えば制約 $j = x.\mathit{indexOf}_{\mathtt{aab}}(0)$ に対応する $\mathrm{INDEXOF}_{\mathtt{aab},0,j}$ は $(u, \eta) \in \mathrm{INDEXOF}_{\mathtt{aab},0,j}$ が以下のいずれかが成り立つことと同値である：

- $\eta(j) = -1$ かつ w は $\mathtt{aab}$ を含まない．
- $u = w\mathtt{aab}w'$ かつ $|w| = \eta(j)$ かつ $w\mathtt{aa}$ は $\mathtt{aab}$ を含まない．

$\mathrm{INDEXOF}_{\mathtt{aab},0,j}$ は図 3 の Parikh オートマトンが表現する．また，この構成法は一般の $w \neq \varepsilon$ と自然数定数 c に対して $y = x.\mathit{indexOf}_w(c)$ に対応する Parikh オートマトンの構成法へ一般化できる．さらに，c が整数変数であるときは SUBSTR のように非決定的に読み飛ばす部分を加えることで表現できる．

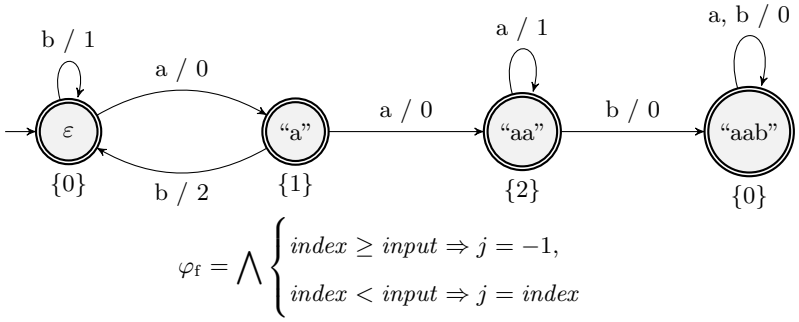


図 3: $\mathrm{INDEXOF}_{\mathtt{aab},0,j}$ を表現する Parikh オートマトン．$A = \{\mathtt{a}, \mathtt{b}\}$, $L = \{input, index\}$ とする．$input$ は入力文字列長を，$index$ は aab が現れるまでの接頭語の長さを記録する．遷移は KMP 法を模倣しており，“aab” 以外の状態に割り当てられた文字列はその時点までに読んだ入力の接尾語で検索対象文字列の接頭語に一致するもののうち最長のものを意味する．以上の意味を達成するために，すべての遷移で $input$ には 1 を加える．また，ラベルの数字は $index$ に加える数を表している．各状態に割り当てられた数はその状態で入力がなくなったとき $index$ に加えられる数である．また，どの状態で入力がなくなっても $input$ には 0 が加えられる．

A.2 正規表現による replaceAll

表 1 のトランスダクションの内，$w.replaceAll_{r,\alpha}$ については SMT-LIB の仕様と異なる点がある．SMT-LIB の関数では正規表現 r は優先度をもたない．置換対象文字列を先頭から読み，正規表現 r に最初にマッチした非空な文字列で最短のものを α に置き換えることを繰り返すよう定めている．この仕様では例えば $\mathtt{aa}.replaceAll(/(\mathtt{aa})|\mathtt{a}/, \mathtt{b}) = \mathtt{bb}$ となる．

しかし現実のプログラミング言語のライブラリに含まれる正規表現マッチングの多くは選択や反復の場合に優先度を考慮する．本研究ではそのような正規表現置換を採用した．正規表現 r の構文は $r ::= w \mid r_1 r_2 \mid r_1|r_2 \mid r^* \mid r^{*?} \mid (r)_x$ であり，$r_1|r_2$ は r_1 を優先する選択，$r^{*?}$ は非貪欲な反復，$(r)_x$ は名前 x を持つキャプチャグループを表す．マッチングの詳細な意味論や SST への翻訳は文献 [12] に詳しいため繰り返さないが，例を示すと

$$\mathtt{aa}.replaceAll(/(\mathtt{aa})|\mathtt{a}/, \mathtt{b}) = \mathtt{b}$$
$$\mathtt{aa}.replaceAll(/\mathtt{a}|(\mathtt{aa})/, \mathtt{b}) = \mathtt{bb}$$
$$\mathtt{abb}.replaceAll(/.^{*?}\mathtt{b}/, \mathtt{c}) = \mathtt{cc}$$
$$\mathtt{abb}.replaceAll(/(.^{*?})_1\mathtt{b}/, \mathtt{c}\backslash 1) = \mathtt{cac}$$

となる．ここで ‘\1’ はキャプチャグループ 1 にマッチした文字列を表す．

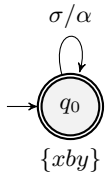
なお本研究では意味論の複雑さを避けるため r が次の条件を満たすもののみを対象とした．

- r に空文字列はマッチしない．
- r の部分式で，$(r')^*$ または $(r')^{*?}$ の形のものについて，r' に空文字列はマッチしない．

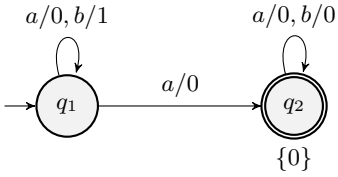
B 合成例

B.1 SST とモノイド出力トランスデューサの合成

$\alpha(x) = x\sigma$, $\alpha(y) = \sigma y$ とするとき，次の SST は関数 $w \mapsto wb(w.reverse)$ を表す．

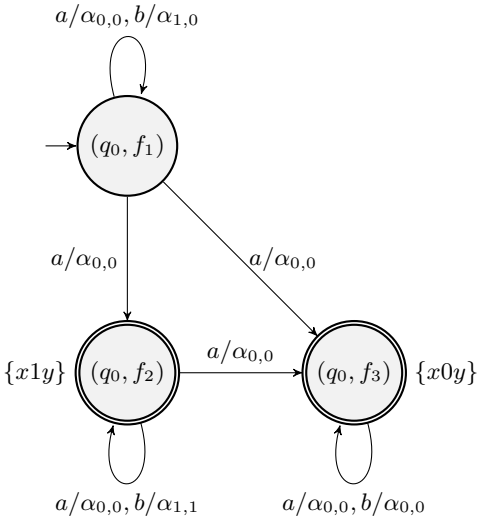


次の $(\mathbb{N}, +)$ 出力トランスデューサは入力の a で終わる接頭辞に含まれる b の個数を出力する．



これらの合成は次の $(\mathbb{N}, +)$ 出力 SST と等価である．ただし $\alpha_{i,j}(x) = xi$, $\alpha_{i,j}(y) = jy$ とする．変数には数の列が格納され，動作終了時に列を加法で畳み

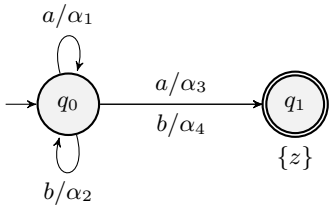
込んで出力を計算する．



推測	x	y
f_1	(q_1, q_1)	(q_2, q_2)
f_2	(q_1, q_1)	(q_1, q_2)
f_3	(q_1, q_2)	(q_2, q_2)

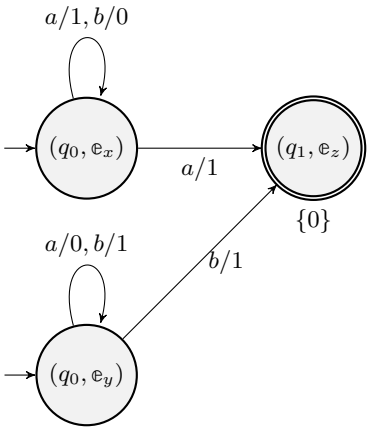
B. 2　モノイド出力 SST の変換

次の $(\mathbb{N}, +)$ を出力する SST は最後に読む文字の個数をカウントする．



更新	x	y	z
α_1	$x1$	y	z
α_2	x	$y1$	z
α_3	ε	ε	$x1$
α_4	ε	ε	$y1$

$\mathbb{e}_x$ を x のみ 1 な単位ベクトルとし，$\mathbb{e}_y$, $\mathbb{e}_z$ も同様とすると，次のトランスデューサに変換できる．



C　命題 4.3 の証明

ここでは 4. 2 節で示した SST とトランスデューサの合成の第一段階が正しい M 出力 SST を構成していることを証明する．

補題 4.5. $u \in A^*$ について $(q_1, f) \xrightarrow[\Delta]{u/\alpha'} (q_1', f')$ であるとき，$\alpha \in \mathrm{Update}(X, B)$ が存在して，$q_1 \xrightarrow[\Delta_1]{u/\alpha} q_1'$ 及び次の条件を満たす：任意の $q_2, q_2' \in Q_2$, $w \in (X \cup B)^*$, $w' \in (X \cup M)^*$ に対して

$$q_2 \xrightarrow[\Delta_2^{(f')}]{w/w'} q_2' \text{ ならば } q_2 \xrightarrow[\Delta_2^{(f)}]{\alpha(w)/\alpha'(w')} q_2'$$

が成り立つ．

Proof. $u \in A$ の場合は Δ の定義の 2 つ目の条件を $w \in (X \cup B)^*$ へ拡張した主張であり明らか．u の長さに関する帰納法も容易である．　□

補題 4.6. $w \in (X \cup B)^*$ は重複する変数の出現を持たないとする．$u \in A^*$ について $q_1 \xrightarrow[\Delta_1]{u/\alpha} q_1'$ かつ $q_2 \xrightarrow[\Delta_2^{(f)}]{\alpha(w)/w''} q_2'$ かつ $dom(f) = \mathrm{Var}(\alpha(w))$ であるとき，ある $\alpha' \in \mathrm{Update}(X, M)$, $f' \colon X \rightharpoonup (Q_2 \times Q_2)$, $w' \in (X \cup M)^*$ が存在して，$(q_1, f) \xrightarrow[\Delta]{u/\alpha'} (q_1', f')$ かつ $q_2 \xrightarrow[\Delta_2^{(f')}]{w/w'} q_2'$ かつ $\alpha'(w') = w''$ かつ $dom(f') = \mathrm{Var}(w)$ が成り立つ．

Proof. $u \in A$ の場合を示す．$w = w_1 x_1 \cdots w_n x_n w_{n+1}$, $w_i \in B^*$ とする．$\alpha(w_i) = w_i$ であるから，仮定より $q_2 \xrightarrow[\Delta_2^{(f)}]{w_1/w_1'} q_2^1 \xrightarrow[\Delta_2^{(f)}]{\alpha(x_1)/w_1''} r_2^1 \to \cdots \to r_2^n \xrightarrow[\Delta_2^{(f)}]{w_{n+1}/w_{n+1}'} q_2'$ かつ $w'' = w_1' w_1'' \cdots w_{n+1}'$ となる $(q_2^i, r_2^i) \in Q_2 \times Q_2$,

$w'_i \in M^*, w''_i \in (M \cup X)^*$ が存在する．$\alpha'(x_i) = w''_i$, $\alpha'(y) = \varepsilon$ $(y \neq x_i)$, $w' = w'_1 x_1 \cdots w'_n x_n w'_{n+1}$, $f'(x_i) = (q_2^i, r_2^i)$ とすると条件を満たす．実際，$q_2 \xrightarrow[\Delta_2^{(f')}]{w_1/w'_1} q_2^1 \xrightarrow[\Delta_2^{(f')}]{x_1/x_1} r_2^1 \to \cdots \to r_2^n \xrightarrow[\Delta_2^{(f')}]{w_{n+1}/w'_{n+1}} q'_2$, $\alpha'(w') = w'_1 \alpha'(x_1) \cdots w'_{n+1} = w''$, $dom(f') = \{x_1, \cdots, x_n\} = \mathrm{Var}(w)$ が成り立つ．Δ の遷移の条件も $dom(f) = \mathrm{Var}(\alpha(w)) = \bigcup_{x \in \mathrm{Var}(w)} \mathrm{Var}(\alpha(x)) = \bigcup_{x \in dom(f')} \mathrm{Var}(\alpha(x))$, $q_2^i \xrightarrow[\Delta_2^{(f)}]{\alpha(x_i)/\alpha'(x_i)} r_2^i$ $(x_i \in dom(f'))$, $\alpha'(x) = \varepsilon$ $(x \notin dom(f'))$ のように満たされる．□

命題の証明では次の性質を用いる．

補題 C.1. 重複する変数の出現を持たない任意の $w \in (X \cup B)^*$ と $m \in M$ について，$q_2 \xrightarrow[\Delta_2]{\varepsilon_X(w)/m} q'_2$ であるとき，またそのときに限り，ある $f_0 \colon X \rightharpoonup Q_2 \times Q_2$ と $w' \in (X \cup M)^*$ が存在して (i) $dom(f_0) = \mathrm{Var}(w)$ (ii) $f_0(x) = (q, r)$ ならば $q = r$, (iii) $q_2 \xrightarrow[\Delta_2^{(f_0)}]{w/w'} q'_2$, (iv) $m = \tau(w')$ が全て成り立つ．

命題 4.3. 任意の $w \in A^*$ に対して $\mathcal{M}(w) = \mathcal{T}(\mathcal{S}(w))$.

Proof. $m \in \mathcal{M}(w)$ であるとき，$\mathcal{M}$ の実行を

$$(q_0^1, f_0) \xrightarrow[\Delta]{w/\alpha'} (q_1, f_1), \tag{1}$$

$$(q_1, f_1) \xrightarrow[\Delta]{\$/\alpha''} (q_f^1, f_\$) \tag{2}$$

かつ $\tau(\alpha'(\alpha''(x_{out}))) = m$ であるようにとれる．このとき式 (2) について Δ の $\$$ による遷移の定義と $f_\$(x_{out}) = (q_0^2, q_f^2)$ であることから，ある α_2, w', m' が存在して $q_1 \xrightarrow[\Delta_1]{\$/\alpha_2} q_f^1$ かつ $\alpha''(x_{out}) = w'm'$ かつ

$$q_0^2 \xrightarrow[\Delta_2^{(f_1)}]{\alpha_2(x_{out})/w'} q_2, \tag{3}$$

$$q_2 \xrightarrow[\Delta_2^{(f_1)}]{\$/m'} q_f^2 \tag{4}$$

となる．式 (1), (3) に補題 4.5 を使って $q_0^1 \xrightarrow[\Delta_1]{w/\alpha_1} q_1$ かつ $q_0^2 \xrightarrow[\Delta_2^{(f_0)}]{\alpha_1(\alpha_2(x_{out}))/\alpha'(w')} q_2$ なる α_1 の存在が言える．また式 (4) より $q_2 \xrightarrow[\Delta_2^{(f_0)}]{\$/m'} q_f^2$ である．よって $q_0^1 \xrightarrow[\Delta_1]{w\$/\alpha_1 \cdot \alpha_2} q_f^1$ かつ $q_0^2 \xrightarrow[\Delta_2^{(f_0)}]{\alpha_1(\alpha_2(x_{out}))\$/\alpha'(w'm')} q_f^2$ である．補題 C.1 と等式 $\tau(\alpha'(w'm')) = \tau(\alpha'(\alpha''(x_{out}))) = m$ により $q_0^2 \xrightarrow[\Delta_2^{(f_0)}]{(\alpha_1 \cdot \alpha_2)(x_{out})\$/m} q_f^2$ であるから，$m \in \mathcal{T}(\mathcal{S}(w))$ を言える．

逆に $m \in \mathcal{T}(\mathcal{S}(w))$ であるとき，$\mathcal{S}$ と $\mathcal{T}$ に次のような実行がある：$q_0^1 \xrightarrow[\Delta_1]{w/\alpha_1} q_1 \xrightarrow[\Delta_1]{\$/\alpha_2} q_f^1$, $q_0^2 \xrightarrow[\Delta_2^{(f_0)}]{\alpha_1(\alpha_2(x_{out}))/w''} q_2 \xrightarrow[\Delta_2^{(f_0)}]{\$/m'} q_f^2$ かつ $m = \tau(w''m')$．ここで f_0 は $(q_0^1, f_0) \in Q_0$ かつ $dom(f_0) = \mathrm{Var}(\alpha_1(\alpha_2(x_{out})))$ を満たすようにとれる（補題 C.1）．補題 4.6 より $(q_0^1, f_0) \xrightarrow[\Delta]{w/\alpha'_1} (q_1, f_1)$ かつ $q_0^2 \xrightarrow[\Delta_2^{(f_1)}]{\alpha_2(x_{out})/w'} q_2$ かつ $\alpha'_1(w') = w''$ かつ $dom(f_1) = \mathrm{Var}(\alpha_2(x_{out}))$ を満たす α'_1, f_1, w' がある．よって $q_0^2 \xrightarrow[\Delta_2^{(f_1)}]{\alpha_2(x_{out})\$/w'm'} q_f^2$ となり，Δ の定義より $\mathcal{M}$ 上で $(q_1, f_1) \xrightarrow[\Delta]{\$/\alpha'_2} (q_f^1, f_\$)$ かつ $\alpha'_2(x_{out}) = w'm'$ となる α'_2 がある．以上を使って $\mathcal{M}$ が m を出力する実行を構成できる．□

D Parikh SST による Parikh オートマトンの逆像の構成法

ここでは 4.3 節で結果のみ述べた，Parikh SST による Parikh オートマトンの逆像の構成法を示す．つまり，与えられた Parikh SST $\mathcal{P} \colon \mathbb{Z}^I \to A^* \to 2^{B^*}$ と Parikh オートマトン $\mathcal{A} \colon \mathbb{Z}^I \to 2^{B^*}$ に対し，次を満たす Parikh オートマトン $\mathcal{B} \colon \mathbb{Z}^I \to 2^{A^*}$ を構成する：任意の $\eta \in \mathbb{Z}^I$ と $w \in A^*$ について "$w \in \mathcal{B}(\eta) \iff \exists w'$ s.t. $w' \in \mathcal{P}(\eta)(w)$ かつ $w' \in \mathcal{A}(\eta)$" が成り立つ．構成は通常の SST による逆像（4.2 節）の場合と同様に二段階に分けられる．

D.1 第一段階

Parikh SST $\mathcal{P} = (Q_1, X, L_1, \Delta_1, q_0^1, q_f^1, x_{out}, \varphi_1)$ と Parikh オートマトン $\mathcal{A} = (Q_2, L_2, \Delta_2, q_0^2, q_f^2, \varphi_2)$ から，まず中間的な構造

$\mathcal{M} = (Q, X, L_1, L_2, \Delta, Q_0, q_f, x_{out}, \varphi_1, \varphi_2)$ を構成する．部分構造 (Q, Δ, Q_0, q_f) は直積モノイド $\mathrm{Update}(X, \mathbb{N}^{L_2}) \times \mathbb{N}^{L_1}$ の元を出力するトランスデューサであり，$A^* \times \mathrm{Update}(X, \mathbb{N}^{L_2}) \times \mathbb{N}^{L_1}$ の部分集合と解釈される．この関係に関数 $\mathrm{Update}(X, \mathbb{N}^{L_2}) \ni \alpha \mapsto \tau(\alpha(x_{out})) \in \mathbb{N}^{L_2}$ と，$\varphi_1 \wedge \varphi_2$ の定める関係 $(\subseteq (\mathbb{N}^{L_2} \times \mathbb{N}^{L_1}) \times \mathbb{Z}^I)$ を順に合成することで $A^* \times \mathbb{Z}^I$

の部分集合となり，Parikh オートマトンと同じ型を持つ．

$\mathcal{M}$ の構成は 4.2.1 節の Update(X, M) 出力 SST の構成と同様だが，$\mathcal{P}$ による自然数ベクトル $\mathbb{N}^{L_1}$ の計算を模倣し，それを受理の判定に用いることが追加で必要になる．$\mathcal{M}$ の状態は通常の SST との合成の場合と同じで，$Q = Q_1 \times (X \rightharpoonup Q_2 \times Q_2)$, $Q_0 = \{(q_0^1, f_0) \mid \forall x, q, r. f_0(x) = (q, r) \Rightarrow q = r\}$, $q_f = (q_f^1, f_\$)$ ただし $dom(f_\$) = \{x_{out}\}$ かつ $f_\$(x_{out}) = (q_0^2, q_f^2)$ と定める．一方で遷移の集合 $\Delta \subseteq Q \times A_\$ \times (\text{Update}(X, \mathbb{N}^{L_2}) \times \mathbb{N}^{L_1}) \times Q$ には $\mathcal{P}$ の自然数ベクトル計算の模倣が加わる．そのため，$q_1, q_1' \in Q_1$, $f, f' \colon X \rightharpoonup Q_2 \times Q_2$, $a \in A_\$$, $v_1 \in \mathbb{N}^{L_1}$, $\alpha' \in \text{Update}(X, \mathbb{N}^{L_2})$ について，$(q_1, f) \xrightarrow[\Delta]{a/(\alpha', v_1)} (q_1', f')$ が次と同値であるように Δ を定義する：ある $\alpha \in \text{Update}(X, B)$ が存在して，$q_1 \xrightarrow[\Delta_1]{a/(\alpha, v_1)} q_1'$ かつ次の 3 つを全て満たす．

- $dom(f) = \cup_{x \in dom(f')} \text{Var}(\alpha(x))$,
- 任意の $x \in dom(f')$ について，$f'(x) = (q_2, q_2')$ とするとき $q_2 \xrightarrow[\Delta_2^{(f)}]{\alpha(x)\$a/\alpha'(x)} q_2'$,
- 任意の $x \notin dom(f')$ について $\alpha'(x) = \varepsilon$.

以上の $\mathcal{M}$ は逆像 $\mathcal{P}^{-1}(\mathcal{A})$ と等価である．

命題 D.1. *$\mathcal{M}$ は任意の $\eta \in \mathbb{Z}^I$ と $w \in A^*$ について “$w \in \mathcal{M}(\eta) \iff \exists w'$ s.t. $w' \in \mathcal{P}(\eta)(w)$ かつ $w' \in \mathcal{A}(\eta)$” を満たす．*

Proof. 命題 4.3 と同様にして示される． □

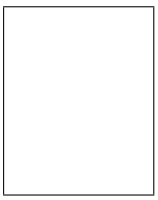
D.2 第二段階

前段の $\mathcal{M}$ を，等価な Parikh オートマトン $\mathcal{B}$ へと変形する．$\mathcal{B} = (Q', L, \Delta', Q_0', q_f', \varphi)$ であり，Q', Δ', Q_0', q_f' は以下のように構成する．状態については 4.2.2 節と同じで，$Q' = Q \times 2^X$, $Q_0 = \{(q_0, u) \mid q_0 \in Q_0, u \subseteq X\}$, $q_f' = (q_f, \{x_{out}\})$ とする．またラベル集合を $L = L_1 \cup L_2$ として，遷移集合 $\Delta' \subseteq Q' \times A_\$ \times \mathbb{N}^L \times Q'$ は $q \xrightarrow[\Delta]{a/(\alpha, v_1)} r$, $a \in A_\$$ であるとき任意の $u \in 2^X$ $(\subseteq \mathbb{N}^X)$ について $(q, \Phi(\alpha) \cdot u) \xrightarrow[\Delta']{a/v_1 \oplus (\tau(\alpha) \cdot u)} (r, u)$ とする．最後に，論理式は $\varphi \equiv \varphi_1 \wedge \varphi_2$ である．

このとき $\mathcal{B}$ は $\mathcal{M}$ と等価であり，したがって逆像 $\mathcal{P}^{-1}(\mathcal{A})$ と等価になる．

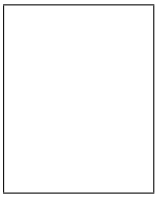
命題 D.2. *$\mathcal{B}$ は任意の $\eta \in \mathbb{Z}^I$ と $w \in A^*$ について $w \in \mathcal{B}(\eta) \iff w \in \mathcal{M}(\eta)$ を満たす．*

Proof. 命題 4.7 と同様にして示される． □



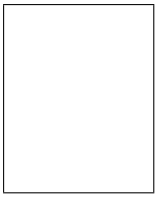
釜野雅基

2021 年東京工業大学情報理工学院数理・計算科学系学士課程卒業．現在，同大学大学院数理・計算科学系修士課程在籍．形式言語理論に興味を持つ．



福田大我

2020 年東京工業大学情報理工学院数理・計算科学系卒業．2022 年同大学大学院数理・計算科学系修士課程修了．プログラミング言語，形式言語理論に興味を持つ．



南出靖彦

1993 年京都大学大学院理学研究科数理解析専攻修士課程修了．同年同大学数理解析研究所助手．1999 年筑波大学電子・情報工学系講師．2007 年同大学大学院システム情報工学研究科准教授．2015 年東京工業大学大学院システム情報工学研究科教授．現在，同大学情報理工学院教授．博士（理学）．ソフトウェア検証，プログラミング言語，形式言語理論に興味を持つ．