Simplifying transducers
removing two-wayness and non-determinism

Pierre-Alain Reynier

LIF, Aix-Marseille Université & CNRS

Based on joint works with L Daviaud, E Filiot, O Gauwin, I Jecker, B Monmege, F Servais, JM Talbot and D Villevalois
Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
- PostDoc at ULB on controller synthesis with Jean-François Raskin
Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
- PostDoc at ULB on controller synthesis with Jean-François Raskin
- Ass Prof at Marseille
  - Mid 2009: New collaboration between ULB and UMarseille: with JF Raskin, E Filliot, F Servais and JM Talbot
  - Topic: streamability of XML transformations
Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
- PostDoc at ULB on controller synthesis with Jean-François Raskin
- Ass Prof at Marseille
  ➔ Mid 2009: New collaboration between ULB and UMarseille: with JF Raskin, E Filliot, F Servais and JM Talbot
  Topic: streamability of XML transformations

Determinization of visibly pushdown transducers
Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
- PostDoc at ULB on controller synthesis with Jean-François Raskin
- Ass Prof at Marseille
  ➔ Mid 2009: New collaboration between ULB and UMarseille: with JF Raskin, E Filliot, F Servais and JM Talbot
  Topic: streamability of XML transformations

Determinization of visibly pushdown transducers

We did not succeed to solve this problem...
...but we found interesting open problems on word transducers
## From Languages to Transductions

<table>
<thead>
<tr>
<th>Languages</th>
<th>Transductions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input → {0, 1}</td>
<td>Input → Outputs</td>
</tr>
<tr>
<td>automata</td>
<td>transducers</td>
</tr>
<tr>
<td>accept inputs</td>
<td>transform inputs</td>
</tr>
</tbody>
</table>

Applications:
- Word-to-word transducers:
  - language and speech processing
  - model-checking infinite state-space systems
  - verification of web sanitizers
  - string pattern matching

- Nested-word-to-word transducers:
  - XML transformations
  - model for recursive programs
From Languages to Transductions

<table>
<thead>
<tr>
<th>Languages</th>
<th>Transductions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Input \rightarrow {0, 1}$</td>
<td>$Input \rightarrow Outputs$</td>
</tr>
<tr>
<td>automata</td>
<td>transducers</td>
</tr>
<tr>
<td>accept inputs</td>
<td>transform inputs</td>
</tr>
</tbody>
</table>

Applications:

- Word-to-word transducers:
  - language and speech processing
  - model-checking infinite state-space systems
  - verification of web sanitizers
  - string pattern matching

- Nested-word-to-word transducers:
  - XML transformations
  - model for recursive programs
(Two-way) finite state transducers
= associate output words with transitions of a finite state automaton

Example (A transducer $T$)

\[
\begin{array}{c}
\begin{array}{c}
\epsilon \\
a \quad b \\
\epsilon
\end{array}
\quad \begin{array}{c}
a \quad a \\
| \quad | \quad |
\end{array}
\quad \begin{array}{c}
\epsilon \\
\epsilon
\end{array}
\end{array}
\]

Semantics $\mathcal{J} T$: $f : \downarrow w \downarrow \mapsto a \#_{a}(w)$, with $w \in \{a, b\}^*$

Non-determinism: semantics is a relation
(Two-way) finite state transducers
= associate output words with transitions of a finite state automaton

Example (A transducer $T$)

Semantics $\llbracket T \rrbracket$: $f : \Downarrow w \Downarrow \mapsto a \# a(w)$, with $w \in \{a, b\}^*$

Non-determinism: semantics is a relation

A transducer is:

- **functional** if it realizes a function
- **deterministic** if the underlying automaton is deterministic

Classes: DFT, fNFT, NFT
(Two-way) finite state transducers

= associate output words with transitions of a finite state automaton

Example (A transducer $T$)

$\begin{align*}
t | \epsilon, +1 & \quad a | a, +1 \\
\quad & \quad a | \epsilon, -1 \\
\quad b | \epsilon, +1 & \quad b | b, -1 \\
\quad & \quad \epsilon, 0
\end{align*}$

Semantics $[T]$: $f : \begin{array}{c}
\begin{array}{c}
s | \epsilon, +1 \\
\quad & \quad \epsilon, -1 \\
\quad b | \epsilon, +1 & \quad b | b, -1 \\
\quad & \quad \epsilon, 0
\end{array}
\end{array} \mapsto a \# a(w) b \# b(w)$, with $w \in \{a, b\}^*$

Non-determinism: semantics is a relation

A transducer is:

- functional if it realizes a function
- deterministic if the underlying automaton is deterministic

Classes: DFT, fNFT, NFT, 2DFT, f2NFT, 2NFT
Classes of Transductions

- DFTs
- fNFTs
- NFTs
- 2NFTs
- 2DFTs = f2NFTs = MSOT

Valuedness

Expressiveness

[EH01]
Classes of Transductions

- DFTs
- fNFTs
- NFTs
- 2NFTs
- 2DFTs = f2NFTs = MSOT

$u \mapsto \text{mirror}(u)$

Expressiveness

Valuelessness
Classes of Transductions

DFTs ⊃ fNFTs ⊃ NFTs ⊃ \{(a, a), (a, b)\} ⊃ 2NFTs = 2DFTs = f2NFTs = MSOT

expressiveness

valuedness
Classes of Transductions

\[ u \mapsto \text{last}(u)^{|u|} \]

- DFTs \( \subseteq \) \( u \mapsto \text{last}(u)^{|u|} \) \( \subseteq \) \( \text{NFTs} \)
- fNFTs \( \subseteq \) \( \text{NFTs} \)
- \( 2\text{DFTs} = f2\text{NFTs} \)
- MSOT

\[ u \mapsto \text{last}(u)^{|u|} \]

expressiveness
Classes of Transductions

- DFTs
- fNFTs
- NFTs
- 2NFTs

\[ \text{sequential functions} \]

- expressiveness

\[ \text{valuedness} \]

\[ \text{DFTs} \subseteq f\text{NFTs} \subseteq \text{NFTs} \subseteq 2\text{NFTs} \]

\[ 2\text{DFTs} = f\text{2NFTs} = \text{MSOT} \]
Classes of Transductions

- DFTs
- fNFTs
- NFTs
- 2NFTs

Expressiveness:
- Sequential functions
- Rational functions

Valuedness:
- $\subseteq$

$\text{DFTs} \subseteq \text{fNFTs} \subseteq \text{NFTs} \subseteq \text{2NFTs} = \text{MSOT}$

Twinning Property

$\text{delay}(v_1, w_1) = \text{delay}(v_1, v_2, w_1, w_2)$
Classes of Transductions

- DFTs ⊆ NFTs ⊆ 2NFTs
- 2DFTs = f2NFTs = MSOT
- sequential functions
- rational functions
- regular functions
- valuedness
- expressiveness
Classes of Transductions

Twinning Property

\[ q_0 \xrightarrow{u_1|v_1} q \]
\[ p_0 \xrightarrow{u_1|w_1} p \]

\[ \text{delay}(v_1, w_1) = \text{delay}(v_1 v_2, w_1 w_2) \]

\[ \text{NFTs} \subset \text{2NFTs} \]
\[ \text{DFTs} \subset \text{fNFTs} \subset \text{2DFTs=f2NFTs=MSOT} \]

valuedness

sequential functions

rational functions

regular functions

expressiveness

PTime

[Choffrut77, WK95, BCPS03]
Classes of Transductions

- **DFTs**
- **fNFTs**
- **NFTs**
- **2NFTs**

**Valuedness**

- Sequential functions
- Rational functions
- Regular functions

**Expressiveness**

- PTIME

- \[ \text{PTIME} \subseteq \text{NFTs} \]
- \[ \text{NFTs} \subseteq \text{fNFTs} \]
- \[ \text{fNFTs} \subseteq \text{2NFTs} \]
- \[ \text{2DFTs} = \text{f2NFTs} = \text{MSOT} \]

References:

- [Schützenberger75]
- [GI83, BCPS03]
- [Choffrut77, WK95, BCPS03]
- [Schützenberger75]
- [GI83, BCPS03]
- [FGRS13]
- [CK87]
Classes of Transductions

- DFTs ⊂ fNFTs ⊂ NFTs ⊂ 2NFTs
- PTIME ∪ decidable [CK87]
- sequential functions ⊂ rational functions ⊂ regular functions
- valuedness expressiveness
Classes of Transductions

- DFTs
- fNFTs
- 2NFTs
- NFTs

Classes are ordered by valuedness and expressiveness.

PTIME

sequential functions

rational functions

regular functions

undecidable [BGMP15]

decidable

Twinning Property

Pierre-Alain Reynier (LIF, AMU & CNRS)

Simplifying transducers

LSV’20th Anniversary
Classes of Transductions

- DFTs
- fNFTs
- NFTs
- 2NFTs
- 2DFTs = f2NFTs = MSOT

Expressiveness:
- Sequential functions
- Rational functions
- Regular functions

Valuedness:
- PTIME
- Decidable

Undecidable:
- Twinning Property

References:
- Choffrut77, WK95, BCPS03
- Schützenberger75, GI83, BCPS03
- CK87, BGMP15, FGRS13

PTime
Simplification of models

Given a (complex) model of a transformation, does there exist an equivalent simpler model?

Natural question:
- minimization of automata
Simplification of models

Given a (complex) model of a transformation, does there exist an equivalent simpler model?

Natural question:
- minimization of automata
- determinization
Simplification of models

Given a (complex) model of a transformation, does there exist an equivalent simpler model?

→ Natural question:
  - minimization of automata
  - determinization
  - 2way: minimize number of passes
Simplification of models

Given a (complex) model of a transformation, does there exist an equivalent simpler model?

- Natural question:
  - minimization of automata
  - determinization
  - 2way: minimize number of passes
  - automata with registers: minimize number of registers
  - ...
This talk

- Sequential functions
- Rational functions
- Regular functions
- NFTs
- fNFTs
- 2DFTs = f2NFTs = MSOT

Removing two-wayness
Removing non-determinism

Pierre-Alain Reynier (LIF, AMU & CNRS)
Simplifying transducers
LSV’20th Anniversary
This talk

valuedness

DFTs

NFTs

fNFTs

2NFTs

Removing two-wayness

expressiveness

sequential functions

rational functions

regular functions

2DFTs=f2NFTs = MSOT

Removing non-determinism

Simplifying transducers
This talk

Removing non-determinism

NFTs ⊂ 2NFTs

2DFTs = f2NFTs = MSOT

Expressiveness

Sequential functions

Rational functions

Regular functions

Valuedness
Overview

1. Introduction

2. Removing two-wayness

3. Removing non-determinism

4. Conclusion
Overview

1. Introduction

2. Removing two-wayness

3. Removing non-determinism

4. Conclusion
Rabin and Scott for transducers

Lemma

If $T$ is one-way definable then every zigzag transducer of $T$ is also one-way definable.

Decision Procedure:

1. Repeat $N$ times:
   - Are all zigzag transducers of $T$ $NFT$-definable?
     - Yes: $T \leftarrow \text{squeeze}(T)$
     - No: STOP: the initial $2NFT$ was not $NFT$-definable!

2. Remove backward transitions: you get an equivalent $NFT$
Rabin and Scott for transducers

Lemma

If $T$ is one-way definable then every zigzag transducer of $T$ is also one-way definable.

Decision Procedure:

1. repeat $N$ 2 times:
   - ▶ are all zigzag transducers of $T$ NFT-definable?
     - ⋆ yes: $T \leftarrow$ squeeze ($T$)
     - ⋆ no: STOP: the initial NFT was not NFT-definable!

2. remove backward transitions: you get an equivalent NFT
Lemma

If $T$ is one-way definable then every zigzag transducer of $T$ is also one-way definable.

Decision Procedure:
1. repeat $N$ times:
   - are all zigzag transducers of $T$ $N$FT-definable?
     - yes: $T \leftarrow \text{squeeze}(T)$
     - no: STOP: the initial $N$FT was not $N$FT-definable!

2. remove backward transitions: you get an equivalent NFT
Rabin and Scott for transducers

Lemma

If $T$ is one-way definable then every zigzag transducer of $T$ is also one-way definable.

Decision Procedure:
1. repeat $N$ 2 times:
   - ▶ are all zigzag transducers of $T$ NFT-definable?
     - yes: $T \leftarrow \text{squeeze}(T)$
     - no: STOP: the initial 2NFT was not NFT-definable!
2. remove backward transitions: you get an equivalent NFT
Lemma

If $T$ is one-way definable then every zigzag transducer of $T$ is also one-way definable.

Decision Procedure:

1. repeat $N$ 2 times:
   - ▶ are all zigzag transducers of $T$ NFT-definable?
     - ⋆ yes: $T \leftarrow \text{squeeze}(T)$
     - ⋆ no: STOP: the initial 2NFT was not NFT-definable!

2. remove backward transitions: you get an equivalent NFT.
Rabin and Scott for transducers

Lemma

If \( T \) is one-way definable then every zigzag transducer of \( T \) is also one-way definable.

Decision Procedure:

1. \( \text{repeat} \)

   \( N \) \( \times 2 \) times:

   ▶ are all zigzag transducers of \( T \) \( \text{NFT-definable?} \)

   ⋆ yes:

   \( T \leftarrow \text{squeeze} (T) \)

   ⋆ no: STOP: the initial \( 2\text{NFT} \) was not \( \text{NFT-definable!} \)

2. \( \text{remove backward transitions: you get an equivalent NFT} \)

\( \leq \#\text{states}^2 \) steps

\( \iff \) squeeze

\( \uparrow \) squeeze
Lemma

If \( T \) is one-way definable then every zigzag transducer of \( T \) is also one-way definable.
Lemma

If $T$ is one-way definable then every zigzag transducer of $T$ is also one-way definable.

Decision Procedure:

1. repeat $N^2$ times:
   - are all zigzag transducers of $T$ NFT-definable?
     - yes: $T \leftarrow \text{squeeze}(T)$
     - no: STOP: the initial 2NFT was not NFT-definable!

2. remove backward transitions: you get an equivalent NFT
Rabin and Scott for transducers (2)

Consider a zigzag transducer $Z$

\[ x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0 \]
Rabin and Scott for transducers (2)

Consider a zigzag transducer $Z$

![Diagram of zigzag transducer]

$x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0$

$\implies \alpha, \beta, \gamma, \delta$ have conjugate primitive roots of length polynomial in $|Z|$. 

Consider a zigzag transducer $Z$

\[
\begin{array}{c}
  x_2 \\
  x_1 \\
  x_0 \\
\end{array}
\begin{array}{c}
  \beta \\
  \alpha \\
  \delta \\
\end{array}
\begin{array}{c}
  y_2 \\
  y_1 \\
  y_0 \\
\end{array}
\]

\[
x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0
\]

$\implies \alpha, \beta, \gamma, \delta$ have conjugate primitive roots of length polynomial in $|Z|$

**Theorem**

*Given a zigzag transducer $Z$*, we build a fNFT $Z'$ s.t.:

1. $|Z'| = \exp(O(|Z|))$
2. $R(Z') \subseteq R(Z)$
3. $Z$ is NFT-definable $\iff$ $\text{dom}(Z) \subseteq \text{dom}(Z')$

$\implies$ is decidable.
Consider a zigzag transducer $Z$

\[
\begin{array}{c}
 x_0 \quad \delta \quad y_0 \\
 x_1 \quad \alpha \quad y_1 \\
 x_2 \quad \beta \quad y_2 \\
 \end{array}
\]

$x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0$

$\implies \alpha, \beta, \gamma, \delta$ have conjugate primitive roots of length polynomial in $|Z|$

Yields a non-elementary decision procedure

**Theorem**

Given a zigzag transducer $Z$, we build a fNFT $Z'$ s.t.:

1. $|Z'| = \exp(O(|Z|))$
2. $R(Z') \subseteq R(Z)$
3. $Z$ is NFT-definable $\iff$ $\dom(Z) \subseteq \dom(Z')$
4. is decidable.
Streaming model: Deterministic Turing Transducer

Input Tape (read only)

1 0 0 1 1 1 #

Working Tape (read/write)

1 0 0 0 0 1 0 #

Output Tape (write only)

0 1 1 0 1 1 #

∃ B ∈ N · ∀ u ∈ dom(T) T(u) can be computed with B-bounded memory?

Memory Measured on this tape only!
Streaming model: Deterministic Turing Transducer

Bounded Memory Problem

Input: a transformation $T$
Output: can $T$ be realized with bounded memory?

$$\exists B \in \mathbb{N} \cdot \forall u \in \text{dom}(T)$$

$T(u)$ can be computed with $B$-bounded memory?
Streamability of word-to-word transformations

**Observation:**

\[ f : \Sigma^* \rightarrow \Sigma^* \text{ is bounded memory iff it is DFT-definable.} \]
Streamability of word-to-word transformations

Observation:

\[ f : \Sigma^* \rightarrow \Sigma^* \text{ is bounded memory} \text{ iff it is DFT-definable.} \]

Corollary

**Bounded memory is decidable for regular word functions.**
Recent works on removing two-wayness

Two new techniques to obtain elementary complexity:

- [BGMP17] (Bordeaux)
  - careful analysis of loops
  - algebraic techniques (idempotent element, Simon)
  - decision in 2ExpSpace
  - also yields decidability results related to sweeping transducers

- [MRT17] (Marseille, unpublished)
  - normalization procedure
  - more amenable to implementation
  - promising technique to address determinization of visibly pushdown transducers
Recent works on removing two-wayness

Two new techniques to obtain elementary complexity:

- **[BGMP17]** (Bordeaux)
  - careful analysis of loops
  - algebraic techniques (idempotent element, Simon)
  - decision in 2ExpSpace
  - also yields decidability results related to sweeping transducers

- **[MRT17]** (Marseille, unpublished)
  - normalization procedure
  - more amenable to implementation
  - promising technique to address determinization of visibly pushdown transducers
Overview

1 Introduction
2 Removing two-wayness
3 Removing non-determinism
4 Conclusion
Removing non-determinism

\[ \text{sequential functions} \subset \text{rational functions} \subset \text{regular functions} \]

\[ \text{DFTs} \subset \text{fNFTs} \subset \text{NFTs} \subset 2\text{NFTs} \subset 2\text{DFTs} = f2\text{NFTs} = \text{MSOT} \]
Multi-sequential functions

**Fact:** Not every one-way transducer can be determinized

**Example**

Semantics $[T] = \text{Last}$: $w\sigma \mapsto \sigma|w|+1$, with $\sigma \in \{a, b\}$, $w \in \{a, b\}^+$
Multi-sequential functions

**Fact:** Not every one-way transducer can be determinized

Example

![Diagram of multi-sequential functions](image)

Semantics \([T] = \text{LAST} : w\sigma \mapsto \sigma^{\text{\#}w+1}\), with \(\sigma \in \{a, b\}\), \(w \in \{a, b\}^+\)

But \(\text{LAST}\) can be written as the “sum” of two sequential functions
Multi-sequential functions

**Fact:** Not every one-way transducer can be determinized

**Definition ([CS86])**

*Multi-sequential functions* are defined as functions that can be realized as finite union of sequential transducers.

Motivation: parallel evaluation of the function

Examples: (\( \text{LAST} : \ w\sigma \mapsto \sigma|w|+1 \), with \( \sigma \in \{a, b\}, w \in \{a, b\}^+ \))

- \( \text{LAST} \) is multi-sequential: split the domain as \( \Sigma^*a \uplus \Sigma^*b \)
**Multi-sequential functions**

**Fact:** Not every one-way transducer can be determinized

**Definition ([CS86])**

*Multi-sequential functions* are defined as functions that can be realized as finite union of sequential transducers.

Motivation: parallel evaluation of the function

Examples: (\(\text{LAST} : \ w\sigma \mapsto \sigma|w|^{+1}\), with \(\sigma \in \{a, b\}\), \(w \in \{a, b\}^{+}\))

- \(\text{LAST}\) is multi-sequential: split the domain as \(\Sigma^*a \cup \Sigma^*b\)
- \(\text{LAST}^2 : u_1 \# u_2 \mapsto \text{LAST}(u_1) \# \text{LAST}(u_2)\) is multi-sequential: split the domain according to \(\text{last}(u_1), \text{last}(u_2) \in \{a, b\}\)
Multi-sequential functions

**Fact:** Not every one-way transducer can be determinized

**Definition ([CS86])**

*Multi-sequential functions* are defined as functions that can be realized as finite union of sequential transducers.

Motivation: parallel evaluation of the function

Examples:

1. \( \text{Last} : \ w\sigma \mapsto \sigma^{|w|+1} \), with \( \sigma \in \{a, b\}, w \in \{a, b\}^+ \)
   - \( \text{Last} \) is multi-sequential: split the domain as \( \Sigma^*a \cup \Sigma^*b \)
   - \( \text{Last}^2 : u_1 \# u_2 \mapsto \text{Last}(u_1) \# \text{Last}(u_2) \) is multi-sequential:
     - split the domain according to \( \text{last}(u_1), \text{last}(u_2) \in \{a, b\} \)
   - \( \text{Last}^* : u_1 \# \ldots \# u_n \mapsto \text{Last}(u_1) \# \ldots \# \text{Last}(u_n) \) is not multi-seq.
### Multi-sequential functions

**Fact:** Not every one-way transducer can be determinized

**Definition ([CS86])**

*Multi-sequential functions* are defined as functions that can be realized as finite union of sequential transducers.

**Motivation:** parallel evaluation of the function

**Theorem ([CS86,FJ15])**

*Multi-sequentiality can be decided in Ptime.*

This work: minimization of the size of the union.
From rational to sequential functions [Choffrut77]

Sequentiality Problem

Input: a fNFT
Question: does there exist an equivalent DFT?

Theorem ([Choffrut77])

Let $T$ be a fNFT. $T$.f.a.e:

- there exists a DFT $T'$ s.t. $T \equiv T'$
- $T$ satisfies the twinning property
- $\mathcal{T}[T]$ satisfies the Lipschitz property

$$\exists L \in \mathbb{N} \mid \forall u, v \in \text{dom}(f), d(f(u), f(v)) \leq L \cdot d(u, v)$$
From rational to sequential functions [Choffrut77]

Sequenciality Problem
Input: a fNFT
Question: does there exist an equivalent DFT?

Theorem ([Choffrut77])
Let $T$ be a fNFT. $T$.f.a.e:
- there exists a DFT $T'$ s.t. $T \equiv T'$
- $T$ satisfies the twinning property
- $\lceil T \rceil$ satisfies the Lipschitz property
  \[ \exists L \in \mathbb{N} \mid \forall u, v \in \text{dom}(f), d(f(u), f(v)) \leq L.d(u, v) \]

Twinning Property can be decided in PTime ([WK95])

Corollary
The Sequentiality Problem is decidable in PTime.
Twinning Property [Choffrut77]

Define:
\[ \text{delay}(u, v) = \text{lcp}(u, v)^{-1} . (u, v) \]

Example:
\[ \text{lcp}(aaa, aab) = aa \]
\[ \text{delay}(aaa, aab) = (a, b) \]

For all situations like:
we have \( \text{delay}(v_1, w_1) = \text{delay}(v_1 v_2, w_1 w_2) \)
Twinning Property [Choffrut77]

Define:
\[ \text{delay}(u, v) = \text{lcp}(u, v)^{-1} \cdot (u, v) \]

Example:
\[ \text{lcp}(aaa, aab) = aa \]
\[ \text{delay}(aaa, aab) = (a, b) \]

Determinization procedure:
- subset construction from initial states
- output longest common prefix
- store the unproduced outputs in the state

States are sets \{((p, a), (q, \varepsilon), (s, bb))\}

For all situations like:
\[ \text{we have } \text{delay}(v_1, w_1) = \text{delay}(v_1 v_2, w_1 w_2) \]
Twinning Property [Choffrut77]

Define:
\[
\text{delay}(u, v) = \text{lcp}(u, v)^{-1}.(u, v)
\]

Example:
\[
\text{lcp}(aaa, aab) = aa \\
\text{delay}(aaa, aab) = (a, b)
\]

Determinization procedure:
- subset construction from initial states
- output longest common prefix
- store the unproduced outputs in the state

States are sets \\{ (p, a), (q, \varepsilon), (s, bb) \\}

For all situations like:

\[
\text{we have delay}(v_1, w_1) = \text{delay}(v_1 v_2, w_1 w_2)
\]

Lemma

\( T \models \text{Twinning Property} \implies \text{bounded delays} \implies \text{termination of subset constr.} \)
Characterization of $k$-sequential functions

$k$-sequentiality Problem

Input: a fNFT $T$ and $k \in \mathbb{N}$
Question: does there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$?
Characterization of $k$-sequential functions

$k$-sequentiality Problem

Input: a fNFT $T$ and $k \in \mathbb{N}$
Question: does there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$?

Theorem

Let $T$ be a fNFT and $k \in \mathbb{N}$. T.f.a.e:

- there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$
- $T$ satisfies the branching twinning property of order $k$
- $\ll T \gg$ satisfies the $k$-Lipschitz property
Characterization of $k$-sequential functions

$k$-sequentiality Problem

Input: a fNFT $T$ and $k \in \mathbb{N}$
Question: does there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$?

Theorem

Let $T$ be a fNFT and $k \in \mathbb{N}$. T.f.a.e:
- there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$
- $T$ satisfies the branching twinning property of order $k$
- $[T]$ satisfies the $k$-Lipschitz property

Theorem

The $k$-sequentiality Problem is PSpace-complete ($k$ given in unary).
Branching twinning property of order $k$

For all situations like:

$k$ not synchronised loops

For all $0 \leq i < j \leq k$ s.t. for every loop $\ell$ with same input words,
we have $\text{delay}(w_{1,i} \ldots w_{\ell,i}, w_{1,j} \ldots w_{\ell,j}) = \text{delay}(w_{1,i} \ldots w_{\ell,i}w'_{\ell,i}, w_{1,j} \ldots w_{\ell,j}w'_{\ell,j})$
Branching twinning property of order $k$

Tree representation of input words:
Branching twinning property of order $k$

**Theorem**

A fNFT is definable by a union of $k$ DFT iff it satisfies the BTP of order $k$. 

Sketch of proof of $\Leftarrow$:

- Induction on $k$; case $k=1$ already solved
- Consider subset construction with delays
- If a delay is too large in some set $S$, find a loop split $S = S' \cup S''$, depending on what happens on the loop
- Define $k'$ as the smallest integer s.t. $T_{|S'} = BTP_{k'}$ (same with $k''$)
- $T = BTP_k$ entails $k' + k'' \leq k$

Pierre-Alain Reynier (LIF, AMU & CNRS)  
Simplifying transducers  
LSV'20th Anniversary
Branching twinning property of order $k$

**Theorem**

A fNFT is definable by a union of $k$ DFT iff it satisfies the BTP of order $k$.

Sketch of proof of $\Leftarrow$:
Induction on $k$; case $k = 1$ already solved
Consider subset construction with delays
Branching twinning property of order $k$

Theorem

A fNFT is definable by a union of $k$ DFT iff it satisfies the BTP of order $k$.

Sketch of proof of $\Leftarrow$:

Induction on $k$; case $k = 1$ already solved

Consider subset construction with delays

- if a delay is too large in some set $S$, find a loop
- split $S$ into $S' \cup S''$, depending on what happens on the loop
- define $k'$ as the smallest integer s.t. $T_{S'} \models BTP_{k'}$ (same with $k''$)
- $T \models BTP_k$ entails $k' + k'' \leq k$
- apply induction hypothesis
Overview

1  Introduction

2  Removing two-wayness

3  Removing non-determinism

4  Conclusion
Summary

- Streamability/efficient evaluation of transformations
- Open problems on word transducers
- Removing two-wayness: decidable, now in 2ExpSpace
- Removing non-determinism: $k$-sequentiality PSpace-complete
**Summary**

- Streamability/efficient evaluation of transformations
- Open problems on word transducers
- Removing two-wayness: decidable, now in 2ExpSpace
- Removing non-determinism: $k$-sequentiality PSpace-complete

**Extensions:**

- logical presentation of transductions
- functional $\leadsto$ finite-valued
- transducers $\leadsto$ weighted automata on semigroups ($+$ hypotheses)
Summary

- Streamability/efficient evaluation of transformations
- Open problems on word transducers
- Removing two-wayness: decidable, now in 2ExpSpace
- Removing non-determinism: $k$-sequentiality PSpace-complete

Extensions:

- logical presentation of transductions
- functional $\sim$ finite-valued
- transducers $\sim$ weighted automata on semigroups (+ hypotheses)

Growing interest for transducers:

- LIF LIS
- LaBRI (2Way to 1Way)
- LSV!
- Warsaw (origin semantics)
- Microsoft (symbolic transducers)
- UPenn (streaming string transducers)
Streaming String Transducers

Definition ([AC10])

Streaming String Transducer = DFA + finite set of registers $X, Y, Z$

\[ b; X := Xa \]
\[ a; X := Xa \]
\[ b; X := Xa \]
Streaming String Transducers

Definition ([AC10])

Streaming String Transducer = DFA + finite set of registers $X, Y, Z$

→ allows to recover previously defined classes of functions:

- **k-sequential functions**: $k$ registers $X_1, \ldots, X_k$
  
  $X_i := X_i u$

- **Rational functions**: $X := Yu$

- **Regular functions**: $X := uYv$
  
  $X := YZ$

  linear updates
Streaming String Transducers

**Definition ([AC10])**
Streaming String Transducer = DFA + finite set of registers $X, Y, Z$

→ allows to recover previously defined classes of functions:

- **$k$-sequential functions**
  - $k$ registers $X_1, \ldots, X_k$
  - $X_i := X_i u$

- **Rational functions**
  - $X := Y u$

- **Regular functions**
  - $X := u Y v$
  - $X := Y Z$
  - linear updates

**Theorem**
One can minimize the number of registers in the class of rational functions.
Landscape of rational functions

- DFTs
- fNFTs

- Twinning Ppty
- Multi-seq

2-app. SSTs: TP of order 2
k-app. SSTs: TP of order k
2-seq: BTP of order 2
k-seq: BTP of order k
Landscape of rational functions

2-app. SSTs: TP of order 2

Twinning Ppty

DFTs
Landscape of rational functions

k-app. SSTs: TP of order k

2-app. SSTs: TP of order 2

Twinning Ppty

DFTs

fNFTs
Landscape of rational functions

- fNFTs
- DFTs
- Twinning Ppty
- 2-app. SSTs: TP of order 2
- $k$-app. SSTs: TP of order $k$
- 2-seq: BTP of order 2
- Multi-seq

Pierre-Alain Reynier (LIF, AMU & CNRS)
Simplifying transducers
LSV'20th Anniversary
Landscape of rational functions

- **DFTs**
  - **Twinning Ppty**
  - **2-app. SSTs**: TP of order 2
  - **k-app. SSTs**: TP of order k

- **fNFTs**
  - **2-seq**: BTP of order 2
  - **k-seq**: BTP of order k
Landscape of rational functions

- 2-app. SSTs: TP of order 2
- 2-seq: BTP of order 2
- k-app. SSTs: TP of order k
- k-seq: BTP of order k
- Twinning Ppty
- DFTs
- Multi-seq

fNFTs
Some perspectives

- Register minimization for regular functions
  - deal with prepending of words ($X := uYv$)
  - deal with concatenation of registers ($X := YZ$)

- Algebraic characterizations (bimachines [RS91])

- Specification formalism

- Tool development

- Symbolic transducers

- Back to nested words!
Some perspectives

- Register minimization for regular functions
  - deal with prepending of words \((X := uYv)\)
  - deal with concatenation of registers \((X := YZ)\)

- Algebraic characterizations (bimachines \([RS91]\))

- Specification formalism

- Tool development

- Symbolic transducers

- Back to nested words!

Thanks!