Conditional Transformable Pushdown System: スタックの変換と検査が可能なプッシュダウンシステム

上里 友弥 [1], 南出 靖彦 [2]

[1] 筑波大学情報学群情報科学類
uezato@score.cs.tsukuba.ac.jp

[2] 筑波大学システム情報系情報工学域
minamide@cs.tsukuba.ac.jp

概 要　プッシュダウンシステム（PDS）はスタックを備えた計算モデルであり，再帰的なプログラムの解析などに用いられて来た．また，より複雑な解析の為にスタックの検査が出来る Conditional PDS と，スタックの変更が出来る Timed PDS などが提案され，それぞれ PDS へ還元出来ることが示されている．

本論文では，これら Conditional PDS と Timed PDS を統一したような，スタックに対する検査と変換の両方を行うことの出来る計算モデルである Conditional Transformable PDS（CTPDS）と，これを形式化する為に用いるトランスダクション規則で拡張した PDS（TrPDS）を提案する．特に TrPDS から PDS への変形により，CTPDS 上の位置到達可能性問題が決定可能であることを示す．

1　はじめに

プッシュダウンシステム（PDS）は，再帰的な振る舞いをするプログラムの解析の為の計算モデルとして広く研究されてきた [1]．特に PDS においては到達可能性問題（reachability problem）が決定可能であり [2, 3]，LTL によって拡張したモデル検査も決定可能であることが分かっている [4]．

PDS は色々な形で拡張されて来た．例えば，Esparza らは *pushdown system with checkpoints* という，スタックが与えられた正則言語に入るかどうかを調べる能力を持った PDS を提案し，*pushdown system with checkpoints* は PDS と等価であることと，実行中のスタック検査（stack inspection）が必要になるプログラム解析への応用を示した [5]．Li と小川はスタックを正則言語によって検査するというアイディアを，*conditional pushdown system* として見直し，特に重み付きプッシュダウンシステムを *conditional weighted pushdown system* へと拡張し，プログラム解析へ応用した [6]．

スタックの内容を変更することを許す拡張としては，離散時間の概念でスタックアルファベットを拡張した *Timed Pushdown Systems*[7] がある．Timed PDS ではスタックアルファベットが年齢によって拡張され，スタック中の全ての年齢を増やす遷移を持つ．特に Abdulla らは [7] において，Timed PDS の到達可能性問題が，従来の PDS の到達可能性問題へ還元出来ることを示している．

本論文では，スタックの変更と検査の両方を許す PDS として Conditional Transformable PDS（CTPDS）を提案し，位置到達可能性問題が決定可能であることを示す．CTPDS における検査はスタックが与えられた正則言語に入るかどうかを調べるもので，Li らの提案した Conditional PDS と同等である．一方 CTPDS における変換は，アルファベット上の関数をスタック全体に作用させてスタックを変換することを指し，Timed PDS を十分に実現出来るような，より一般化した操作である．

CTPDS の形式化及び，位置到達可能性問題が決定可能であることを示す為に，トランスダクション規則を備えた PDS，TrPDS を導入した．我々の方針は次のようである．TrPDS の位置到達可能性問題が，PDS の位置到達可能性問題に還元出来ることを示し，有限のスタックアルファベットを

持つ PDS に還元出来る場合には，前者の問題が決定可能であることを示す．次に，CTPDS で必要になる各操作をトランスダクションとして定義出来ることを示し，CTPDS を TrPDS の具体例として定義することによって，CTPDS の位置到達可能性問題が決定可能であることを示す．

TrPDS から PDS への問題の還元は，実際には PDS と本質的に同等な $\mathcal{F}$PDS というシステムを導入し，TrPDS から $\mathcal{F}$PDS の構成をすることで行うが，この構成は Abdulla らが [7] において Timed PDS から PDS を構成するアイディアから示唆を得たものである．

論文の構成は以下の通りである．2 節で，本論文で用いる有限 letter-to-letter トランスデューサ及びトランスダクションを定義する．3 節で，トランスダクション規則を備えた PDS，TrPDS を導入する．TrPDS は，トランスダクションを用いたスタックの変換を行う計算モデルであり，CTPDS の検査と変換はトランスダクションで自然に表現出来るので，TrPDS として CTPDS を扱うことが出来る．4 節と 5 節では，TrPDS と対になる計算モデルとして $\mathcal{F}$PDS を導入し，トランスダクションとその上の操作を代数系として捉えた $\mathcal{T}$-*Alg* を導入する．$\mathcal{F}$PDS は，TrPDS におけるスタック全体への作用をスタックトップに対する作用として表現する為の計算モデルであり，2 つのシステムの関係性として強双模倣性を示す．$\mathcal{T}$-*Alg* は，$\mathcal{F}$PDS のスタック上で（局所的に）トランスダクションを実行する為の代数である．6 節で与えられた TrPDS から $\mathcal{F}$PDS を構成し，7 節で TrPDS と構成された $\mathcal{F}$PDS の強双模倣性を示す．続く 8 節で TrPDS における位置到達可能性問題を，$\mathcal{F}$PDS における同様の問題に還元する．9 節では，CTPDS の位置到達可能性問題が決定可能であることを示す為に，CTPDS の実現に必要なトランスダクションの集合から有限 $\mathcal{T}$-*Alg* を構成する．

2 準備：有限 letter-to-letter トランスデューサ

アルファベット Γ 上の有限 letter-to-letter トランスデューサとは，$\Gamma \times \Gamma$ 上の有限オートマトンである．本論文では単に Γ 上のトランスデューサ，もしくは Γ が文脈から明らかな時はトランスデューサと書くことにする．Γ 上のトランスデューサ $\mathfrak{t}$ はオートマトンであるから，次のような 5 つ組で表すことが出来る．

$$\mathfrak{t} = (Q, \Gamma, \Delta, q_I, Q_F)$$

それぞれ Q は状態（state）の有限集合，Γ は有限のアルファベット，$\Delta \subseteq Q \times (\Gamma \times \Gamma) \times Q$ は有限の遷移関係，$q_I \in Q$ は初期状態（initial state）で，$Q_F \subseteq Q$ は受理状態（final state) の有限集合である．

Γ 上のトランスデューサ $\mathfrak{t}$ に対する言語 $L(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$ もオートマトン同様に定義される．トランスデューサのアルファベット $\langle i, o\rangle \in \Gamma \times \Gamma$ は，入力が i，出力が o であると考えられて，この時に言語 $L(\mathfrak{t})$ は文字列から文字列の集合を生成する関数 $\mathfrak{t} : \Gamma^* \to \mathcal{P}(\Gamma^*)$ として捉えられる．特に文字列 w に対する作用が次のように定義される．

$$\mathfrak{t}(w) = \mathfrak{t}(a_1 \cdots a_n) = \{w' \mid w' = b_1 \cdots b_n \ , \ \langle a_1, b_1\rangle \cdots \langle a_n, b_n\rangle \in L(\mathfrak{t})\}$$

一般に $\tau \in \Gamma^* \to \mathcal{P}(\Gamma^*)$ は，Γ^* から Γ^* へのトランスダクション（transduction）と呼ばれ，特に上の $\mathfrak{t}$ のように（letter-to-letter な）トランスデューサで特徴付けられる場合は，length-preserving transduction と呼ばれる．またトランスダクション τ は，Γ^* 上の関係 $\tau \subseteq \Gamma^* \times \Gamma^*$ としても捉えることができ，以後関係としても用いることにする．

ここで Γ 上のトランスデューサによって特徴付けられる length-preserving transduction 全体を $\mathbf{Transd}_\Gamma$ と書くことにする．$\mathbf{Transd}_\Gamma$ は，$\Gamma \times \Gamma$ 上の正則言語全体と明らかに同型である．また，以後トランスデューサ $\mathfrak{t}$ とトランスダクション $\mathfrak{t}$ は，特に区別せずに書くことにする．

トランスダクションの作用を，次のように言語に拡張する．

$$\mathfrak{t}(L) = \bigcup_{w \in L} \mathfrak{t}(w)$$

トランスダクション上の重要な演算として合成がある．$\mathtt{t}_1, \mathtt{t}_2 \in \mathbf{Transd}_\Gamma$ の合成は，

$$\mathtt{t}_2 \circ \mathtt{t}_1 = \{\langle w, w'' \rangle \mid \langle w, w' \rangle \in \mathtt{t}_1, \langle w', w'' \rangle \in \mathtt{t}_2\}$$

として定義され，$\mathtt{t}_2 \circ \mathtt{t}_1 \in \mathbf{Transd}_\Gamma$ となる．また，合成については $\mathtt{t}_2(\mathtt{t}_1(w)) = (\mathtt{t}_2 \circ \mathtt{t}_1)(w)$ が成立する．特に次のトランスデューサ $\mathbb{1}$ は，合成についての単位元になっている．

$$\mathbb{1} = (\{q\}, \Gamma, \Delta, \{q\}, \{q\}) \quad \Delta = \{\langle q, \langle \gamma, \gamma \rangle, q \rangle \mid \gamma \in \Gamma\}$$

$\mathtt{t}$ は $\Gamma \times \Gamma$ 上の正則言語としてみられるので，その商（left quotient）を考えることができ，特にこれを effective に計算出来る [8, 9].

$$\langle \gamma, \gamma' \rangle^{-1} \mathtt{t} = \mathtt{t}' \quad \text{ただし } \mathtt{t}' = \{\langle w, w' \rangle \mid \langle \gamma w, \gamma' w' \rangle \in \mathtt{t}\}$$

また，商を用いても入力 w に対するトランスダクションの作用を定義することが出来る．

$$\mathtt{t}(\lambda) = \nu(\mathtt{t}) \quad \nu(\mathtt{t}) = \begin{cases} \{\lambda\} & \langle \lambda, \lambda \rangle \in \mathtt{t} \\ \varnothing & \text{otherwise} \end{cases}$$

$$\mathtt{t}(\gamma w) = \bigcup_{\gamma' \in \Gamma} \gamma' \lhd \left(\langle \gamma, \gamma' \rangle^{-1} \mathtt{t}\right)(w)$$

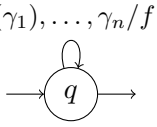
ただし，$w \lhd W = \{ww' \mid w' \in W\}$ である．また，$\lambda \in \Gamma^*$ は空文字列である．

例 1：正則言語から生成されるトランスダクション　L を Γ 上の正則言語であるとすると，これを受理する有限オートマトン A が存在し，$A = (Q, \Gamma, \Delta, q_I, Q_F)$ と書くことが出来る．この時，

$$\widetilde{A} = (Q, \Gamma, \Delta', q_I, Q_F) \quad \Delta' = \{\langle p, \langle \gamma, \gamma \rangle, q \rangle \mid \langle p, \gamma, q \rangle \in \Delta\}$$

としてトランスデューサ $\widetilde{A}$ を定義すると，トランスダクションとして見た時に $w' \in \widetilde{A}(w) \iff w' = w$ かつ $w \in L(A)$ が成立する．

例 2：Γ 上の関数から生成されるトランスダクション　Γ 上の関数 $f \in \Gamma \to \Gamma$ が与えられた時に，次のようなトランスデューサ $\widehat{f}$ を構成出来る．



$$\widehat{f} = (\{q\}, \Gamma, \{\langle \gamma, f(\gamma) \rangle \mid \gamma \in \Gamma\}, q, \{q\})$$

この時，トランスダクション $\widehat{f}$ について $\widehat{f}(c_1 c_2 \cdots c_n) = \{f(c_1) f(c_2) \cdots f(c_n)\}$ が成立し，$\widehat{f}$ は f を文字列上に拡張した関数と同一視出来る．

3　トランスダクション規則で拡張された PDS（TrPDS）

TrPDS は，トランスダクションによってスタックの変換が出来るように拡張した PDS である．TrPDS は次のような 4 つ組で表される．

$$\text{TrPDS } \mathcal{S} = (Q, \Gamma, \mathcal{T}, \Delta)$$

それぞれ Q は位置（control location）を表す有限集合，Γ はスタックアルファベットを表す有限集合，$\mathcal{T} \subseteq \mathbf{Transd}_\Gamma$ はトランスダクションの有限集合，$\Delta \subseteq (Q \times \Gamma \times Q) \uplus (Q \times Q) \uplus (Q \times \mathcal{T} \times Q)$ は遷移規則を表す有限集合となっている．ただし，$\mathbb{1} \in \mathcal{T}$ とする．

TrPDS $\mathcal{S}$ における計算状況（configuration）を，位置 q とスタックアルファベット上の文字列 w を用いて対 $\langle q, w \rangle$ で表す．従って，計算状況の全体 $Conf(\mathcal{S})$ は次のように定義される．

$$Conf(\mathcal{S}) = Q \times \Gamma^*$$

定義 **1** (遷移関係)**.** TrPDS 上の遷移を表すラベル（$\delta \in \Delta$）付きの遷移関係

$$\xrightarrow{\delta} \subseteq \mathit{Conf}(\mathcal{S}) \times \mathit{Conf}(\mathcal{S})$$

は，次のように遷移規則をもとにして定義される．

Push $\delta = \langle p, \gamma, q\rangle$ の時，$\langle p, w\rangle \xrightarrow{\delta} \langle q, \gamma w\rangle$

Pop $\delta = \langle p, q\rangle$ の時，$\langle p, \gamma w\rangle \xrightarrow{\delta} \langle q, w\rangle$

Transduce $\delta = \langle p, \mathfrak{t}, q\rangle$ の時，$\langle p, w\rangle \xrightarrow{\delta} \langle q, w'\rangle$　ただし $w' \in \mathfrak{t}(w)$

TrPDS には，スタックトップのアルファベットを調べるような遷移規則は無いが，Transduce 遷移規則を用いて表現出来る．例えばスタックトップが $\gamma \in \Gamma$ であるかどうかを調べたい場合には，$\mathfrak{t} = (\gamma, \gamma)(\Gamma, \Gamma)^*$ であるようなトランスダクション $\mathfrak{t}$ を用いれば良い．これによって PDS を TrPDS で実現することが出来る．

例 **1 2-**計数機械の実現　TrPDS $\mathcal{S} = (Q, \Gamma = \{0, 1, \varepsilon\}, \{\mathfrak{d}_0, \mathfrak{t}_0, \mathfrak{d}_1, \mathfrak{t}_1\}, \Delta)$ によって（ε は空文字列ではなく単にアルファベットであることに注意），カウンタ $0, 1$ を持つ 2-計数機械を実現することが出来る．ここで $\mathfrak{d}_0$ はスタックから 0 をデクリメントするようなトランスダクションで，$\mathfrak{d}_0 = [(1,1)|(\varepsilon,\varepsilon)]^*(0,\varepsilon)(\Gamma,\Gamma)^*$ となる．これから分かるように，アルファベット ε で消去を表す．また，$\mathfrak{t}_0$ はスタックに 0 が含まれているかどうかを調べるトランスダクションで，$\mathfrak{t}_0 = (\Gamma,\Gamma)^*(0,0)(\Gamma,\Gamma)^*$ となる．カウンタ 1 についても同様に定義する．このようにして 2-計数機械を実現出来るので，TrPDS は一般にチューリング完全であることが分かる．

例 **2 Conditional PDS** の実現　Conditional PDS は，スタックを文字列としてみた時に，それが与えられた正則言語に入るかどうかを調べることが出来る PDS である．与えられた言語は正則であるからそれを受理する有限オートマトン A が存在し，$\widetilde{A}$ を用いて TrPDS で Conditional PDS を実現出来る．

4　$\mathcal{F}$PDS

TrPDS はスタック全体に対する処理を行う計算モデルであるが，その各処理をスタックトップへの処理だけが許される PDS で摸倣する．しかし通常の PDS で摸倣しようとすると，TrPDS の 1 つの遷移を PDS では複数の遷移によって表現することになり，構成と証明が複雑になる．そこで TrPDS の 1 つの遷移を 1 つの遷移で表現出来て，かつスタックトップへの処理だけを行う，すなわち PDS にコンパイルすることが出来るような計算モデルとして $\mathcal{F}$PDS を導入する．

$\mathcal{F}$PDS はスタックトップ 2 文字以下に対する関数適用（$\mathcal{F}$unction application）を行うことを意図しており，次のように 3 つ組で表される．

$$\mathcal{F}\text{PDS } \mathcal{S} = (Q, \Gamma, \Delta)$$

TrPDS と同様に，Q は位置を表す有限集合，Γ はスタックアルファベットを表す有限集合である．ただし，遷移規則の集合 Δ に関しては次のように少し複雑になる．

$$\Delta \subseteq Q \times \mathcal{P}(\Gamma^*) \times (\Gamma \to \mathcal{P}(\Gamma^*)) \times (\Gamma^2 \to \mathcal{P}(\Gamma^*)) \times Q$$

$\mathcal{F}$PDS $\mathcal{S}$ における計算状況を，位置 q とスタックアルファベット上の文字列 w を用いて対 $\langle q, w\rangle$ で表す．従って，計算状況の全体 $\mathit{Conf}(\mathcal{S})$ は次のように定義される．

$$\mathit{Conf}(\mathcal{S}) = Q \times \Gamma^*$$

定義 **2** (遷移関係)**.** $\mathcal{F}$PDS 上の遷移を表すラベル（$\delta \in \Delta$）付きの遷移関係

$$\xrightarrow{\delta} \subseteq \mathit{Conf}(\mathcal{S}) \times \mathit{Conf}(\mathcal{S})$$

は，次のように遷移規則をもとにして定義される．

Apply　$\delta = \langle p, f_0, f_1, f_2, q\rangle$ において，

1. $\langle p, \lambda\rangle \xrightarrow{\delta} \langle q, w\rangle$ ただし $w \in f_0$ これはスタックが空の時の振る舞いを表している．
2. $\langle p, \gamma\rangle \xrightarrow{\delta} \langle q, w\rangle$ ただし $w \in f_1(\gamma)$ これはスタックが 1 文字の時の，その 1 文字への関数適用を表している．
3. $\langle p, \gamma\gamma' w\rangle \xrightarrow{\delta} \langle q, w' w\rangle$ ただし $w' \in f_2(\gamma\gamma')$ これはスタックに 2 文字以上ある時の，上部 2 文字への関数適用を表している．

push と pop に直接対応する遷移規則はないが，Apply 遷移規則を用いれば同等の遷移関係を生成出来る．

5　$\mathcal{T}$-*Alg* ：トランスダクションの代数的表現

トランスダクションの有限部分集合 $\mathcal{T} \subseteq \mathbf{Transd}_\Gamma$ をもとにして，その上の合成と商に関して閉じた代数系，$\mathcal{T}$-*Alg* を導入する．$\mathcal{T}$-*Alg* は，6 つ組 $(\mathcal{U}, \bullet, \langle\cdot,\cdot\rangle^{-1}, \nu(\cdot), F, G)$ で表され，特に演算と関数は以下のように定義される．

定義 3.

1. $\bullet : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$ トランスダクションにおける合成に対応する 2 項演算である．
2. $\langle\cdot,\cdot\rangle^{-1} : \Gamma \times \Gamma \times \mathcal{U} \to \mathcal{U}$ トランスダクションにおける商に対応する 3 項演算である．
3. $\nu(\cdot) : \mathcal{U} \to \mathcal{P}(\Gamma^*)$ ただし，$\nu(\mathfrak{u}) = \{\lambda\}$ もしくは $\nu(\mathfrak{u}) = \varnothing$ のどちらかである．この単項演算は，トランスダクションにおける λ から λ を生成出来るかどうかを調べる演算 ν に対応している．
4. $F : \mathcal{T} \to \mathcal{U}$ トランスダクションの有限集合 $\mathcal{T}$ を，$\mathcal{T}$-*Alg* に埋め込む関数である．
5. $G : \mathcal{U} \to \mathbf{Transd}_\Gamma$ $\mathcal{T}$-*Alg* から，トランスダクションの空間へ引き戻す関数である．

トランスダクション空間からの埋め込みと引き戻し　トランスダクション空間からの埋め込みが F で，トランスダクション空間への引き戻しが G となっている．この時，2 つの関数に対して以下の条件が成立する必要がある．

1. $G \circ F = \mathrm{id}$ この条件は，トランスダクション空間から F によって $\mathcal{T}$-*Alg* へ埋め込んだものは，G によってもとに戻すことが出来ることを表している．
2. $\nu(\mathfrak{u}) = \nu(G(\mathfrak{u})), G(\mathfrak{u}_2 \bullet \mathfrak{u}_1) = G(\mathfrak{u}_2) \circ G(\mathfrak{u}_1), G(\langle\gamma,\gamma'\rangle^{-1}\mathfrak{u}) = \langle\gamma,\gamma'\rangle^{-1} G(\mathfrak{u})$ これらの条件は，G が準同型写像になっていることを要求する．

既に商によるトランスデューサの作用を定義したが，$\mathcal{T}$-*Alg* においても入力に対する作用 $\mathfrak{u}(\cdot) : \Gamma^* \to \mathcal{P}(\Gamma^*)$ を以下のように定義し，同時に言語に拡張する．

$$\mathfrak{u}(\lambda) = \nu(\mathfrak{u}) \quad \mathfrak{u}(\gamma w) = \bigcup_{\gamma' \in \Gamma} \gamma' \lhd \left(\langle\gamma,\gamma'\rangle^{-1}\mathfrak{u}\right)(w)$$

$$\mathfrak{t}(L) = \bigcup_{w \in L} \mathfrak{t}(w)$$

この時，以下の重要な性質が成立する．

補題 1.

$$\mathfrak{u}(w) = G(\mathfrak{u})(w) \quad \mathfrak{t}(w) = F(\mathfrak{t})(w) \quad \mathfrak{u}_2(\mathfrak{u}_1(w)) = (\mathfrak{u}_2 \bullet \mathfrak{u}_1)(w)$$

6　TrPDS から $\mathcal{F}$PDS を構成する

TrPDS $(Q, \Gamma, \mathcal{T}, \Delta)$ が与えられた時に，$\mathcal{T}$ を代数化して得られる $\mathcal{T}$-Alg $(\mathcal{U}, \bullet, \langle\cdot,\cdot\rangle^{-1}, \nu(\cdot), F, G)$ を用いて $\mathcal{F}$PDS $(Q, \Gamma \times \mathcal{U}, \Delta')$ を構成し，これを用いて模倣を行う．Δ' は以下のように構成する．

TrPDS での Push γ　$\langle p, \gamma, q\rangle \in \Delta$ という遷移規則がある時，$\langle p, f_0, f_1, f_2, q\rangle$ という遷移規則を Δ' に加える．

$$f_0 = \{\langle \gamma, F(\mathbb{1})\rangle\} \quad f_1(c) = \{\langle \gamma, F(\mathbb{1})\rangle\, c\} \quad f_2(c_1 c_2) = \{\langle \gamma, F(\mathbb{1})\rangle\, c_1 c_2\}$$

この時，この遷移規則から生成される遷移は次のようにスタック上の関数として捉えられる．

$$\mathrm{push}_\gamma(w) = \{\langle \gamma, F(\mathbb{1})\rangle\, w\}$$

TrPDS での Transduce $\mathfrak{t}$　$\langle p, \mathfrak{t}, q\rangle \in \Delta$ という遷移規則がある時，$\langle p, f_0, f_1, f_2, q\rangle$ という遷移規則を Δ' に加える．

$$f_0 = \nu(F(\mathfrak{t})) \quad f_1(\langle \gamma, \mathfrak{a}\rangle) = \{\langle \gamma, F(\mathfrak{t}) \bullet \mathfrak{a}\rangle\} \quad f_2(\langle \gamma, \mathfrak{a}\rangle \langle \gamma', \mathfrak{b}\rangle) = \{\langle \gamma, F(\mathfrak{t}) \bullet \mathfrak{a}\rangle \langle \gamma', \mathfrak{b}\rangle\}$$

この時，この遷移規則から生成される遷移は次のようにスタック上の関数として捉えられる．

$$T_\mathfrak{t}(\lambda) = \nu(F(\mathfrak{t})) \quad T_\mathfrak{t}(\langle \gamma, \mathfrak{a}\rangle\, w) = \{\langle \gamma, F(\mathfrak{t}) \bullet \mathfrak{a}\rangle\, w\}$$

TrPDS での Pop　$\langle p, q\rangle \in \Delta$ という遷移規則がある時，$\langle p, \varnothing, f_1, f_2, q\rangle$ という遷移規則を Δ' に加える．

$$f_1(\langle \gamma, \mathfrak{a}\rangle) = \bigcup_{\gamma' \in \Gamma} \nu\left(\langle \gamma, \gamma'\rangle^{-1}\, \mathfrak{a}\right)$$
$$f_2(\langle \gamma, \mathfrak{a}\rangle \langle \gamma', \mathfrak{b}\rangle) = \bigcup_{\gamma'' \in \Gamma} \left\{\langle \gamma', \mathfrak{a}' \bullet \mathfrak{b}\rangle \mid \mathfrak{a}' = \langle \gamma, \gamma''\rangle^{-1}\, \mathfrak{a}\right\}$$

この時，この遷移規則から生成される遷移は次のようにスタック上の関数として捉えられる．

$$\mathrm{pop}(\lambda) = \varnothing$$
$$\mathrm{pop}(\langle \gamma, \mathfrak{a}\rangle) = \bigcup_{\gamma' \in \Gamma} \nu\left(\langle \gamma, \gamma'\rangle^{-1}\, \mathfrak{a}\right)$$
$$\mathrm{pop}(\langle \gamma, \mathfrak{a}\rangle \langle \gamma', \mathfrak{b}\rangle\, w) = \bigcup_{\gamma'' \in \Gamma} \left\{\langle \gamma', \mathfrak{a}' \bullet \mathfrak{b}\rangle\, w \mid \mathfrak{a}' = \langle \gamma, \gamma''\rangle^{-1}\, \mathfrak{a}\right\}$$

7　TrPDS と $\mathcal{F}$PDS の強双模倣性

TrPDS と $\mathcal{F}$PDS の関係性を表すのに強双模倣性（strong bisimilarity）を示したい．

$$\begin{array}{ccc} X & \sim & Y \\ {\scriptstyle t}\downarrow & & \downarrow{\scriptstyle t} \\ X' & \sim & Y' \end{array}$$

$\sim$ を適当な双模倣関係（bisimulation）としたいが，これを単に計算状況上の 2 項関係だと捉えても上手くいかない．TrPDS では Transduce 遷移によって非決定的に遷移し，$\mathcal{F}$PDS では Transduce 遷移規則と Pop 遷移規則から生成した遷移で非決定的に遷移するからである．そこで計算状況の集合上の双模倣関係を考える．

計算状況の集合上の双模倣関係を定義するのに，TrPDS と $\mathcal{F}$PDS でのスタックの関係を考える．TrPDS は与えられたトランスダクションを実際に利用して遷移を行うので，その都度スタック全体

が変換され，同時に遷移の成功と失敗が直ちに反映される．一方 $\mathcal{F}$PDS ではスタックトップにおける合成と商の計算を行うのみで，スタック全体への作用ではない．

そこで $\mathcal{F}$PDS の計算状況のスタックを具体化（concretization）し，スタックが表す文字列の集合を生成する．スタックを具体化した結果は，TrPDS の計算状況の集合と等しい筈であるというアイディアをもとに双摸倣関係を定義する．

定義 4 (スタックの具体化)**.** $\mathcal{F}$PDS のスタックから文字列の集合を生成する．

$$\begin{aligned}
&\|\cdot\| : (\Gamma \times \mathcal{T}\text{-}Alg)^* \to \mathcal{P}(\Gamma^*)\\
&\|\lambda\| = \{\lambda\}\\
&\|\langle\gamma, \mathfrak{t}\rangle\, w\| = \mathfrak{t}(\gamma \lhd \|w\|)
\end{aligned}$$

補題 2. $\mathcal{F}$PDS の構成で導入した関数 push_γ，$T_\mathfrak{t}$，pop と具体化に関して以下の交換法則が成立する．

$$\begin{aligned}
&\left\|\mathrm{push}_\gamma(w)\right\| = \gamma \lhd \|w\|\\
&\|T_\mathfrak{t}(w)\| = \mathfrak{t}\,(\|w\|)\\
&\|\mathrm{pop}(w)\| = \Gamma^{-1}\|w\|
\end{aligned}$$

ただし $\Gamma^{-1}w = \{x \mid \gamma \in \Gamma, \gamma x = w\}$ で，$\Gamma^{-1}W$ として集合上に拡張する．

TrPDS の計算状況の集合 X と $\mathcal{F}$PDS の計算状況の集合 Y について，次のような関係 $\sim$ を定義する．

$$\begin{aligned}
\|Y\| &= \{\langle q, x\rangle \mid \langle q, w\rangle \in Y, x \in \|w\|\}\\
X \sim Y &\iff X = \|Y\|
\end{aligned}$$

定義 5 (集合上への遷移関係の拡張)**.** TrPDS $\mathcal{S}$ の計算状況の集合 $X \subseteq \mathit{Conf}(\mathcal{S})$ が与えられた時に，次のようにして遷移規則 t を用いた集合上の遷移を定義する．

$$X \overset{t}{\twoheadrightarrow} X' = \{x' \mid x \in X, x \overset{t}{\to} x'\}$$

$\mathcal{F}$PDS に関しても同様に定義する．

定理 1 (強双摸倣性)**.** $X \sim Y$ は $\twoheadrightarrow$ に関して強双摸倣性を満たす．

$$\begin{array}{ccc}
X & \sim & Y\\
{\scriptstyle t}\big\downarrow & & \big\downarrow{\scriptstyle t}\\
X' & \sim & Y'
\end{array}$$

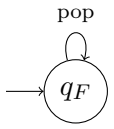
このことは，補題 2 により示せる．

8 位置到達可能性問題の決定可能性

与えられた位置から与えられた位置に到達出来るかどうかを考える問題を，位置到達可能性問題（location reachability problem）と呼び，次のように定義する．

位置到達可能性問題 与えられた TrPDS $\mathcal{S} = (Q, \Gamma, \mathcal{T}, \Delta)$ と $q_I \in Q$，及び $q_F \in Q$ について，$\exists w \in \Gamma^*.\, \langle q_I, \lambda\rangle \overset{*}{\to} \langle q_F, w\rangle$ が成立するか否か?

我々は直接この問題に答えるのではなく，次のように q_F について新たな遷移規則を加え，この状況で q_F に空スタックで到達出来るかどうかを考えることにする．



q_F に何らかのスタックで到達出来るならば，上の遷移規則によって空スタックにすることが出来る．逆に q_F に空スタックで到達出来るならば，当然 q_F に何らかのスタックでは到達出来ているわけなので，空スタックでの到達はもとの位置到達可能性問題と同等になる．

実際には TrPDS の q_F に空スタックで到達出来るかどうかを考えるのではなく，$\mathcal{F}$PDS の q_F に空スタックで到達出来るかどうかを考えるような問題の還元を行う．すなわち以下が成立するかどうかを考える．

$$\text{TrPDS} : \langle q_I, \lambda\rangle \xrightarrow{*} \langle q_F, \lambda\rangle \overset{\text{還元}}{\Longrightarrow} \mathcal{F}\text{PDS} : \langle q_I, \lambda\rangle \xrightarrow{*} \langle q_F, \lambda\rangle$$

還元の健全性　まず $\mathcal{F}$PDS への問題の還元が健全であることを示す．すなわち $\mathcal{F}$PDS で q_F に空スタックで到達出来るならば，TrPDS でも q_F に空スタックで到達出来ることを示す．

$\mathcal{F}$PDS では到達出来るので，次のようなトレースが存在する．

$$\mathcal{F}\text{PDS} : \langle q_I, \lambda\rangle \xrightarrow{t_1} c_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} c_n = \langle q_F, \lambda\rangle$$

このトレースを集合上に拡張すると次のようになる．

$$\mathcal{F}\text{PDS} : \{\langle q_I, \lambda\rangle\} \overset{t_1}{\twoheadrightarrow} C_1 \overset{t_2}{\twoheadrightarrow} \cdots \overset{t_n}{\twoheadrightarrow} C_n \ni c_n$$

$\langle q_F, \lambda\rangle \in C_n$ であるから，$\varnothing \neq \|C_n\|$，特に $\langle q_F, \lambda\rangle \in \|C_n\|$ が明らか．従って全ての $\|C_i\|$ が空でないことが分かる．これより強双模倣性から以下が成立する．

$$\text{TrPDS} : \{\langle q_I, \lambda\rangle\} \overset{t_1}{\twoheadrightarrow} C'_1 \overset{t_2}{\twoheadrightarrow} \cdots \overset{t_n}{\twoheadrightarrow} C'_n \ni \langle q_F, \lambda\rangle$$

よって次のように TrPDS でも q_F に空スタックで到達出来ることが分かる．

$$\text{TrPDS} : \langle q_I, \lambda\rangle \xrightarrow{t_1} c'_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} c'_n = \langle q_F, \lambda\rangle$$

還元の完全性　次に $\mathcal{F}$PDS への問題の還元が完全であることを示す．すなわち TrPDS で q_F に空スタックで到達出来るならば，$\mathcal{F}$PDS でも q_F に空スタックで到達出来ることを示す．

TrPDS では到達出来るので，次のようなトレースが存在する．

$$\text{TrPDS} : \langle q_I, \lambda\rangle \xrightarrow{t_1} c_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} c_n = \langle q_F, \lambda\rangle$$

このトレースを集合上に拡張すると次のようになる．

$$\text{TrPDS} : \{\langle q_I, \lambda\rangle\} \overset{t_1}{\twoheadrightarrow} C_1 \overset{t_2}{\twoheadrightarrow} \cdots \overset{t_n}{\twoheadrightarrow} C_n \ni c_n$$

この時，C_i は全て空でない．従って，強双模倣性より以下が成立する．

$$\mathcal{F}\text{PDS} : \{\langle q_I, \lambda\rangle\} \overset{t_1}{\twoheadrightarrow} C'_1 \overset{t_2}{\twoheadrightarrow} \cdots \overset{t_n}{\twoheadrightarrow} C'_n$$

特に $C_n = \|C'_n\|$ であるから，$\langle q_F, \lambda\rangle \in C'_n$ でなければならない．よって次のように $\mathcal{F}$PDS でも q_F に到達出来ることが分かる．

$$\mathcal{F}\text{PDS} : \langle q_I, \lambda\rangle \xrightarrow{t_1} c'_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} c'_n = \langle q_F, \lambda\rangle$$

以上で $\mathcal{F}$PDS への位置到達可能性問題の還元が出来た．$\mathcal{F}$PDS への還元に用いる $\mathcal{T}$-*Alg* を有限とできるならば，従来の PDS の到達可能性問題が決定可能であることを用いて，TrPDS の位置到達可能性問題が決定可能であることが分かる．

9　Conditional Transformable PDS と $\mathcal{T}$-*Alg*

CTPDS はスタックに対する正則言語を用いた検査と，アルファベット上の書換えをスタック全体に拡張した変換を許したような PDS であると述べたが，TrPDS を用いると次のように定義することが出来る．

定義 6 (Conditional Transformable PDS)**.** CTPDS とは TrPDS $\mathcal{S} = (Q, \Gamma, \mathcal{T}, \Delta)$ である．ここで，$\mathcal{T}$ は次のように定義される．

$$\mathcal{T} = \widetilde{\mathcal{D}} \uplus \widehat{\mathcal{F}}$$
$$\widetilde{\mathcal{D}} = \left\{ \widetilde{A} \mid L(A) = L, L \in \mathcal{D} \right\}$$
$$\widehat{\mathcal{F}} = \left\{ \widehat{f} \mid f \in \mathcal{F} \right\} \cup \{\mathbb{1}\}$$

ただし $\mathcal{D}$ は Γ 上の正則言語からなる有限の族で，定義中の $\widetilde{A}$ は，例 1 で定義した有限オートマトン A の受理言語から生成される関係である．また $\mathcal{F}$ は Γ 上の関数の集合 $\mathcal{F} \subseteq \Gamma \to \Gamma$ で，定義中の $\widehat{f}$ は例 2 で定義した Γ 上の関数を文字列上に拡張したものである．$\widetilde{\mathcal{D}}$ で正則言語を用いた検査を表し，$\widehat{\mathcal{F}}$ でスタックの変換を表す．

有限スタックアルファベットの $\mathcal{F}$PDS を構成したいので，この $\mathcal{T}$ に対応する有限の $\mathcal{T}$-*Alg* を構成する．そのような $\mathcal{T}$-*Alg* として $B = (\mathcal{C} \times \mathcal{M}, \bullet, \langle \cdot, \cdot \rangle^{-1}, \nu(\cdot), F, G)$ を考える．まず集合 $\mathcal{M}$ を次のように定義する．

$$\mathcal{M} = (\widehat{\mathcal{F}}, \circ) \text{ で生成される集合}$$

$\mathcal{M}$ は明らかに有限集合で，準同型写像の集合とみなすことができる．$\mathcal{C}$ は，$\mathcal{D}$ と $\mathcal{M}$ を用いて構成される．

$$\mathcal{D}_1 = \left\{ w^{-1}L \mid w \in \Gamma^*, L \in \mathcal{D} \right\} \cup \{\varnothing, \Gamma^*\}$$
$$\mathcal{D}_2 = \mathcal{M}^{-1}(\mathcal{D}_1) = \bigcup_{m \in \mathcal{M}} m^{-1}(\mathcal{D}_1)$$
$$\mathcal{C} = (\mathcal{D}_2, \cap) \text{ で生成される集合}$$

$\mathcal{D}_1$ は有限個の正則言語に対する商であるから有限（加えて effective に計算可能）．$\mathcal{D}_2$ も有限になる（正則言語に対する準同型写像の逆像は正則集合で effective に計算可能）．$\mathcal{C}$ の要素は，$\cap$ の可換性と冪等性から $\mathcal{D}_2$ の要素を有限個の $\cap$ で結合したもので表現されるため，$\mathcal{C}$ は有限集合となる．

この時，演算と関数は以下のように定義される．

定義 7.

$$\begin{aligned}
F(\mathfrak{t}) &= \begin{cases} \langle c, \mathbb{1} \rangle & \widetilde{c} = \mathfrak{t} \in \widetilde{\mathcal{D}} \\ \langle \Gamma^*, \mathfrak{t} \rangle & \mathfrak{t} \in \widehat{\mathcal{F}} \end{cases} \\
G(\langle c, m \rangle) &= m \circ \widetilde{c} \\
\nu(x) &= \nu(G(x)) \\
\langle c_2, m_2 \rangle \bullet \langle c_1, m_1 \rangle &= \left\langle m_1^{-1}(c_2) \cap c_1, m_2 \circ m_1 \right\rangle \\
\langle \gamma, \gamma' \rangle^{-1} \langle c, m \rangle &= \begin{cases} \langle \gamma^{-1} c, m \rangle & \gamma' = m(\gamma) \\ \langle \varnothing, \mathbb{1} \rangle & \gamma' \neq m(\gamma) \end{cases}
\end{aligned}$$

(ただし $\langle \gamma, \gamma' \rangle^{-1}$ の定義において，$m = \langle \gamma, \gamma' \rangle^{-1} m$ という関係を用いた)

必要な関数と演算が定義出来たので，この時 B が $\mathcal{T}$-*Alg* になっていることを確認する．

B が $\bullet$ で閉じていること

定理 2.

$$\forall m \in \mathcal{M} \,.\, m^{-1}(\mathcal{C}) \subseteq \mathcal{C}$$

証明. 任意の $c \in \mathcal{C}$ について，$m^{-1}(c) \in \mathcal{C}$ であることを証明すれば良い．$c = m_1^{-1}(d_1^1) \cap \cdots \cap m_n^{-1}(d_n^1)$ と書ける．この時，以下が成立するので明らか．

$$\begin{aligned}
m^{-1}\left(m_1^{-1}(d_1^1) \cap \cdots \cap m_n^{-1}(d_n^1) \right) &= m^{-1}(m_1^{-1}(d_1^1)) \cap \cdots \cap m^{-1}(m_n^{-1}(d_n^1)) \\
&= (m_1 \circ m)^{-1}(d_1^1) \cap \cdots \cap (m_n \circ m)^{-1}(d_n^1)
\end{aligned}$$

系 **1.** $\mathcal{C} \times \mathcal{M}$ は演算 $\bullet$ で閉じている．

B が $\langle \gamma, \gamma' \rangle^{-1}$ で閉じていること

補題 **3.**

$$\forall c \in \mathcal{C}, m \in \mathcal{M} \,.\, \gamma^{-1}\left(m^{-1}(c)\right) = m^{-1}\left(m(\gamma)^{-1}(c)\right)$$

定理 **3.**

$$\gamma^{-1}(\mathcal{C}) \subseteq \mathcal{C}$$

系 **2.** $\mathcal{C} \times \mathcal{M}$ は演算 $\langle \gamma, \gamma' \rangle^{-1}$ で閉じている．

B が $\mathcal{T}$-*Alg* であることを示すのに必要な自明でない性質は次の二つである．

1. $G(\mathfrak{u}_2 \bullet \mathfrak{u}_1) = G(\mathfrak{u}_2) \circ G(\mathfrak{u}_1)$
2. $G(\langle \gamma, \gamma' \rangle^{-1} \mathfrak{u}) = \langle \gamma, \gamma' \rangle^{-1} G(\mathfrak{u})$

1 については次の補題によって示される．

補題 **4.**

$$\forall c \in \mathcal{C}, m \in \mathcal{M}.\ \widetilde{c} \circ m = m \circ \widetilde{m^{-1}(c)}$$
$$\forall c_1, c_2 \in \mathcal{C}.\ \widetilde{c_2} \circ \widetilde{c_1} = \widetilde{c_2 \cap c_1}$$

2 については次の補題によって示される．

補題 **5.**

$$\forall t_1, t_2 \in \mathbf{Transd}_\Gamma.\ \langle a, c \rangle^{-1}(t_2 \circ t_1) = \bigcup_{b \in \Gamma} \left(\langle b, c \rangle^{-1} t_2 \circ \langle a, b \rangle^{-1} t_1 \right)$$

10 関連研究

PDS 上の到達可能性解析については [3, 4, 2] がある．与えられた PDS と計算状況の正則集合に対し，その集合に到達出来る計算状況全体の集合（pre*）と，その集合から到達出来る計算状況全体の集合（post*）が共に effective に計算でき，特にそれらの集合が正則集合となることを示している．

有限 letter-to-letter トランスデューサを用いたモデル検査の研究としては，Regular Model Checking [10, 11] がある．Regular Model Checking では，与えられた有限 letter-to-letter トランスデューサの推移閉包を effective に計算し，到達可能な集合を求める．しかし一般には推移閉包の計算は決定可能ではないという問題がある．本論文で扱う話と似ているが，我々の提案では letter-to-letter では表現出来ない push と pop の操作を扱う，という点において違いがある．

南出と森は，Conditional PDS を HTML5 構文解析アルゴリズムのサブセットの形式化に適用し，テストの自動生成を行っている [12]．この研究では，効率的に pre* を計算するために Conditional PDS から PDS へ変換を行わず，pre* を直接，先読み付きオートマトンとして構成している．この構成は，重み付き PDS における重みの構造を一般化することで，重み付き PDS の到達可能性解析と考えられることが示されている [13]．今回提案した CTPDS は，HTML5 構文解析アルゴリズムのより広いサブセットの形式化に応用できると考えられる．

Abdulla らが [7] で導入した Timed Pushdown Systems は，計算状況が $Q \times (\Gamma \times \mathbb{N})^*$ となるようにスタックアルファベット Γ を年齢（$\mathbb{N}$）で拡張し，遷移によって全てのシンボルの年齢を増やすことが出来るような PDS だと言える．この設定だけではスタックシンボルの空間（$\Gamma \times \mathbb{N}$）は有限にならないが，実際にはスタックシンボルの年齢に対する条件付けに必要な年齢の上限 n までが必要であり，$n+1$ 以上の数は全て ω とみなすような有限の空間 $\mathbb{N}_{\leq n}$ で十分である．この条件下で，CTPDS のアルファベットを $\Gamma \times \mathbb{N}_{\leq n}$ として考えると $\gamma \in \Gamma$ 毎に年齢を加えることができ，明らかに Timed Pushdown Systems を実現することが出来る．

11 まとめと今後の課題

スタックに対する正則言語を用いた検査と，書換えによるスタックの変換を許した PDS の拡張 CTPDS を提案した．CTPDS を，より一般の計算モデルであるトランスダクション規則を許した TrPDS として形式化し，その位置到達可能性問題が決定可能であることを示した．

CTPDS の検査と変換に関する更なる一般化が考えられる．例えば検査として文脈自由言語を用いることが出来ないかを考えることは自然な発想だが，文脈自由言語は共通部分に関して閉じていないので困難が生じると考える．そこで良い閉包性（closure property）を持つ visibly pushdown languages[14] などを今後考えたい．

一方変換においては，$\Gamma \to \Gamma$ 上の関数ではなく，$\Gamma \to \Gamma^+$ の文字から空でない文字列への関数を考えたい．しかし $\Gamma \to \Gamma^+$ を許すと次のようにして 2-計数機械が摸倣出来る．

$$\text{TrPDS } \mathcal{S} = \langle Q, \{0,1,Z\}, q_I, \{\mathfrak{t}_0, \mathfrak{t}_1, \mathfrak{t}_Z, \alpha, \beta\}, \Delta \rangle$$

configuration が常に $Q \times ((\Gamma \setminus \{Z\})^*\{Z\})$ となるように工夫し，トランスダクションを以下のように定義する．

$$\mathfrak{t}_0 = (0,0)(\Gamma,\Gamma)^* \quad \mathfrak{t}_1 = (1,1)(\Gamma,\Gamma)^* \quad \mathfrak{t}_Z = (Z,Z)(\Gamma,\Gamma)^*$$

$$\alpha(0) = 0, \alpha(1) = 1, \alpha(Z) = 0Z$$

$$\beta(0) = 0, \beta(1) = 1, \beta(Z) = 1Z$$

2-counter machine としてのインクリメントは push で表現し，デクリメントは α, β と pop を組み合わせて表現する．この例からも，一般の準同型写像でさえ到達可能性は決定不能になる．関数ではなく，letter-to-letter なトランスデューサ一般の話を考えることも出来るが，Regular Model Checking と同様の難しさが生じるだろう．

何れにせよ現在の方法では有限の $\mathcal{T}$-*Alg* で扱える必要があるので，トランスダクションの集合を有限の $\mathcal{T}$-*Alg* に代数化出来ないものは，本論文の手法では扱うことが出来ない．また CTPDS へのモデル検査という点では，位置到達可能性問題のみを示した．しかし応用の面からは，より一般的な到達可能性問題の決定可能性と，pre^* 及び post^* の計算の決定可能性を考える必要があり，現在この問題に取り組んでいる．

参考文献

[1] Ahmed Bouajjani and Javier Esparza. Rewriting models of boolean programs. In *Term Rewriting and Applications*, Vol. 4098 of *Lecture Notes in Computer Science*, pp. 136–150. Springer Berlin Heidelberg, 2006.

[2] Alain Finkel, Bernard Willems, and Pierre Wolper. A direct symbolic approach to model checking pushdown systems. In *INFINITY '97*, Vol. 9 of *ENTCS*, pp. 27–37. 1997.

[3] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR '97: Concurrency Theory*, Vol. 1243 of *Lecture Notes in Computer Science*, pp. 135–150. 1997.

[4] Stefan Schwoon. *Model-Checking Pushdown Systems*. Ph.D. Thesis, Technische Universität München, June 2002.

[5] Javier Esparza, Antonín Kučera, and Stefan Schwoon. Model checking LTL with regular valuations for pushdown systems. *Information and Computation*, Vol. 186, No. 2, pp. 355 – 376, 2003.

[6] Xin Li and Mizuhito Ogawa. Conditional weighted pushdown systems and applications. In *Proceedings of the 2010 ACM SIGPLAN workshop on Partial evaluation and program manipulation*, PEPM '10, pp. 141–150. ACM, 2010.

[7] Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. The minimal cost reachability problem in priced timed pushdown systems. In *Proceedings of the 6th international conference on Language and Automata Theory and Applications*, LATA'12, pp. 58–69, Berlin, Heidelberg, 2012. Springer-Verlag.

[8] Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, Vol. 11, No. 4, pp. 481–494, 1964.

[9] Samuel Eilenberg. *Automata, Languages and Machines, volume A*. Academic Press, 1974.

[10] Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili. Regular Model Checking. In *Computer Aided Verification*, Vol. 1855 of *Lecture Notes in Computer Science*, pp. 403–418. Springer Berlin Heidelberg, 2000.

[11] ParoshAziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Mayank Saksena. A Survey of Regular Model Checking. In *CONCUR 2004*, Vol. 3170 of *Lecture Notes in Computer Science*, pp. 35–48. Springer Berlin Heidelberg, 2004.

[12] Yasuhiko Minamide and Shunsuke Mori. Reachability Analysis of the HTML5 Parser Specification and Its Application to Compatibility Testing. In *FM 2012: Formal Methods*, Vol. 7436 of *Lecture Notes in Computer Science*, pp. 293–307. 2012.

[13] Yasuhiko Minamide. Weighted pushdown systems with indexed weight domains. In *Proc. of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2013. to appear.

[14] Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, Vol. 56, No. 3, 2009.

12 付録：証明

補題 1.

$$\mathfrak{u}(w) = G(\mathfrak{u})(w) \quad \mathfrak{t}(w) = F(\mathfrak{t})(w) \quad \mathfrak{u}_2(\mathfrak{u}_1(w)) = (\mathfrak{u}_2 \bullet \mathfrak{u}_1)(w)$$

証明. $\forall \mathfrak{u} \in \mathcal{U}.\mathfrak{u}(w) = G(\mathfrak{u})(w)$ を w に関する帰納法で示す.

base case

$$\begin{aligned} \mathfrak{u}(\lambda) &= \nu(\mathfrak{u}) \\ &= \nu(G(\mathfrak{u})) \quad (G \text{の条件}) \\ &= G(\mathfrak{u})(\lambda) \end{aligned}$$

induction step

$$\begin{aligned} G(\mathfrak{u})(\gamma w) &= \textstyle\bigcup_{\gamma' \in \Gamma} \gamma' \lhd \left(\langle \gamma, \gamma' \rangle^{-1} G(\mathfrak{u})\right)(w) \\ &= \textstyle\bigcup_{\gamma' \in \Gamma} \gamma' \lhd \left(G\left(\langle \gamma, \gamma' \rangle^{-1} \mathfrak{u}\right)\right)(w) && (G \text{の条件}) \\ &= \textstyle\bigcup_{\gamma' \in \Gamma} \gamma' \lhd \left(\langle \gamma, \gamma' \rangle^{-1} \mathfrak{u}\right)(w) && (\text{帰納法の仮定}) \\ &= \mathfrak{u}(\gamma w) && (\text{定義より}) \end{aligned}$$

補題 **2**（**pop**）**.**

$$\|\mathrm{pop}(w)\| = \Gamma^{-1}\|w\|$$

証明. w のサイズに関する場合分けを用いる.

base case1

$$\|\mathrm{pop}(\lambda)\| = \|\varnothing\| = \varnothing = \Gamma^{-1}\{\lambda\} = \Gamma^{-1}\|\lambda\|$$

base case2

$$\begin{aligned}
\Gamma^{-1}\|\langle\gamma,\mathfrak{a}\rangle\,\lambda\| &= \Gamma^{-1}(\mathfrak{a}(\gamma \lhd \|\lambda\|)) \\
&= \Gamma^{-1}(\mathfrak{a}(\gamma)) \\
&= \Gamma^{-1}\left(\textstyle\bigcup_{\gamma'\in\Gamma}\gamma' \lhd \left(\langle\gamma,\gamma'\rangle^{-1}\mathfrak{a}\right)(\lambda)\right) \\
&= \textstyle\bigcup_{\gamma'\in\Gamma}\nu\left(\langle\gamma,\gamma'\rangle^{-1}\mathfrak{a}\right) \\
&= \|\mathrm{pop}(\langle\gamma,\mathfrak{a}\rangle\,\lambda)\|
\end{aligned}$$

induction step

$$\begin{aligned}
& \Gamma^{-1}\|\langle\gamma,\mathfrak{a}\rangle\,\langle\gamma',\mathfrak{b}\rangle\,w\| \\
= \;& \Gamma^{-1}(\mathfrak{a}(\gamma \lhd \|\langle\gamma',\mathfrak{b}\rangle\,w\|)) \\
= \;& \Gamma^{-1}\textstyle\bigcup_{\gamma''\in\Gamma}\left\{\gamma'' \lhd \mathfrak{a}'\|\langle\gamma',\mathfrak{b}\rangle\,w\| \mid \mathfrak{a}' = \langle\gamma,\gamma''\rangle^{-1}\mathfrak{a}\right\} \quad \text{(prefix lemma より)} \\
= \;& \textstyle\bigcup_{\gamma''\in\Gamma}\left\{\mathfrak{a}'\|\langle\gamma',\mathfrak{b}\rangle\,w\| \mid \mathfrak{a}' = \langle\gamma,\gamma''\rangle^{-1}\mathfrak{a}\right\} \\
= \;& \textstyle\bigcup_{\gamma''\in\Gamma}\left\{\mathfrak{a}'(\mathfrak{b}(\gamma' \lhd \|w\|)) \mid \mathfrak{a}' = \langle\gamma,\gamma''\rangle^{-1}\mathfrak{a}\right\} \\
= \;& \textstyle\bigcup_{\gamma''\in\Gamma}\left\{(\mathfrak{a}'\bullet\mathfrak{b})(\gamma' \lhd \|w\|) \mid \mathfrak{a}' = \langle\gamma,\gamma''\rangle^{-1}\mathfrak{a}\right\} \qquad (\bullet\text{に関する補題}) \\
= \;& \textstyle\bigcup_{\gamma''\in\Gamma}\left\{\|\langle\gamma',\mathfrak{a}'\bullet\mathfrak{b}\rangle\,w\| \mid \mathfrak{a}' = \langle\gamma,\gamma''\rangle^{-1}\mathfrak{a}\right\} \\
= \;& \left\|\textstyle\bigcup_{\gamma''\in\Gamma}\left\{\langle\gamma',\mathfrak{a}'\bullet\mathfrak{b}\rangle\,w \mid \mathfrak{a}' = \langle\gamma,\gamma''\rangle^{-1}\mathfrak{a}\right\}\right\| \\
= \;& \|\mathrm{pop}(\langle\gamma,\mathfrak{a}\rangle\,\langle\gamma',\mathfrak{b}\rangle\,w)\|
\end{aligned}$$

補題 (prefix lemma)**.**

$$\mathfrak{t}(\gamma \lhd W) = \bigcup_{\gamma'\in\Gamma}\left\{\gamma' \lhd \mathfrak{t}'(W) \mid \mathfrak{t}' = \langle\gamma,\gamma'\rangle^{-1}\mathfrak{t}\right\}$$

証明**.**

$$\begin{aligned}
\mathfrak{t}(\gamma \lhd W) &= \textstyle\bigcup_{w\in W}\mathfrak{t}(\gamma w) \\
&= \textstyle\bigcup_{w\in W}\left(\bigcup_{\gamma'\in\Gamma}\left\{\gamma' \lhd \mathfrak{t}'(w) \mid \mathfrak{t}' = \langle\gamma,\gamma'\rangle^{-1}\mathfrak{t}\right\}\right) \\
&= \textstyle\bigcup_{\gamma'\in\Gamma}\left\{\bigcup_{w\in W}\gamma' \lhd \mathfrak{t}'(w) \mid \mathfrak{t}' = \langle\gamma,\gamma'\rangle^{-1}\mathfrak{t}\right\} \\
&= \textstyle\bigcup_{\gamma'\in\Gamma}\left\{\gamma' \lhd \mathfrak{t}'(W) \mid \mathfrak{t}' = \langle\gamma,\gamma'\rangle^{-1}\mathfrak{t}\right\}
\end{aligned}$$

定理 **1**（強双摸倣性） $X \sim Y$ は $\twoheadrightarrow$ に関して強双摸倣性を満たす.

$$\begin{array}{ccc}
X & \sim & Y \\
{\scriptstyle t}\big\downarrow & & \big\downarrow{\scriptstyle t} \\
X' & \sim & Y'
\end{array}$$

証明. 遷移規則を場合分けして証明する. 全て同じ形で証明出来るので, pop の場合だけをここでは証明する.

$t = \mathbf{Pop} = \langle p, q\rangle \quad X' = \|Y'\|$ を示す.

$$
\begin{aligned}
\|Y'\| &= \|\{\langle q, x\rangle \mid \langle p, w\rangle \in Y, x \in \mathrm{pop}(w)\}\| \\
&= \{\langle q, x\rangle \mid \langle p, w\rangle \in Y, x \in \|\mathrm{pop}(w)\|\} \\
&= \{\langle q, x\rangle \mid \langle p, w\rangle \in Y, x \in \Gamma^{-1}\,\|w\|\} \quad \text{(補題 2)} \\
&= \{\langle q, x\rangle \mid \langle p, w\rangle \in \|Y\|, x \in \Gamma^{-1} w\} \\
&= \{\langle q, x\rangle \mid \langle p, w\rangle \in X, x \in \Gamma^{-1} w\} \quad (X \sim Y) \\
&= X'
\end{aligned}
$$

系 1. 演算 $\bullet$ は確かに $\mathcal{C} \times \mathcal{M}$ 上で閉じている．

証明.

$$\langle c_2, m_2\rangle \bullet \langle c_1, m_1\rangle = \left\langle m_1^{-1}(c_2) \cap c_1, m_2 \circ m_1\right\rangle$$

左項 $m_1^{-1}(c_2) \cap c_1$ については，定理 2 より $m_1^{-1}(c_2) \in \mathcal{C}$ であることから $m_1^{-1}(c_2) \cap c_1 \in \mathcal{C}$．右項については $m_2 \circ m_1 \in \mathcal{M}$ が明らか．従って $\mathcal{C} \times \mathcal{M}$ 上について確かに閉じている．

定理 3.

$$\gamma^{-1}(\mathcal{C}) \subseteq \mathcal{C}$$

証明. 任意の $c \in \mathcal{C}$ について，$\gamma^{-1}(c) \in \mathcal{C}$ であることを証明すれば良い．$c = m_1^{-1}(d_1^1) \cap \cdots \cap m_n^{-1}(d_1^1)$ と書ける．この時，

$$
\begin{aligned}
&\gamma^{-1}\left(m_1^{-1}(d_1^1) \cap \cdots \cap m_n^{-1}(d_n^1)\right) &&= \\
&\gamma^{-1}(m_1^{-1}(d_1^1)) \cap \cdots \cap \gamma^{-1}(m_n^{-1}(d_n^1)) &&= \\
&m_1^{-1}(m_1(\gamma)^{-1} d_1^1) \cap \cdots \cap m_n^{-1}(m_n(\gamma)^{-1} d_n^1)
\end{aligned}
$$

となる．$\mathcal{D}_1$ は商について閉じているので，$m_i(\gamma)^{-1} d_i^1 \in \mathcal{D}_1$ となり，$m_i^{-1}(m_i(\gamma)^{-1} d_i^1) \in \mathcal{D}_2$ が成立する．よって $\gamma^{-1}(c) \in \mathcal{C}$ が示せた．

系 2. 演算 $\langle \gamma, \gamma'\rangle^{-1}$ は確かに $\mathcal{C} \times \mathcal{M}$ 上で閉じている．

証明.

$$\langle \gamma, \gamma'\rangle^{-1} \langle c, m\rangle = \begin{cases} \left\langle \gamma^{-1} c, \langle \gamma, \gamma'\rangle^{-1} m\right\rangle & \gamma' = m(\gamma) \\ \langle \varnothing, \mathbb{1}\rangle & \gamma' \neq m(\gamma) \end{cases}$$

先に $\gamma' \neq m(\gamma)$ の場合を考えるが，これは明らか．次に $\gamma' = m(\gamma)$ の場合を考える．$\gamma^{-1} c \in \mathcal{C}$ は定理 3 より明らか．$\gamma' = m(\gamma)$ より $\langle \gamma, \gamma'\rangle^{-1} m = m$ であるから $\mathcal{M}$ に閉じている．

補題 4.

$$\forall c \in \mathcal{C}, m \in \mathcal{M}\ .\ \widetilde{c} \circ m = m \circ \widetilde{m^{-1}(c)}$$

証明.

$$
\begin{aligned}
\widetilde{c} \circ m &= \{\langle w, w'\rangle \mid \langle w, w'\rangle \in m, w' \in c\} \\
&= \{\langle w, w'\rangle \mid \langle w, w'\rangle \in m, m(w) \in c\} \\
&= \{\langle w, w'\rangle \mid \langle w, w'\rangle \in m, w \in m^{-1}(c)\} \\
&= \{\langle w, w'\rangle \mid \langle w, w'\rangle \in m, \langle w, w\rangle \in \widetilde{m^{-1}(c)}\} \\
&= m \circ \widetilde{m^{-1}(c)}
\end{aligned}
$$

補題 **5**

$$\forall t_1, t_2 \in \mathbf{Transd}_\Gamma.\ \langle a, c\rangle^{-1}(t_2 \circ t_1) = \bigcup_{b\in\Gamma}\left(\langle b, c\rangle^{-1} t_2 \circ \langle a, b\rangle^{-1} t_1\right)$$

証明**.**

$$\begin{aligned}
& \langle w_1, w_2\rangle \in \langle a, c\rangle^{-1}(t_2 \circ t_1) \\
\iff\ & \langle aw_1, cw_2\rangle \in t_2 \circ t_1 \\
\iff\ & \langle aw_1, bx\rangle \in t_1 \text{かつ} \langle bx, cw_2\rangle \in t_2 \text{ とする } b \text{ と } x \text{ が存在} \quad (t_1 \text{と } t_2 \text{は length preserving}) \\
\iff\ & \langle w_1, x\rangle \in \langle a, b\rangle^{-1} t_1 \text{かつ} \langle x, w_2\rangle \in \langle b, c\rangle^{-1} t_2 \text{ とする } b \text{ と } x \text{ が存在} \\
\iff\ & \langle w_1, w_2\rangle \in \textstyle\bigcup_{b\in\Gamma}\left(\langle b, c\rangle^{-1} t_2 \circ \langle a, b\rangle^{-1} t_1\right)
\end{aligned}$$