# Pushdown Systems with Stack Manipulation

Yuya Uezato    Yasuhiko Minamide

University of Tsukuba

ATVA 2013

# Pushdown System(PDS)

**PDS** $\mathcal{P}$ is a structure $(Q, \Gamma, \Delta)$:

- $Q$ is a finite set of control locations,
- $\Gamma$ is a finite set of stack alphabets,
- $\Delta \subseteq Q \times (\Gamma \times \Gamma^*) \times Q$ is a finite set of transition rules.

◇ A pair of control location and stack is called *configuration*,

◇ $\mathit{Conf}(\mathcal{P}) = Q \times \Gamma^*$ are *configurations* of $\mathcal{P}$.

◇ Transition relation $\to \subseteq \mathit{Conf}(\mathcal{P}) \times \mathit{Conf}(\mathcal{P})$ :

$$\left\langle p, \begin{array}{|c|}\gamma \\ \hline v \\ \hline \end{array} \right\rangle \to \left\langle q, \begin{array}{|c|} w \\ \hline v \\ \hline \end{array} \right\rangle \quad \mathit{if}\ \ p \xhookrightarrow{\gamma/w} q \in \Delta$$

# Reachability problem of PDS

**Reachability problem :**
For a PDS $\mathcal{P}$, configurations $\mathbf{c}_{initial}$ and $\mathbf{c}_{final}$,
is $\mathbf{c}_{final}$ reachable from $\mathbf{c}_{initial}$ in $\mathcal{P}$ ?

Decidability[Finkel et al., 1997, Bouajjani et al., 1997]

*The reachability problem of PDS is decidable.*

- The reachability problem of PDS is not only fundamental problem, but also useful for software model checking.
- Extended pushdown systems that preserve decidability of reachability problems have been introduced for reasoning more complex systems.

# Conditional Pushdown System [Esparza et al., 2003]

- Extending rules by regular expressions : $p \xhookrightarrow[R]{\gamma/w} q$,
- $R$ inspects the entire stack except the top.

Example: $p \xhookrightarrow[1^*]{1/\varepsilon} q$

- $p \begin{array}{|c|}1\\\hline 1\\\hline 1\\\hline\end{array} \to q \begin{array}{|c|}1\\\hline 1\\\hline\end{array} \quad (\because \begin{array}{|c|}1\\\hline 1\\\hline\end{array} \in 1^*)$
- $p \begin{array}{|c|}1\\\hline 2\\\hline\end{array} \to$ ✗ $\quad (\because \begin{array}{|c|}2\\\hline\end{array} \notin 1^*)$

Every conditional PDS is simulated by ordinary PDS and reachability problem of conditional PDS is decidable.

# Discrete Timed PDS[Abdulla et al., 2012b]

Timed PDS can modify the entire stack.

- $(\gamma, \alpha) \in \Gamma \times \mathbb{N}^{\omega}_{\leq m}$ : $\alpha$ is the age of the symbol $\gamma$

$$\mathbb{N}^{\omega}_{\leq m} = \{n \in \mathbb{N} \mid n \leq m\} \cup \{\omega\}$$

- *delay* transitions increment the age of every symbol

$$\begin{array}{|c|}(b,1)\\\hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}(a,0)\\\hline(b,1)\\\hline\end{array} \xrightarrow{delay} \begin{array}{|c|}(a,1)\\\hline(b,2)\\\hline\end{array} \xrightarrow{delay} \begin{array}{|c|}(a,2)\\\hline(b,3)\\\hline\end{array} \xrightarrow{\text{pop}} \begin{array}{|c|}(b,3)\\\hline\end{array}$$

Every timed PDS is simulated by ordinary PDS and reachability problem of timed PDS is decidable.

# Contribution

1. We introduce an extended PDS, *TrPDS*
   - TrPDS modifies the entire stack by using *transduction*
   - Conditional PDS and Timed PDS are simple instances of TrPDS
2. We show that *finite* TrPDS can be simulated by PDS
   - TrPDS is finite if the closure of transductions appearing in the transitions is finite.
   - as a corollary, we show the reachability problem of *finite* TrPDS is decidable.

# Outline

1. TrPDS : Extending PDS with transductions

2. Simulating TrPDS by ordinary PDS

3. Decidability of reachability problem of finite TrPDS

# Transducer and Transduction

## (Letter-to-letter) Transducer

states, transitions, accepting states

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

alphabets

initial state

# Transducer and Transduction

## (Letter-to-letter) Transducer

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

states

transitions

accepting states

alphabets

initial state

- $p \xrightarrow{\sigma \mid \sigma'} q \in E \subseteq S \times (\Gamma \times \Gamma) \times S$,
- The language of $\mathfrak{t}$ is $\mathcal{L}(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$,
- Each transducer $\mathfrak{t}$ defines function $\mathfrak{t} \in \Gamma^* \to \mathcal{P}(\Gamma^*)$.

# Transducer and Transduction

## (Letter-to-letter) Transducer

transitions

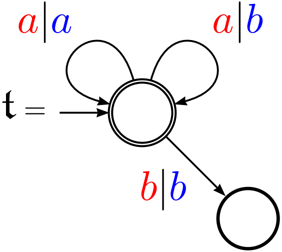
states    accepting states

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

alphabets

initial state

- $p \xrightarrow{\sigma \mid \sigma'} q \in E \subseteq S \times (\Gamma \times \Gamma) \times S$,
- The language of $\mathfrak{t}$ is $\mathcal{L}(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$,
- Each transducer $\mathfrak{t}$ defines function $\mathfrak{t} \in \Gamma^* \to \mathcal{P}(\Gamma^*)$.




$$
\begin{aligned}
a\mathfrak{t} &= \{a, b\} \\
aa\mathfrak{t} &= \{aa, ab, ba, bb\} \\
b\mathfrak{t} &= \varnothing \\
&\vdots
\end{aligned}
$$

# Transducer and Transduction

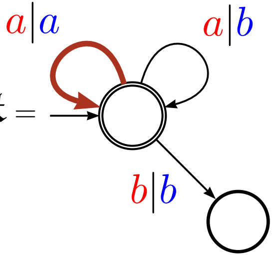
## (Letter-to-letter) Transducer

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

states, transitions, accepting states, alphabets, initial state

- $p \xrightarrow{\sigma \mid \sigma'} q \in E \subseteq S \times (\Gamma \times \Gamma) \times S$,
- The language of $\mathfrak{t}$ is $\mathcal{L}(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$,
- Each transducer $\mathfrak{t}$ defines function $\mathfrak{t} \in \Gamma^* \to \mathcal{P}(\Gamma^*)$.

$\mathfrak{t} =$




$$
\begin{aligned}
\boxed{a}\,\mathfrak{t} &= \{\boxed{a}, b\} \\
aa\mathfrak{t} &= \{aa, ab, ba, bb\} \\
b\mathfrak{t} &= \varnothing \\
&\vdots
\end{aligned}
$$

# Transducer and Transduction

states, transitions, accepting states

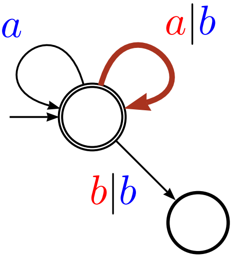
## (Letter-to-letter) Transducer

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

alphabets, initial state

- $p \xrightarrow{\sigma \mid \sigma'} q \in E \subseteq S \times (\Gamma \times \Gamma) \times S$,
- The language of $\mathfrak{t}$ is $\mathcal{L}(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$,
- Each transducer $\mathfrak{t}$ defines function $\mathfrak{t} \in \Gamma^* \to \mathcal{P}(\Gamma^*)$.

$\mathfrak{t} =$




$$
\begin{aligned}
\boxed{a}\,\mathfrak{t} &= \{a, \boxed{b}\} \\
aa\mathfrak{t} &= \{aa, ab, ba, bb\} \\
b\mathfrak{t} &= \varnothing \\
&\vdots
\end{aligned}
$$

# Transducer and Transduction

transitions

states

accepting states

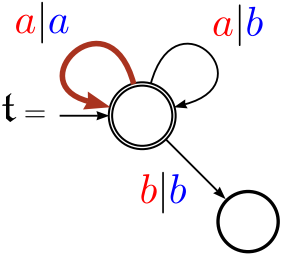
## (Letter-to-letter) Transducer

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

alphabets

initial state

- $p \xrightarrow{\sigma \mid \sigma'} q \in E \subseteq S \times (\Gamma \times \Gamma) \times S$,
- The language of $\mathfrak{t}$ is $\mathcal{L}(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$,
- Each transducer $\mathfrak{t}$ defines function $\mathfrak{t} \in \Gamma^* \to \mathcal{P}(\Gamma^*)$.




$$
\begin{aligned}
a\mathfrak{t} &= \{a, b\} \\
\boxed{a}\,a\mathfrak{t} &= \{aa, \boxed{a}\,b, ba, bb\} \\
b\mathfrak{t} &= \emptyset \\
&\vdots
\end{aligned}
$$

# Transducer and Transduction

## (Letter-to-letter) Transducer

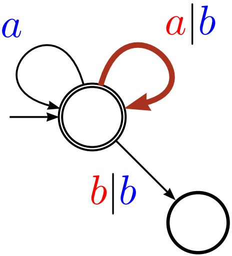
states, transitions, accepting states

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

alphabets, initial state

- $p \xrightarrow{\sigma \mid \sigma'} q \in E \subseteq S \times (\Gamma \times \Gamma) \times S$,
- The language of $\mathfrak{t}$ is $\mathcal{L}(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$,
- Each transducer $\mathfrak{t}$ defines function $\mathfrak{t} \in \Gamma^* \to \mathcal{P}(\Gamma^*)$.

$\mathfrak{t} =$




$$
\begin{aligned}
a\mathfrak{t} &= \{a, b\} \\
a\boxed{a}\mathfrak{t} &= \{aa, a\boxed{b}, ba, bb\} \\
b\mathfrak{t} &= \varnothing \\
&\vdots
\end{aligned}
$$

# Transducer and Transduction

transitions

states  accepting states

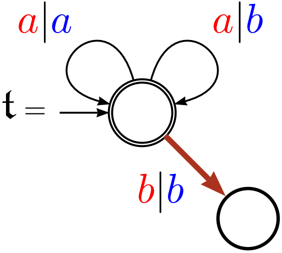
## (Letter-to-letter) Transducer

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

alphabets

initial state

- $p \xrightarrow{\sigma \mid \sigma'} q \in E \subseteq S \times (\Gamma \times \Gamma) \times S$,
- The language of $\mathfrak{t}$ is $\mathcal{L}(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$,
- Each transducer $\mathfrak{t}$ defines function $\mathfrak{t} \in \Gamma^* \to \mathcal{P}(\Gamma^*)$.




$$\begin{aligned} a\mathfrak{t} &= \{a, b\} \\ aa\mathfrak{t} &= \{aa, ab, ba, bb\} \\ \boxed{b}\,\mathfrak{t} &= \boxed{\varnothing} \\ &\vdots \end{aligned}$$

# Transducer and Transduction

transitions

states accepting states

## (Letter-to-letter) Transducer

Finite automaton over $\underbrace{\Gamma}_{\text{INPUT}} \times \underbrace{\Gamma}_{\text{OUTPUT}}$, $\mathfrak{t} = (S, \Gamma, E, i, F)$.

alphabets

initial state

- $p \xrightarrow{\sigma \mid \sigma'} q \in E \subseteq S \times (\Gamma \times \Gamma) \times S$,
- The language of $\mathfrak{t}$ is $\mathcal{L}(\mathfrak{t}) \subseteq (\Gamma \times \Gamma)^*$,
- Each transducer $\mathfrak{t}$ defines function $\mathfrak{t} \in \Gamma^* \to \mathcal{P}(\Gamma^*)$.

## (Length-preserving) Transduction

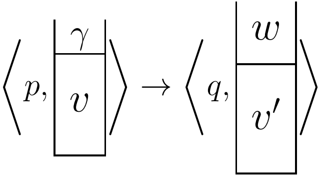
A function $\mathfrak{t}$ is a transduction if $\mathfrak{t}$ can be realized by a transducer.

# PDS with stack manipulation(transductions)

TrPDS $\mathcal{P}$ is a structure $(Q, \Gamma, T, \Delta)$,

- $T$ is a finite set of transductions,
- $\Delta \subseteq Q \times (\Gamma \times \Gamma^* \times T) \times Q$ is a finite set of transition rules.

Transition rule $p \xhookrightarrow{\gamma/w \mid \mathfrak{t}} q \in \Delta$ derives a transition




*if* $v' \in v\mathfrak{t}$

# PDS with stack manipulation(transductions)

TrPDS $\mathcal{P}$ is a structure $(Q, \Gamma, T, \Delta)$,

- $T$ is a finite set of transductions,
- $\Delta \subseteq Q \times (\Gamma \times \Gamma^* \times T) \times Q$ is a finite set of transition rules.

Transition rule $p \xhookrightarrow{\gamma / w \mid \mathfrak{t}} q \in \Delta$ derives a transition

$$\left\langle p, \begin{array}{|c|} \gamma \\ \hline v \\ \hline \end{array} \right\rangle \rightarrow \left\langle q, \begin{array}{|c|} w \\ \hline v' \\ \hline \end{array} \right\rangle \quad \textit{if } v' \in v\mathfrak{t}$$

Example : $p \xhookrightarrow{c/\varepsilon \mid \mathfrak{t}} q \quad (aa\mathfrak{t} = \{aa, ab, ba, bb\}, ab\mathfrak{t} = \varnothing)$

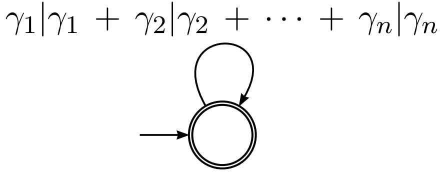
$$p \begin{array}{|c|} c \\ \hline a \\ \hline a \\ \hline \end{array} \rightarrow \left\{ q \begin{array}{|c|} a \\ \hline a \\ \hline \end{array}, q \begin{array}{|c|} a \\ \hline b \\ \hline \end{array}, q \begin{array}{|c|} b \\ \hline a \\ \hline \end{array}, q \begin{array}{|c|} b \\ \hline b \\ \hline \end{array} \right\}, \quad p \begin{array}{|c|} c \\ \hline a \\ \hline b \\ \hline \end{array} \rightarrow \varnothing$$

# Example : Ordinary PDS

Ordinary PDS $(Q, \Gamma, \Delta)$,

TrPDS $(Q, \Gamma, \{\mathbb{1}\}, \Delta')$,

- $\mathbb{1}$ is the identity function.

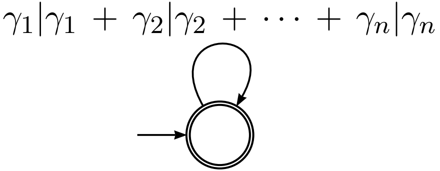


# Example : Ordinary PDS

Ordinary PDS $(Q, \Gamma, \Delta)$,

TrPDS $(Q, \Gamma, \{\mathbb{1}\}, \Delta')$,

- $\mathbb{1}$ is the identity function.




$p \xhookrightarrow{\gamma/w} q \in \Delta$

derives $\left\langle p, \begin{array}{|c|} \gamma \\ \hline v \\ \hline \end{array} \right\rangle \to \left\langle q, \begin{array}{|c|} w \\ \hline v \\ \hline \end{array} \right\rangle$

$p \xhookrightarrow{\gamma/w \,|\, \mathbb{1}} q \in \Delta'$

derives $\left\langle p, \begin{array}{|c|} \gamma \\ \hline v \\ \hline \end{array} \right\rangle \to \left\langle q, \begin{array}{|c|} w \\ \hline v \\ \hline \end{array} \right\rangle$

# Example : Conditional PDS

Conditional PDS $(Q, \Gamma, \mathcal{R}, \Delta)$,

- $\mathcal{R}$ is a finite set of regular expressions over $\Gamma$

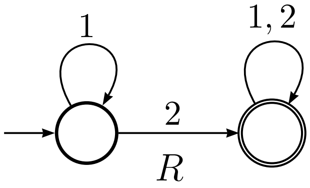
TrPDS $(Q, \Gamma, \widetilde{\mathcal{R}}, \Delta')$,

- $\widetilde{\mathcal{R}}$ is copycat version of $\mathcal{R}$

# Example : Conditional PDS

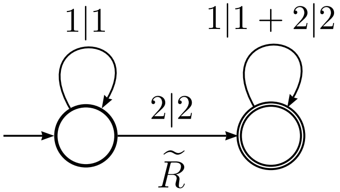
Conditional PDS $(Q, \Gamma, \mathcal{R}, \Delta)$,

- $\mathcal{R}$ is a finite set of regular expressions over $\Gamma$




TrPDS $(Q, \Gamma, \widetilde{\mathcal{R}}, \Delta')$,

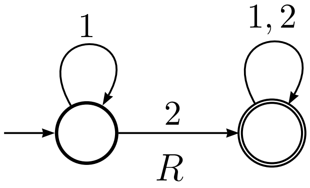
- $\widetilde{\mathcal{R}}$ is copycat version of $\mathcal{R}$



# Example : Conditional PDS

Conditional PDS $(Q, \Gamma, \mathcal{R}, \Delta)$,

- $\mathcal{R}$ is a finite set of regular expressions over $\Gamma$

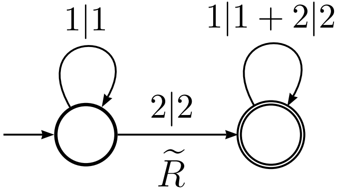



$p \xhookrightarrow[R]{\gamma/w} q \in \Delta$ and $\underline{\underline{v \in R}}$

derives $\left\langle p, \begin{array}{|c|} \gamma \\ \hline v \\ \hline \end{array} \right\rangle \to \left\langle q, \begin{array}{|c|} w \\ \hline v \\ \hline \end{array} \right\rangle$.

TrPDS $(Q, \Gamma, \widetilde{\mathcal{R}}, \Delta')$,

- $\widetilde{\mathcal{R}}$ is copycat version of $\mathcal{R}$




$p \xhookrightarrow{\gamma/w \mid \widetilde{R}} q \in \Delta'$ and $\underline{\underline{v \in R}}$

derives $\left\langle p, \begin{array}{|c|} \gamma \\ \hline v \\ \hline \end{array} \right\rangle \to \left\langle q, \begin{array}{|c|} w \\ \hline v \\ \hline \end{array} \right\rangle$.

# Example : Timed PDS

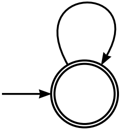
- Every timed PDS is an instance of TrPDS,
  - because *delay* : $\mathbb{N}^{\omega}_{\leq m} \to \mathbb{N}^{\omega}_{\leq m}$ is a transduction.

# Example : Timed PDS

- Every timed PDS is an instance of TrPDS,
  - because *delay* : $\mathbb{N}^{\omega}_{\leq m} \to \mathbb{N}^{\omega}_{\leq m}$ is a transduction.
- The transduction *delay* is realized by a transducer $\mathbb{D}$.

$$delay(n) = \begin{cases} n+1 & \textit{if}\ 0 \leq n \leq (m-1) \\ \omega & \textit{if}\ n = m \text{ or } n = \omega \end{cases}$$

$$0|1 \ + \ 1|2 \ + \ \cdots \ + \ (m-1)|m \ + \ m|\omega \ + \ \omega|\omega$$



transducer $\mathbb{D}$

# TrPDS is very powerful

## Minsky two-counter machine

- equips two counters, $c_0$ and $c_1$, of natural number.
- For both counters, we can increment, decrement, zero-test.

$\diamond$ We encode two-counter machine into TrPDS.

# TrPDS is very powerful

## Minsky two-counter machine

- equips two counters, $c_0$ and $c_1$, of natural number.
- For both counters, we can increment, decrement, zero-test.

◇ We encode two-counter machine into TrPDS.

Idea the value of $c_0$ is encoded as the number of alphabet $0$ in TrPDS. Using padding alphabet $\Box$.

# TrPDS is very powerful

## Minsky two-counter machine

- equips two counters, $c_0$ and $c_1$, of natural number.
- For both counters, we can increment, decrement, zero-test.

◇ We encode two-counter machine into TrPDS.

Idea the value of $c_0$ is encoded as the number of alphabet $0$ in TrPDS. Using padding alphabet $\square$.

$c_0{+}{+}$ is encoded as $p \xhookrightarrow{\gamma/\gamma\,0 \mid \mathbb{1}} q$

$c_0{-}{-}$ is encoded as $p \xhookrightarrow{\gamma/\gamma \quad \mid \mathfrak{d}_0} q$

$c_0 \stackrel{?}{=} 0$ is encoded as $p \xhookrightarrow{\gamma/\gamma \quad \mid \{\square, 1\}^*} q$

# TrPDS is very powerful

## Minsky two-counter machine

- equips two counters, $c_0$ and $c_1$, of natural number.
- For both counters, we can increment, decrement, zero-test.

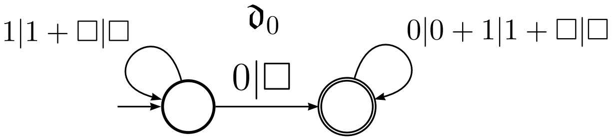
◇ We encode two-counter machine into TrPDS.

**Idea** the value of $c_0$ is encoded as the number of alphabet $0$ in TrPDS. Using padding alphabet $\square$.

$c_0{+}{+}$ is encoded as $p \xhookrightarrow{\gamma/\gamma\, 0 \;\mid\; \mathbb{1}} q$

$c_0{-}{-}$ is encoded as $p \xhookrightarrow{\gamma/\gamma \quad \mid\; \mathfrak{d}_0} q$

$c_0 \overset{?}{=} 0$ is encoded as $p \xhookrightarrow{\gamma/\gamma \quad \mid\; \{\square, 1\}^*} q$



# Main theorem : sufficient condition for simulation

## Finite TrPDS

TrPDS $\mathcal{P} = (Q, \Gamma, T, \Delta)$ is finite $\iff$
the closure $\langle T \rangle$ generated by $\left(T, ;, \langle \cdot, \cdot \rangle^{-1}\right)$ is finite.

$$\mathfrak{t}_1 ; \mathfrak{t}_2 = \{\langle a, c \rangle \mid \langle a, b \rangle \in \mathfrak{t}_1, \langle b, c \rangle \in \mathfrak{t}_2\}$$
$$\langle \gamma, \gamma' \rangle^{-1} \mathfrak{t} = \{\langle w, w' \rangle \mid \langle \gamma w, \gamma' w \rangle \in \mathfrak{t}\}$$

# Main theorem : sufficient condition for simulation

## Finite TrPDS

TrPDS $\mathcal{P} = (Q, \Gamma, T, \Delta)$ is finite $\iff$
the closure $\langle T \rangle$ generated by $\left(T, ;, \langle \cdot, \cdot \rangle^{-1}\right)$ is finite.

$$\mathfrak{t}_1 ; \mathfrak{t}_2 = \{\langle a, c \rangle \mid \langle a, b \rangle \in \mathfrak{t}_1, \langle b, c \rangle \in \mathfrak{t}_2\}$$
$$\langle \gamma, \gamma' \rangle^{-1} \mathfrak{t} = \{\langle w, w' \rangle \mid \langle \gamma w, \gamma' w \rangle \in \mathfrak{t}\}$$

$\langle T \rangle$ is inductively defined as follows:

- $\forall \mathfrak{t} \in T, \mathfrak{t} \in \langle T \rangle$
- $\forall \mathfrak{t}_1, \mathfrak{t}_2 \in \langle T \rangle, \mathfrak{t}_1 ; \mathfrak{t}_2 \in \langle T \rangle$
- $\forall \mathfrak{t} \in \langle T \rangle, \forall \gamma, \gamma' \in \Gamma, \langle \gamma, \gamma' \rangle^{-1} \mathfrak{t} \in \langle T \rangle$

# Main theorem : sufficient condition for simulation

## Finite TrPDS

TrPDS $\mathcal{P} = (Q, \Gamma, T, \Delta)$ is finite $\iff$
the closure $\langle T \rangle$ generated by $\left(T, ;, \langle \cdot, \cdot \rangle^{-1}\right)$ is finite.

$$\mathfrak{t}_1 ; \mathfrak{t}_2 = \{\langle a, c \rangle \mid \langle a, b \rangle \in \mathfrak{t}_1, \langle b, c \rangle \in \mathfrak{t}_2\}$$
$$\langle \gamma, \gamma' \rangle^{-1} \mathfrak{t} = \{\langle w, w' \rangle \mid \langle \gamma w, \gamma' w \rangle \in \mathfrak{t}\}$$

## Main theorem

If TrPDS $\mathcal{P}$ is finite, then $\mathcal{P}$ is simulated by ordinary PDS.

## Fact (*Please see our paper*)

- Conditional and Timed PDS are finite TrPDS.
- CTPDS which equips inspecting(by regular expression)-transition and delay-transition is finite TrPDS.

# Overview

1. TrPDS : Extending PDS with transductions

2. Simulating TrPDS by ordinary PDS

3. Decidability of reachability problem of finite TrPDS

# Simulating Timed PDS [Abdulla et al., 2012b]*

◇ The key idea is accumulating effects of delay as elements of the stack.

$\underline{|\,(b, 1)\,|}$

$\rightsquigarrow$ simulated by

$\underline{|\,\langle b, 1\rangle\,|}$

- $(\gamma, \alpha)$ : $\alpha$ is the age of the symbol $\gamma$.
- $\langle \gamma, \beta \rangle$ : $\beta$ is the count of accumulated effects.
    - If $\langle \gamma, \beta \rangle$ is in the top of stack, $\beta$ is the *real* age.

---

* **The minimal cost reachability problem in priced timed pushdown systems** P.A.Abdulla , M.F.Atig , J.Stenmen *LATA'12*

# Simulating Timed PDS [Abdulla et al., 2012b]$^\star$

◇ The key idea is accumulating effects of delay as elements of the stack.

$$\begin{array}{|c|}(b,1)\\ \hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}(a,0)\\ \hline (b,1)\\ \hline\end{array}$$

$\rightsquigarrow$ simulated by

$$\begin{array}{|c|}\langle b,1\rangle\\ \hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}\langle a,0\rangle\\ \hline \langle b,1\rangle\\ \hline\end{array}$$

- $(\gamma, \alpha)$ : $\alpha$ is the age of the symbol $\gamma$.
- $\langle \gamma, \beta \rangle$ : $\beta$ is the count of accumulated effects.
  - If $\langle \gamma, \beta \rangle$ is in the top of stack, $\beta$ is the *real* age.

---

$\star$ **The minimal cost reachability problem in priced timed pushdown systems** P.A.Abdulla , M.F.Atig , J.Stenmen *LATA'12*

# Simulating Timed PDS [Abdulla et al., 2012b]$^\star$

- The key idea is accumulating effects of delay as elements of the stack.

$$\begin{array}{|c|}(b,1)\\ \hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}(a,0)\\ \hline (b,1)\\ \hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}(a,1)\\ \hline (b,2)\\ \hline\end{array}$$

$\rightsquigarrow$ simulated by

$$\begin{array}{|c|}\langle b,1\rangle\\ \hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}\langle a,0\rangle\\ \hline \langle b,1\rangle\\ \hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}\langle a,1\rangle\\ \hline \langle b,1\rangle\\ \hline\end{array}$$

- $(\gamma,\alpha)$ : $\alpha$ is the age of the symbol $\gamma$.
- $\langle\gamma,\beta\rangle$ : $\beta$ is the count of accumulated effects.
    - If $\langle\gamma,\beta\rangle$ is in the top of stack, $\beta$ is the *real* age.

---

$\star$ **The minimal cost reachability problem in priced timed pushdown systems** P.A.Abdulla , M.F.Atig , J.Stenmen *LATA'12*

# Simulating Timed PDS [Abdulla et al., 2012b]⋆

◇ The key idea is accumulating effects of delay as elements of the stack.

$$\begin{array}{|c|} (b,1) \\ \hline \end{array} \xrightarrow{\text{push}} \begin{array}{|c|} (a,0) \\ \hline (b,1) \\ \hline \end{array} \xrightarrow{\text{delay}} \begin{array}{|c|} (a,1) \\ \hline (b,2) \\ \hline \end{array} \xrightarrow{\text{delay}} \begin{array}{|c|} (a,2) \\ \hline (b,3) \\ \hline \end{array}$$

$\rightsquigarrow$ simulated by

$$\begin{array}{|c|} \langle b,1\rangle \\ \hline \end{array} \xrightarrow{\text{push}} \begin{array}{|c|} \langle a,0\rangle \\ \hline \langle b,1\rangle \\ \hline \end{array} \xrightarrow{\text{delay}} \begin{array}{|c|} \langle a,1\rangle \\ \hline \langle b,1\rangle \\ \hline \end{array} \xrightarrow{\text{delay}} \begin{array}{|c|} \langle a,2\rangle \\ \hline \langle b,1\rangle \\ \hline \end{array}$$

- $(\gamma, \alpha)$ : $\alpha$ is the age of the symbol $\gamma$.
- $\langle \gamma, \beta \rangle$ : $\beta$ is the count of accumulated effects.
  - If $\langle \gamma, \beta \rangle$ is in the top of stack, $\beta$ is the *real* age.

---

⋆ **The minimal cost reachability problem in priced timed pushdown systems** P.A.Abdulla , M.F.Atig , J.Stenmen *LATA'12*

# Simulating Timed PDS [Abdulla et al., 2012b]⋆

◇ The key idea is accumulating effects of delay as elements of the stack.

$$\begin{array}{|c|}(b,1)\\\hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}(a,0)\\\hline (b,1)\\\hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}(a,1)\\\hline (b,2)\\\hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}(a,2)\\\hline (b,3)\\\hline\end{array} \xrightarrow{\text{pop}} \begin{array}{|c|}(b,3)\\\hline\end{array}$$

$$\rightsquigarrow \text{simulated by}$$

$$\begin{array}{|c|}\langle b,1\rangle\\\hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}\langle a,0\rangle\\\hline \langle b,1\rangle\\\hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}\langle a,1\rangle\\\hline \langle b,1\rangle\\\hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}\langle a,2\rangle\\\hline \langle b,1\rangle\\\hline\end{array} \xrightarrow{\text{pop}} \begin{array}{|c|}\langle b,1+2\rangle\\\hline\end{array}$$

- $(\gamma,\alpha)$ : $\alpha$ is the age of the symbol $\gamma$.
- $\langle\gamma,\beta\rangle$ : $\beta$ is the count of accumulated effects.
  - If $\langle\gamma,\beta\rangle$ is in the top of stack, $\beta$ is the *real* age.

---

⋆ **The minimal cost reachability problem in priced timed pushdown systems** P.A.Abdulla , M.F.Atig , J.Stenmen *LATA'12*

# Simulating Timed PDS [Abdulla et al., 2012b]*

◇ The key idea is accumulating effects of delay as elements of the stack.

$$\begin{array}{|c|}\hline (b,1) \\ \hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}\hline (a,0) \\ \hline (b,1) \\ \hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}\hline (a,1) \\ \hline (b,2) \\ \hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}\hline (a,2) \\ \hline (b,3) \\ \hline\end{array} \xrightarrow{\text{pop}} \begin{array}{|c|}\hline (b,3) \\ \hline\end{array}$$

$$\rightsquigarrow \text{simulated by}$$

$$\begin{array}{|c|}\hline \langle b,1\rangle \\ \hline\end{array} \xrightarrow{\text{push}} \begin{array}{|c|}\hline \langle a,0\rangle \\ \hline \langle b,1\rangle \\ \hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}\hline \langle a,1\rangle \\ \hline \langle b,1\rangle \\ \hline\end{array} \xrightarrow{\text{delay}} \begin{array}{|c|}\hline \langle a,2\rangle \\ \hline \langle b,1\rangle \\ \hline\end{array} \xrightarrow{\text{pop}} \begin{array}{|c|}\hline \langle b,1+2\rangle \\ \hline\end{array}$$

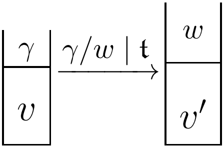
- $(\gamma,\alpha)$ : $\alpha$ is the age of the symbol $\gamma$.
- $\langle\gamma,\beta\rangle$ : $\beta$ is the count of accumulated effects.
  - If $\langle\gamma,\beta\rangle$ is in the top of stack, $\beta$ is the *real* age.
  - When $\langle\gamma,\beta\rangle$ gets popped, $\beta$ should be added to the immediately lower element.

---

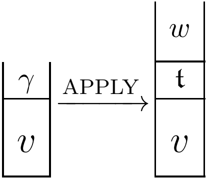
$\star$ **The minimal cost reachability problem in priced timed pushdown systems** P.A.Abdulla , M.F.Atig , J.Stenmen *LATA'12*

# Simulating TrPDS by ordinary PDS

Simulated TrPDS $(Q, \Gamma, T, \Delta)$

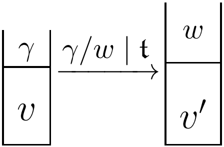



Simulating PDS $(Q, \Gamma \cup \langle T \rangle, \Delta')$



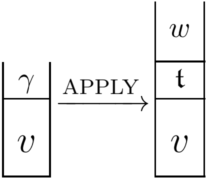
# Simulating TrPDS by ordinary PDS

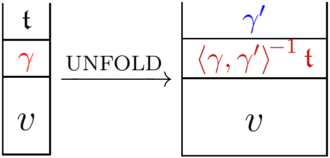
Simulated TrPDS $(Q, \Gamma, T, \Delta)$




- This reflects one-step unfolding of $\mathfrak{t}$
- Consume $\gamma$ and output $\gamma'$,
  and put a continuation
- $\langle \gamma, \gamma' \rangle^{-1} \mathfrak{t} = \{ \langle w, w' \rangle \mid \langle \gamma w, \gamma' w \rangle \in \mathfrak{t} \}$

Simulating PDS $(Q, \Gamma \cup \langle T \rangle, \Delta')$

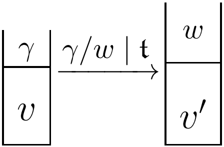


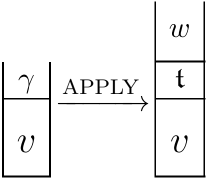



# Simulating TrPDS by ordinary PDS

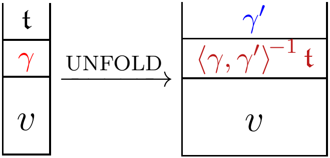
Simulated TrPDS $(Q, \Gamma, T, \Delta)$

Simulating PDS $(Q, \Gamma \cup \langle T \rangle, \Delta')$

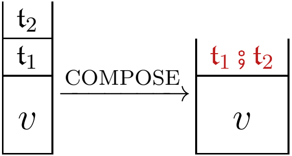


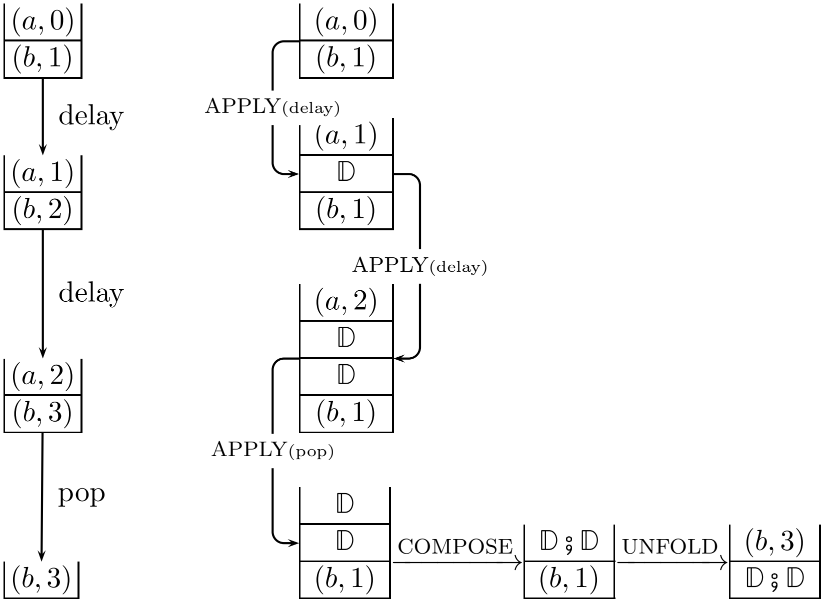

- This reflects one-step unfolding of $\mathfrak{t}$
- Consume $\gamma$ and output $\gamma'$, and put a continuation
- $\langle \gamma, \gamma' \rangle^{-1} \mathfrak{t} = \{ \langle w, w' \rangle \mid \langle \gamma w, \gamma' w \rangle \in \mathfrak{t} \}$




- Two configurations are equivalent
- $\mathfrak{t}_1$ was accumulated before $\mathfrak{t}_2$
- $\mathfrak{t}_1 \fatsemi \mathfrak{t}_2 = \{ \langle a, c \rangle \mid \langle a, b \rangle \in \mathfrak{t}_1, \langle b, c \rangle \in \mathfrak{t}_2 \}$



# Simulating Timed PDS by PDS with transduction



# Overview

1. TrPDS : Extending PDS with transductions
2. Simulating TrPDS by ordinary PDS
3. Decidability of reachability problem of finite TrPDS

# Concretization of stack

## Concretizing function

Obtaining stacks of TrPDS from a stack of simulating PDS.

$$\begin{array}{lcl} \|\cdot\| & : & (\Gamma \cup \langle T \rangle)^* \to \mathcal{P}(\Gamma^*) \\ \|\,\varepsilon\,\| & = & \{\ \varepsilon\ \} \\ \|\,\gamma w\,\| & = & \{\gamma w' \mid w' \in \|\,w\,\|\} \\ \|\,\mathfrak{t} w\,\| & = & \{w'\mathfrak{t} \mid w' \in \|w\|\} \end{array}$$

## Example

$$\left\|\ \begin{array}{|c|} 1 \\ \hline \mathbb{D} \\ \hline \mathbb{D} \\ \hline 0 \\ \hline \end{array}\ \right\| = \begin{array}{|c|} 1 \\ \hline 2 \\ \hline \end{array}$$

- Two effects of delay $\mathbb{D}$ were accumulated
- By concretizing, we obtain the expected stack

$$\left\|\ \begin{array}{|c|} 1 \\ \hline \widetilde{1^*} \\ \hline 2 \\ \hline \end{array}\ \right\| = \varnothing$$

- Simulating PDS does not detect a failure of application $\widetilde{1^*}$
- By concretizing, we notice a failure of application

# Simulation theorem

### Concretization of stack

Obtaining stacks of TrPDS from a stack of simulating PDS.

$$\|\cdot\| \;:\; (\Gamma \cup \langle T \rangle)^* \to \mathcal{P}(\Gamma^*)$$

### Theorem (Simulation theorem)

For a given $C \subseteq Q \times \Gamma^*$, $post^*(C) = \|Post^*(C)\|$

- $post^*(C)$ is the set of forward-reachable configurations from $C$ in (simulated) TrPDS,
- $Post^*(C)$ is the set of forward-reachable configurations from $C$ in (simulating) PDS.

# Calculating $post^*(C)$

We can effectively compute $post^*(C)$,

1. $post^*(C) = \|Post^*(C)\|$ holds,

# Calculating $post^*(C)$

We can effectively compute $post^*(C)$,

1. $post^*(C) = \|Post^*(C)\|$ holds,
2. $Post^*(C)$ is a regular set from reachability analysis for ordinary PDS,

# Calculating $post^*(C)$

We can effectively compute $post^*(C)$,

1. $post^*(C) = \|Post^*(C)\|$ holds,
2. $Post^*(C)$ is a regular set from reachability analysis for ordinary PDS,
3. $\|Post^*(C)\|$ is a regular set and effectively computable,
    - $\| \cdot \|$ is a (not length-preserving) transduction

# Calculating $post^*(C)$

We can effectively compute $post^*(C)$,

1. $post^*(C) = \|Post^*(C)\|$ holds,
2. $Post^*(C)$ is a regular set from reachability analysis for ordinary PDS,
3. $\|Post^*(C)\|$ is a regular set and effectively computable,
    - $\|\cdot\|$ is a (not length-preserving) transduction

$\Rightarrow$ We can solve the reachability problem of finite TrPDS.

# Conclusion and Future works

We have

- introduced extended pushdown system TrPDS in a more general and uniform manner,
  - Conditional and Timed PDSs and two-counter system without input are instances of TrPDS
- shown finite TrPDS can be simulated by PDS,
  - the reachability problem of finite TrPDS is decidable.
  - as the result, we have clarified why conditional and timed PDSs can be simulated by ordinary PDS.

## Future works

- Proving the undecidability of checking finiteness of $\langle T \rangle$
  - the closure $\langle T \rangle$ generated by $\left(T, ⨾, \langle \cdot, \cdot \rangle^{-1}\right)$
- Using rational transducers as stack manipulators rather than letter-to-letter transducers

# Finiteness of checking+homomorphism

TrPDS $\mathcal{S} = (Q, \Gamma, T, \Delta)$
$T = \text{checking} \uplus H$
$\text{checking} = \left\{ \widehat{R} \mid R \in \mathcal{R} \right\}$
$H$ is a finite set of homomorphism over $\Gamma$

## Lemma (exchanging law)

$$h \mathbin{;} \widetilde{R} = \widetilde{h^{-1}(R)} \mathbin{;} h$$

We can reduce any composition sequences into the following normal form:

$$(\widetilde{R_1} \mathbin{;} \cdots \mathbin{;} \widetilde{R_m}) \mathbin{;} (h_1 \mathbin{;} \cdots \mathbin{;} h_n).$$

# Another Applications

- We can use TrPDS to formalize a larger part of the HTML5 parser specification more than the existing work:
  - Ref *Reachability Analysis of the HTML5 Parser Specification and Its Application to Compatibility Testing.*[Minamide and Mori, 2012]
- I think TrPDS is useful for pushdown-analysis:
  - Ref *Introspective pushdown analysis of higher-order programs*[Earl et al., 2012]
  - Ref *Concrete Semantics for Pushdown Analysis: The Essence of Summarization*[Johnson and Horn, ]
- We have not yet adopted TrPDS to *Dense Timed PDS*[Abdulla et al., 2012a].

# Time complexity of reachability problem

| | |
|---|---|
| PDS | **PTIME**[Bouajjani et al., 1997, Esparza et al., 2000] |
| Conditional PDS | **EXPTIME**-complete[Esparza et al., 2003] |

At least, from the result of Conditional PDS, time complexity of reachability problem of TrPDS is **EXPTIME**-hard.
We cannot estimate the size of $\langle T \rangle$ from $T$, so we have not ingestigated the precise analysis.

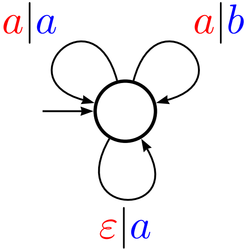
# Checking finiteness of $T$

We have shown undecidability of checking finiteness of the closure $(T, \fatsemi)$ by using the old result *uniform halting problem*[Hughes and Selkow, 1981].

On the other hand, undecidability of checking finiteness of the closure $(T, \fatsemi, \langle \cdot, \cdot \rangle^{-1})$ is not derived from undecidability of checking finitenss of $(T, \fatsemi)$.

# Rational transducer

If we employ general transducer or transduction as stack manipulator, then we should use transducer rather than transduction to treat $\varepsilon$-transitions.




$$
\begin{aligned}
\mathcal{L}(\mathfrak{t}) \quad = \{ \quad & (\varepsilon, a), (\varepsilon, aa), (\varepsilon, aaa), \dots, \\
& (a, aa), (a, aaa), (a, aaaa), \dots, \\
& (a, ba), (a, baa), (a, baaa), \dots, \\
& (a, ab), (a, aab), (a, aaab), \dots, \}
\end{aligned}
$$

Abdulla, P. A., Atig, M. F., and Stenman, J. (2012a).
Dense-timed pushdown automata.
In *LICS'12*, pages 35–44. IEEE Computer Society.

Abdulla, P. A., Atig, M. F., and Stenman, J. (2012b).
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, pages 58–69, Berlin, Heidelberg. Springer-Verlag.

Bouajjani, A., Esparza, J., and Maler, O. (1997).
Reachability analysis of pushdown automata: Application to model-checking.
In *CONCUR '97: Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150.

Earl, C., Sergey, I., Might, M., and Van Horn, D. (2012).
Introspective pushdown analysis of higher-order programs.

In *Proceedings of the 17th ACM SIGPLAN international conference on Functional programming*, ICFP '12, pages 177–188. ACM.

Esparza, J., Hansel, D., Rossmanith, P., and Schwoon, S. (2000).
Efficient Algorithms for Model Checking Pushdown Systems.
In *CAV'00*, volume 1855 of *Lecture Notes in Computer Science*, pages 232–247. Springer Berlin Heidelberg.

Esparza, J., Kučera, A., and Schwoon, S. (2003).
Model checking LTL with regular valuations for pushdown systems.
*Information and Computation*, 186(2):355 – 376.

Finkel, A., Willems, B., and Wolper, P. (1997).
A direct symbolic approach to model checking pushdown systems.
In *INFINITY '97*, volume 9 of *ENTCS*, pages 27–37.

Hughes, C. E. and Selkow, S. M. (1981).

The finite power property for context-free languages.
*Theoretical Computer Science*, 15(1):111–114.

Johnson, J. and Horn, D. V.
Concrete semantics for pushdown analysis: The essence of summarization.
Workshop on Higher-Order Program Analysis(HOPA).

Minamide, Y. and Mori, S. (2012).
Reachability Analysis of the HTML5 Parser Specification and Its Application to Compatibility Testing.
In *FM 2012: Formal Methods*, volume 7436 of *Lecture Notes in Computer Science*, pages 293–307.