A survey on transducers
decidability, logic and algebra

Pierre-Alain Reynier

LIF, Aix-Marseille University & CNRS
Formal methods to improve software

Software systems are complex and ubiquitous

- critical systems ➔ reliability
- widespread ➔ efficiency, scalability

➔ need for formal methods
Formal methods to improve software

Software systems are complex and ubiquitous
- critical systems ➔ reliability
- widespread ➔ efficiency, scalability

➔ need for formal methods

Automata-based approaches:
- model checking
- controller synthesis
- performance evaluation
- model optimization

Objective: Improve our theoretical understanding of automata models
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>
From Languages to Transductions

<table>
<thead>
<tr>
<th>Languages</th>
<th>Transductions</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Input</em> → {0, 1}</td>
<td><em>Input</em> → 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
  - reactive systems
  - verification of web sanitizers (tool BEK)
  - string pattern matching

- **Nested-word-to-word transducers:**
  - XML transformations
  - model for recursive programs
Two-fold objective: theory and applications

Develop the **theory** of transducers/transductions:

- expressiveness
- closure properties
- decidability
Two-fold objective: theory and applications

Develop the **theory** of transducers/transductions:

- expressiveness
- closure properties
- decidability

and their **applications**:

- verification
  - equivalence, type checking \( T(L_{in}) \subseteq L_{out} \)
- model optimization
  - streaming evaluation, minimization of resources
Simplification of models

General Problem

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

Natural question:

- minimization of automata
- make model deterministic
- reduce number of registers
- 2way: reduce number of passes
- ...
Automata/Logic/Algebra connections

Rich theory of regular languages: multiple presentation

Each presentation owns its assets:

- Automata: decision problems
- Logic: closure properties
- Algebra: characterizations of subclasses

⇒ try to lift this theory to transductions
Overview

1. Introduction

2. Models of transducers

3. Decidability results (based on patterns)

4. Connections with Logic and Algebra

5. Nested words

6. Conclusion
Overview

1. Introduction

2. Models of transducers

3. Decidability results (based on patterns)

4. Connections with Logic and Algebra

5. Nested words

6. Conclusion
(Two-way) finite state transducers
= associate output words with transitions of a finite state automaton

Example (A transducer $T$)

Semantics $\llbracket T \rrbracket$: $f: \vdash w \leadsto a \#_a(w)$, 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 $T$)

Semantics $\mathbb{T}$: $f : \leftarrow w \mapsto a^{\#_a(w)} b^{\#_b(w)}$, 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 \(\subset\) fNFTs \(\subset\) NFTs \(\subset\) 2NFTs \(\subset\) 2DFTs = f2NFTs

valuedness

expressiveness

\[\text{[EH01]}\]
Classes of Transductions

\[ u \mapsto \text{mirror}(u) \]

\[ \begin{array}{c}
\text{valuedness} \\
\downarrow
\end{array} \]

\[ \begin{array}{c}
\text{DFTs} \\
\text{fNFTs} \\
\text{NFTs} \\
\text{2NFTs} \\
\end{array} \]

\[ \begin{array}{c}
\text{expressiveness} \\
\uparrow
\end{array} \]

\[ \begin{array}{c}
\subseteq \text{2DFTs}=\text{f2NFTs} \\
\subseteq \text{2NFTs} \\
\subseteq \text{NFTs} \\
\subseteq \text{DFTs} \\
\end{array} \]
Classes of Transductions

- DFTs
- fNFTs
- NFTs
- 2NFTs

\{(a, a), (a, b)\}

\[ \text{expressiveness} \]

\[ \text{valuedness} \]
Classes of Transductions

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

- DFTs \( \subset \) fNFTs \( \subset \) 2DFTs = f2NFTs
- NFTs \( \subset \) 2NFTs

\( u \mapsto \text{last}(u) | u | \)
Classes of Transductions

- DFTs
- fNFTs
- 2DFTs $= f2NFTs$
- NFTs
- 2NFTs

Expressiveness:
- sequential functions

Valuedness:
Classes of Transductions

- DFTs
- fNFTs
- 2DFTs = f2NFTs
- NFTs

Expressiveness:
- sequential functions
- rational functions

Valuedness:

\[ \text{PTime}\ [\text{Choffrut77, WK95, BCPS03}] \]

\[ \text{PTime}\ [\text{Schützenberger75}, \text{GI83, BCPS03}] \]

\[ \text{decidable}\ [\text{CK87}] \]

\[ \text{undecidable}\ [\text{BGMP15}, \text{FGRS, LICS'13}] \]
Classes of Transductions

valuedness

expressiveness

DFTs

\( \subset \)

fNFTs

\( \subset \)

2DFTs = f2NFTs

NFTs

\( \subset \)

2NFTs

\( \cup \)

sequential functions

rational functions

regular functions

PTime [Choffrut77, WK95, BCPS03]

PTime [Schützenberger75, GI83, BCPS03]

decidable

[CK87]

decidable

undecidable

[BGMP15]

undecidable

[FGRS, LICS'13]
Classes of Transductions

- **DFTs** ⊂ **NFTs** ⊂ **2NFTs**
- **fNFTs** ⊂ **2DFTs = f2NFTs**

- **sequential functions** ⊂ **rational functions** ⊂ **regular functions**

- **PTime** [Choffrut77, WK95, BCPS03]

- Valuedness:
  - **expressiveness**

- Decidability:
  - Decidable
  - Undecidable

- Contextual references:
  - [CK87]
  - [Schützenberger75]
  - [GI83, BCPS03]
  - [BGMP15]
Classes of Transductions

- DFTs ⊂ NFTs ⊂ 2NFTs
- 2DFTs = f2NFTs

Valuedness:
- DFTs ⊂ fNFTs
- PTIME

Expressiveness:
- sequential functions
- rational functions
- regular functions

PTIME:
- [Schützenberger75]
- [GI83,BCPS03]

Decidability:
- decidable
- undecidable

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

- **DFTs** ⊂ **fNFTs** ⊂ **NFTs** ⊂ **2NFTs** ⊂ **2DFTs = f2NFTs**

- valuedness
  - sequential functions
  - rational functions
  - regular functions

- expressiveness
  - PTIME

- PTIME decidable
  - [Choffrut77, WK95, BCPS03]
  - [Schützenberger75]
  - [GI83, BCPS03]

- decidable
  - [CK87]
  - [BGMP15]

- undecidable
  - [FGRS, LICS'13]
Classes of Transductions

- **DFTs** ⊂ **fNFTs** ⊂ **NFTs** ⊂ **2NFTs**
- **PTime** ⊕ **undecidable** [BGMP15]
- **sequential functions** ⊂ **rational functions** ⊂ **regular functions**

---

- **PTime** [Choffrut77, WK95, BCPS03]
- **decidable** [Schützenberger75, GI83, BCPS03]
- **decidable** [CK87]
- **undecidable** [BGMP15, FGRS, LICS'13]
Classes of Transductions

- **valuedness**
- **expressiveness**

- **sequential functions**
- **rational functions**
- **regular functions**

- **DFTs** \(\subset\) **fNFTs** \(\subset\) **2DFTs=f2NFTs**
- **NFTs** \(\subset\) **2NFTs**

- **PTIME** \(\cup\)**

- **decidable**
- **undecidable**

- **PTime** \[Choffrut77,WK95,BCPS03\]
- **PTime** \[Schützenberger75\]
- **PTime** \[GI83,BCPS03\]
- **PTime** \[CK87\]
- **PTime** \[BGMP15\]
- **PTime** \[FGRS, LICS'13\]
Streaming String Transducers [AC10]

1-Way DFA + registers

Register updates:
- \( X := u \cdot Y \cdot v \)
- \( X := YZ \)

\( X, Y, Z \): registers
\( u, v \): words in \( \Sigma^* \)

\[ \vdash w \vdash \iff a\#^a(w) b\#^b(w) \]

Expressiveness results:
- sequential functions: 1-register right appending SST
- rational functions: right appending SST
- regular functions: copyless SST (\( X, Y := (X, X) \) is forbidden)
Streaming String Transducers \[AC10\]

1-Way DFA + registers

Register updates:
- \( X := u \cdot Y \cdot v \)
- \( X := YZ \)

\( X, Y, Z \): registers

\( u, v \): words in \( \Sigma^* \)

Expressiveness results:
- **sequential functions**: 1-register right appending SST
  \( X := X.a \)

\[ \vdash w \vdash \iff a \#_a(w) b \#_b(w) \]
Streaming String Transducers [AC10]

1-Way DFA + registers

Register updates:

- $X := u \cdot Y \cdot v$
- $X := YZ$

$X, Y, Z$: registers
$u, v$: words in $\Sigma^*$

Expressiveness results:

- **sequential functions**: 1-register right appending SST
  
  $X := X.a$

- **rational functions**: right appending SST
  
  $X := Y.a$
Streaming String Transducers [AC10]

1-Way DFA + registers

Register updates:

- \( X := u \cdot Y \cdot v \)
- \( X := YZ \)

\( X, Y, Z \): registers

\( u, v \): words in \( \Sigma^* \)

Expressiveness results:

- **sequential functions**: 1-register right appending SST
  \[
  X := X.a
  \]

- **rational functions**: right appending SST
  \[
  X := Y.a
  \]

- **regular functions**: copyless SST
  \[
  (X,Y) := (X,X) \text{ is forbidden}
  \]
Overview

1. Introduction

2. Models of transducers

3. Decidability results (based on patterns)

4. Connections with Logic and Algebra

5. Nested words

6. Conclusion
Sequentiality Problem

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

Standard technique:

- subset construction starting from the set of initial states.
- output longest common prefix
- store the unproduced outputs in the state

States of the form \{(p, a), (q, \varepsilon), (s, bb)\}
An example

\[ \text{dom}(f) = \Sigma^3 \]
\[ f(u) = \text{last}(u)^{|u|} \]
An example

\[ \text{dom}(f) = \sum^3 \]
\[ f(u) = \text{last}(u)|u| \]
An example

\[ \text{dom}(f) = \Sigma^3 \]
\[ f(u) = \text{last}(u)|u| \]

Goal: characterize termination of subset construction
Twinning Property [Choffrut77]

We 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:

\[ \text{we have delay}(v_1, w_1) = \text{delay}(v_1 v_2, w_1 w_2) \]
**Twinning Property** [Choffrut77]

We 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:

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

**Lemma**

*If a fNFT satisfies the Twinning Property, then the delays computed by the subset construction are bounded.*

**Corollary**

*Twinning Property \iff Termination of subset construction.*

**Theorem ([WK95])**

*Twinning Property can be decided in PTime.*
A counter example

\[ \text{dom}(f) = \Sigma^* \]
\[ f(u) = \text{last}(u)|u| \]

After reading an input word \( u \):
- longest common prefix of outputs = \( \varepsilon \)
- subset construction = \( \{(i_1, a|u|), (i_2, b|u|)\} \)

\( \Rightarrow \) The subset construction does not terminate.

The TP is violated: consider synchronised loops around \( i_1 \) and \( i_2 \).
Using right-appending streaming string transducers

$$\text{dom}(f) = \Sigma^*$$
$$f(u) = \text{last}(u) |u|$$

$$\text{upd: } \begin{cases} X_a := X_a.a \\ X_b := X_b.b \end{cases}$$

⇒ can be realized with 2 registers

Can we do better?
Using right-appending streaming string transducers

\[ \text{dom}(f) = \Sigma^* \]
\[ f(u) = \text{last}(u)|u| \]

\[ \text{upd:} \quad \begin{cases} 
  X_a \triangleq X_a.a \\
  X_b \triangleq X_b.b 
\end{cases} \]

\( \rightarrow \) can be realized with 2 registers

Can we do better?
No! 1 register is DFT

Register Complexity Problem

Input: A right-appending SST \( T \)
Question: Minimal \( k \) s.t. there exists a \( k \)-raSST \( T' \) with \( T \equiv T' \)
Register complexity using Twinning Property \cite{LICS16}

Intuition:
2 registers needed if there are 2 runs generating arbitrarily large delays

$k$ registers needed if there are $k$ runs generating pairwise arb. large delays
Register complexity using Twinning Property [LICS'16]

Intuition:
2 registers needed if there are 2 runs generating arbitrarily large delays

\( k \) registers needed if there are \( k \) runs generating pairwise arb. large delays

Contraposition: Twinning Property of order \( k \)

\( k \) synchronised loops

\( k + 1 \) runs

→ there are two runs that remain "close"
Lemma

If $T$ satisfies the TP of order $k$, then from any set of runs on the same input word, one can extract $k$ runs such that every run is "close" from one of these $k$ runs.

Theorem

A fNFT is definable by a $k$-raSST iff it satisfies the TP of order $k$.

Theorem

Given a fNFT $T$ and $k$ (in unary), deciding whether $T$ satisfies the TP of order $k$ is PSpace-complete.
Register complexity using Twinning Property [LICS’16]

An example: how many registers for the following function?

\[
f : u_1 \# u_2 \mapsto \text{last}(u_1)^{|u_1|} \# \text{last}(u_2)^{|u_2|}
\]
Register complexity using Twinning Property \[\text{[LICS'16]}\]

An example: how many registers for the following function?

\[f : u_1 \# u_2 \mapsto \text{last}(u_1)_{u_1} \# \text{last}(u_2)_{u_2}\]

Only 2 registers!
Register complexity using Twinning Property \cite{LICS'16}

An example: how many registers for the following function?

\[ f : u_1 \# u_2 \mapsto \text{last}(u_1)^{|u_1|} \# \text{last}(u_2)^{|u_2|} \]

\[
\begin{align*}
X_a &:= X_b.\# \\
X_b &:= X_a.\#
\end{align*}
\]
Other patterns: finite valuedness [Weber90]

Definition
A NFT $T$ is finite valued iff $\exists k \mid \forall u \in \Sigma^*, \#[T](u) \leq k$.

Theorem
Equivalence of finite valued NFT is decidable.
Other patterns: finite valuedness [Weber90]

**Definition**

A NFT $T$ is **finite valued** iff $\exists k \mid \forall u \in \Sigma^*, \#[T](u) \leq k$.

**Theorem**

*Equivalence of finite valued NFT is decidable.*

**Theorem**

*An NFT $T$ is finite-valued iff it satisfies criteria $G_0$ and $G_1$. *

**Theorem**

*Finite valuedness is decidable in PTime.*
Other patterns: multi-sequential relations \[\text{[CS86,FJ15]}\]

**Definition**

A function/relation \( R \) is multi-sequential iff there exists a finite number of DFTs \( T_1, \ldots, T_n \) such that \( R = \bigcup_i \llbracket T_i \rrbracket \)

**Theorem**

An NFT \( T \) is multi-sequential iff it satisfies the fork property.

**Theorem**

Multi sequentiality is decidable in \( \mathsf{PTime} \).

\[
\begin{align*}
\text{we have } \text{delay}(u_1, u_2) = \text{delay}(u_1 v_1, u_2 v_2)
\end{align*}
\]
A perspective about multi-sequential functions/relations

Multi-sequential functions ≡ SST with updates $X := X.a$

Register complexity in this class ≡ Minimization of size of the union

→ Find the right Twinning Property?
A perspective about multi-sequential functions/relations

Multi-sequential functions \( \equiv \) SST with updates \( X := X.a \)

Register complexity in this class \( \equiv \) Minimization of size of the union

→ Find the right Twinning Property?

\[
f : u_1 \# u_2 \mapsto \\
\text{last}(u_1)|u_1| \# \text{last}(u_2)|u_2|
\]

Minimum=4
Overview

1. Introduction

2. Models of transducers

3. Decidability results (based on patterns)

4. Connections with Logic and Algebra

5. Nested words

6. Conclusion
Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists \text{ DFA } A \text{ s.t. } L = L(A)$
Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists$ \textbf{DFA} $A$ s.t. $L = L(A)$
- $\exists$ \textbf{2NFA} $A$ s.t. $L = L(A)$
Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists$ DFA $A$ s.t. $L = L(A)$
- $\exists$ 2NFA $A$ s.t. $L = L(A)$
- $\exists$ MSO $\varphi$ s.t. $L = L(\varphi)$

**MSO**

$\exists \varphi ::= p_a(x) \mid x < y \mid x \in X \mid \exists x. \varphi \mid \exists X. \varphi \mid \varphi \land \varphi \mid \neg \varphi$
Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists \text{DFA } A \text{ s.t. } L = L(A)$
- $\exists \text{2NFA } A \text{ s.t. } L = L(A)$
- $\exists \text{MSO } \varphi \text{ s.t. } L = L(\varphi)$
- the right congruence $\approx_L$ of $L$ has finite index

$\text{MSO } \exists \varphi ::= p_a(x) | x < y | x \in X | \exists x.\varphi | \exists X.\varphi | \varphi \land \varphi | \neg \varphi$

$u \approx_L v \iff \forall w, uw \in L \text{ iff } vw \in L$

$u^{-1}.L = \{v | uv \in L\}$
Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists$ DFA $A$ s.t. $L = L(A)$
- $\exists$ 2NFA $A$ s.t. $L = L(A)$
- $\exists$ MSO $\varphi$ s.t. $L = L(\varphi)$
- the right congruence $\approx_L$ of $L$ has finite index
- the syntactic congruence $\sim_L$ has finite index

**MSO** $\exists \varphi ::= p_a(x) \mid x < y \mid x \in X \mid \exists x. \varphi \mid \exists X. \varphi \mid \varphi \land \varphi \mid \neg \varphi$

$u \approx_L v \iff \forall w, uw \in L \iff vw \in L$

$u^{-1}.L = \{v \mid uv \in L\}$

$u \sim_L v \iff \forall w, w', wuw' \in L \iff wvw' \in L$

Syntactic monoid: $\Sigma^*/\sim_L$

It is isomorphic to the transition monoid of the minimal automaton of $L$. 
MSO Transductions (Courcelle)

- input string seen as the logical structure over \( \{\text{succ}, (\text{lab}_a)_{a \in \Sigma}\} \)
- output predicates defined with MSO formulas interpreted over the input structure
MSO Transductions (Courcelle)

- input string seen as the logical structure over \( \{ \text{succ}, (\text{lab}_a)_{a \in \Sigma} \} \)
- output predicates defined with MSO formulas interpreted over the input structure
input string seen as the logical structure over \( \{ \text{succ}, (\text{lab}_a)_{a \in \Sigma} \} \)

output predicates defined with MSO formulas interpreted over the input structure

\[
\phi_{\text{succ}}(x, y) \equiv \text{succ}(y, x) \\
\phi_{\text{lab}_a}(x) \equiv \text{lab}_a(x)
\]
MSO Transductions (Courcelle)

- input string seen as the logical structure over \( \{ \text{succ}, (\text{lab}_a)_{a \in \Sigma} \} \)
- output predicates defined with MSO formulas interpreted over the input structure

\[
\phi_{\text{succ}}(x, y) \equiv \text{succ}(y, x) \\
\phi_{\text{lab}_a}(x) \equiv \text{lab}_a(x)
\]
**MSO Transductions (Courcelle)**

- input string seen as the logical structure over \( \{ \text{succ}, (\text{lab}_a)_{a \in \Sigma} \} \)
- output predicates defined with MSO formulas interpreted over the input structure

\[
\phi_{\text{succ}}(x, y) \equiv \text{succ}(y, x)
\]
\[
\phi_{\text{lab}_a}(x) \equiv \text{lab}_a(x)
\]
A Buchi theorem for transductions

Theorem ([Engelfriet, Hoogeboom, 01])

2DFT and MSOT are (effectively) expressively equivalent.

Emptiness

Given an MSOT $\phi$, does $J\phi K = \emptyset$ hold?

Transform $\phi$ into a 2DFT $T$

Check whether $\text{dom}(T) = \emptyset$

Equivalence

Given two MSOT $\phi_1, \phi_2$, does $J\phi_1 K = J\phi_2 K$ hold?

Transform $\phi_1, \phi_2$ into 2DFT $T_1, T_2$

Check equivalence of $T_1$ and $T_2$
A Buchi theorem for transductions

Theorem ([Engelfriet, Hoogeboom, 01])

2DFT and MSOT are (effectively) expressively equivalent.

Emptiness

- Given an MSOT \( \phi \), does \([\phi] = \emptyset\) hold?
  1. Transform \( \phi \) into a 2DFT \( T \)
  2. Check whether \( \text{dom}(T) = \emptyset \)
A Buchi theorem for transductions

**Theorem ([Engelfriet, Hoogeboom, 01])**

2DFT and MSOT are (effectively) expressively equivalent.

**Emptiness**

- Given an MSOT $\phi$, does $\llbracket \phi \rrbracket = \emptyset$ hold?
  1. Transform $\phi$ into a 2DFT $T$
  2. Check whether $\text{dom}(T) = \emptyset$

**Equivalence**

- Given two MSOT $\phi_1, \phi_2$, does $\llbracket \phi_1 \rrbracket = \llbracket \phi_2 \rrbracket$ hold?
  1. Transform $\phi_1, \phi_2$ into 2DFT $T_1, T_2$
  2. Check equivalence of $T_1$ and $T_2$
Order-preserving MSOT

- no backward edges in the MSO graph.
- e.g. reverse is not order-preserving
- syntactic presentation: guard formula $\phi_{\text{succ}}(x, y)$ by $x \leq y$. 

**Theorem** (Bojanczyk14, Filiot15)
Rational functions coincide with order-preserving MSO transductions.

As a corollary of 1-Way definability of 2-Way transducers, we have:

**Theorem**
It is decidable, given an MSOT $\phi$, whether $\phi$ is equivalent to some order-preserving MSOT.
Order-preserving MSOT

- no backward edges in the MSO graph.
- e.g. reverse is not order-preserving
- syntactic presentation: guard formula $\phi_{\text{succ}}(x, y)$ by $x \leq y$.

Theorem ([Bojanczyk14,Filiot15])

Rational functions coincide with order-preserving MSO transductions.

As a corollary of 1-Way definability of 2-Way transducers, we have:

Theorem

It is decidable, given an MSOT $\phi$, whether $\phi$ is equivalent to some order-preserving MSOT.
Sequential functions [Choffrut 03]

Key: consider earliest transducers (produce output asap)

What can be produced after reading input \( u \)?

\[
\hat{f}(u) = \text{lcp} \{ f(uv) \mid uv \in \text{dom}(f) \}
\]
Sequential functions [Choffrut 03]

Key: consider earliest transducers (produce output asap)

What can be produced after reading input $u$?

$$\hat{f}(u) = \text{lcp}\{f(uv) \mid uv \in \text{dom}(f)\}$$

Define $u \approx_f v \iff \begin{cases} u^{-1}.\text{dom}(f) = v^{-1}.\text{dom}(f) \\ \forall w, \hat{f}(u)^{-1}.f(uw) = \hat{f}(v)^{-1}.f(vw) \end{cases}$

$\rightarrow$ same behaviors after $u$ and $v$

Theorem

$f : \Sigma^* \rightarrow \Sigma^*$ is sequential iff $\approx_f$ has finite index.
Sequential functions [Choffrut 03]

Key: consider earliest transducers (produce output asap)

What can be produced after reading input $u$?

$$\hat{f}(u) = \text{lcp}\{f(uv) \mid uv \in \text{dom}(f)\}$$

Define $u \approx_f v \iff \begin{cases} u^{-1}.\text{dom}(f) = v^{-1}.\text{dom}(f) \\ \forall w, \hat{f}(u)^{-1}.f(uw) = \hat{f}(v)^{-1}.f(vw) \end{cases}$

$\Rightarrow$ same behaviors after $u$ and $v$

**Theorem**

$f : \Sigma^* \rightarrow \Sigma^*$ is sequential iff $\approx_f$ has finite index.

Classes of $\approx_f$ can be used to define a minimal DFT:

$$[u] \approx_f \xrightarrow{a|v} [ua] \approx_f \text{ where } v = \hat{f}(u)^{-1}.\hat{f}(ua)$$
Rational functions [Reutenauer, Schutzenberger 01]

Function $f : u \rightarrow \text{last}(u)^{|u|}$ is not sequential

→ needs some information about the suffix of the input word (look-ahead)

- may be infinite: $f : u_1 \# u_2 \# u_3 \ldots \mapsto \text{last}(u_1)^{|u_1|} \# \text{last}(u_2)^{|u_2|} \# \ldots$
- but not arbitrary: mirror image
Rational functions [Reutenauer, Schutzenberger 01]

Function $f : u \to \text{last}(u)^{|u|}$ is not sequential
→ needs some information about the suffix of the input word (look-ahead)
- may be infinite: $f : u_1 \# u_2 \# u_3 \ldots \mapsto \text{last}(u_1)^{|u_1|} \# \text{last}(u_2)^{|u_2|} \# \ldots$
- but not arbitrary: mirror image

Original left congruence: $u \overset{f}{\leftarrow} v \iff \text{dom}(f).u^{-1} = \text{dom}(f).v^{-1}$
- Functions $w \mapsto f(wu)$ and $w \mapsto f(wv)$ are "adjacent"

Adjacent: $\sup_w |\text{delay}(f(wu), f(wv))| < +\infty$
Rational functions [Reutenauer, Schutzenberger 01]

Function $f : u \rightarrow \text{last}(u)^{|u|}$ is not sequential

→ needs some information about the suffix of the input word (look-ahead)

- may be infinite: $f : u_1 \# u_2 \# u_3 \ldots \mapsto \text{last}(u_1)^{|u_1|} \# \text{last}(u_2)^{|u_2|} \# \ldots$

- but not arbitrary: mirror image

Original left congruence: $u \leftrightarrow^f v$ iff

- $\text{dom}(f).u^{-1} = \text{dom}(f).v^{-1}$

- Functions $w \mapsto f(wu)$ and $w \mapsto f(wv)$ are ”adjacent”

**Theorem**

$f : \Sigma^* \rightarrow \Sigma^*$ is rational iff $\leftrightarrow^f$ has finite index.
Rational functions [Reutenauer, Schutzenberger 01]

Function $f : u \rightarrow \text{last}(u)^{|u|}$ is not sequential
→ needs some information about the suffix of the input word (look-ahead)
- may be infinite: $f : u_1 \# u_2 \# u_3 \ldots \mapsto \text{last}(u_1)^{|u_1|} \# \text{last}(u_2)^{|u_2|} \# \ldots$
- but not arbitrary: mirror image

Original left congruence: $u \leftarrow_f v$ iff
- $\text{dom}(f).u^{-1} = \text{dom}(f).v^{-1}$
- Functions $w \mapsto f(wu)$ and $w \mapsto f(wv)$ are ”adjacent”

**Theorem**

$f : \Sigma^* \rightarrow \Sigma^*$ is rational iff $\leftarrow_f$ has finite index.

In addition, definition of a canonical transducer.
Rational functions: an example

Function $f: u \rightarrow \text{last}(u)^{|u|}$ with $\Sigma = \{a, b\}$

Classes of $\leftarrow f$: $[\varepsilon] = \{\varepsilon\}$ $[a] = \Sigma^*a$ $[b] = \Sigma^*b$

Indeed, we have $xa \leftarrow f ya$:

$$f(wxa) = a^{|w|+|x|+1} \quad \text{and} \quad f(wya) = a^{|w|+|y|+1}$$

→ delay between $f(wxa)$ and $f(wya)$ does not depend on $w$
Rational functions: an example

Function \( f : u \rightarrow \text{last}(u)^{|u|} \) with \( \Sigma = \{a, b\} \)

Classes of \( \leftarrow f \): \( [\varepsilon] = \{\varepsilon\} \) \( [a] = \Sigma^* a \) \( [b] = \Sigma^* b \)

Indeed, we have \( xa \leftarrow f ya \):

\[
f(wxa) = a^{\left|w\right|+\left|x\right|+1} \quad \text{and} \quad f(wya) = a^{\left|w\right|+\left|y\right|+1}
\]

\( \Rightarrow \) delay between \( f(wxa) \) and \( f(wya) \) does not depend on \( w \)

The fNFT guesses at each step the class (for \( \leftarrow f \)) of the suffix
Regular functions

No algebraic presentation exists (yet).

But one can define the transition monoid of a transducer.
Regular functions

No algebraic presentation exists (yet).

But one can define the transition monoid of a transducer. For a two-way transducer $T$, we define left-to-left runs of $T$: $Runs_{LL}(u) =$ pairs $(p, q)$ such that:

Similarly for left-to-right, right-to-left...

Congruence of the transition monoid:
Define $u \sim_T v \iff \forall x, y \in \{L, R\}, Runs_{xy}(u) = Runs_{xy}(v)$
Regular functions

No algebraic presentation exists (yet).

But one can define the transition monoid of a transducer. For a two-way transducer \( T \), we define left-to-left runs of \( T \): \( \text{Runs}_{LL}(u) \) = pairs \( (p, q) \) such that:

Similarly for left-to-right, right-to-left...

Congruence of the transition monoid:
Define \( u \sim_T v \iff \forall x, y \in \{L, R\}, \text{Runs}_{xy}(u) = \text{Runs}_{xy}(v) \)

Does not depend on the outputs!

Can also be defined for SST (taking into account the register updates)
First-Order definable transformations

Theorem (First-Order definable languages)

Given a language $L \subseteq \Sigma^*$, the following are equivalent:

- $L$ is definable in $\text{FO}(\langle, \{p_a, a \in \Sigma\})$
- $L$ is accepted by an automaton whose transition monoid is aperiodic
- $L$ is star-free
- $L$ is definable in LTL (linear temporal logic)

Aperiodic monoid $M$: $\exists n \mid \forall m \in M, n^n = m^{n+1}$
First-Order definable transformations

**Theorem ([FGL16])**

Let $T$ be a fNFT. Then $\mathbb{T}$ is order-pres. FOT definable iff its canonical transducer is aperiodic.

**Corollary**

Order-preserving FOT is decidable among rational functions.
First-Order definable transformations

**Theorem ([FGL16])**

Let $T$ be a fNFT. Then $\square T$ is order-pres. FOT definable iff its canonical transducer is aperiodic.

**Corollary**

Order-preserving FOT is decidable among rational functions.

**Theorem ([FKT14,CD15,DJR16])**

Let $f : \Sigma^* \rightarrow \Sigma^*$. The following are equivalent:

- $f$ is FOT-definable
- $f$ is definable by an aperiodic 2DFT
- $f$ is definable by an aperiodic copyless SST
Overview

1 Introduction

2 Models of transducers

3 Decidability results (based on patterns)

4 Connections with Logic and Algebra

5 Nested words

6 Conclusion
Motivation: Streaming XML Transformations

XML document ≡ tree linearization

[Diagram showing an XML document and its linearization process]
Motivation: Streaming XML Transformations

XML document $\equiv$ tree linearization

XML documents are words with a nesting structure:

**Structured alphabet:** call symbols, return symbols, internal symbols
Visibly Pushdown Automata (VPAs) \[\text{[AM04]}\]

VPAs = Pushdown Automata on \textit{structured} alphabet $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$:

- \textbf{push one} stack symbol on \textit{call} symbols $\Sigma_c$
- \textbf{pop one} stack symbol on \textit{return} symbols $\Sigma_r$
- \textbf{don’t touch} the stack on \textit{internal} symbols $\Sigma_i$
Visibly Pushdown Automata (VPAs) \cite{AM04}

VPAs = Pushdown Automata on \textit{structured} alphabet $\Sigma = \Sigma_c \uplus \Sigma_r \uplus \Sigma_i$:

- \textbf{push one} stack symbol on \textit{call} symbols $\Sigma_c$
- \textbf{pop one} stack symbol on \textit{return} symbols $\Sigma_r$
- \textbf{don’t touch} the stack on \textit{internal} symbols $\Sigma_i$

$L(A) = \{c_1 c^n i r^n r_1 \mid n \geq 0\}$

- closure properties
- decidability properties
- determinizable
Visibly Pushdown Transducers (VPTs) [RS08], [MFCS'10] = associate output words in $\Delta^*$ with transitions of a VPA

\[
R(T) = \{(c_1 c^n i r^n r_1, a^n b^n) \mid n \geq 0\}
\]

- nested word-to-word
- ranges are CFL
- functional VPT
- deterministic VPT
Visibly Pushdown Transducers (VPTs) [RS08],[MFCS’10]

associate output words in $\Delta^*$ with transitions of a VPA

\[ R(T) = \{(c_1 c^n i r^n r_1, a^n b^n) \mid n \geq 0\} \]

- nested word-to-word
- ranges are CFL
- functional VPT
- deterministic VPT
- Positive results: functionality, $k$-valuedness, equivalence
- Negative results: composition, type-checking
- Open problem: determinizability, finite valuedness
Visibly Pushdown Transducers (VPTs) \cite{RS08, MFCS10}

- associate output words in $\Delta^*$ with transitions of a VPA

\[ R(T) = \{(c_1 c^n \ i \ r^n \ r_1, a^n b^n) \mid n \geq 0\} \]

- nested word-to-word
- ranges are CFL
- functional VPT
- deterministic VPT

Positive results: functionality, $k$-valuedness, equivalence

Negative results: composition, type-checking

Open problem: determinizability, finite valuedness

Well-nested output words: composition and type-checking \cite{DLT14}
Streamability Problem for VPT [FSTTCS’11]

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT $T$.

How much memory is needed to compute $T(u)$ from an input stream $u$?
Streamability Problem for VPT [FSTTCS’11]

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT $T$.
How much memory is needed to compute $T(u)$ from an input stream $u$?

- bounded memory
- $\text{length}(u)$
- cannot check well-nestedness
- not streamable

Using a twinning property, we prove:
Decidable in $\text{Co-NPTime}$
But:
memory may depend exponentially in $\text{height}(u)$
Streamability Problem for VPT \cite{FSTTCS'11}

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT $T$.

How much memory is needed to compute $T(u)$ from an input stream $u$?

- bounded memory
- $\text{height}(u)$
- $\text{length}(u)$

<table>
<thead>
<tr>
<th>cannot check</th>
<th>not</th>
</tr>
</thead>
<tbody>
<tr>
<td>well-nestedness</td>
<td>streamable</td>
</tr>
</tbody>
</table>

Using a twinning property, we prove: Decidable in Co-NPTime

But: memory may depend exponentially in $\text{height}(u)$
Streamability Problem for VPT [FSTTCS’11]

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT $T$.

How much memory is needed to compute $T(u)$ from an input stream $u$?

**Height Bounded Memory Problem**

**Input:** a transformation $T$ defined by a fVPT

**Output:** can $T$ be realized with height-bounded memory?

\[ \exists f : \mathbb{N} \rightarrow \mathbb{N} \cdot \forall u \in \text{dom}(T) \]

$T(u)$ can be computed with $f(\text{height}(u))$-bounded memory?
Streamability Problem for VPT \cite{FSTTCS'11}

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT $T$.

How much memory is needed to compute $T(u)$ from an input stream $u$?

**Height Bounded Memory Problem**

*Input:* a transformation $T$ defined by a fVPT

*Output:* can $T$ be realized with height-bounded memory?

\[
\exists f : \mathbb{N} \to \mathbb{N} \cdot \forall u \in \text{dom}(T) \quad T(u) \text{ can be computed with } f(\text{height}(u))-\text{bounded memory?}
\]

Using a twinning property, we prove:

**Decidable in Co-NPTime**

**But:** memory may depend exponentially in $\text{height}(u)$
Online Bounded Memory [FSTTCS’11]

**Online Bounded Memory Problem**

**Input:** a transformation $T$ defined by a fVPT

**Output:** can $T$ be realized with current-height-bounded memory?
Online Bounded Memory [FSTTCS’11]

Current height:

Online Bounded Memory Problem

Input: a transformation $T$ defined by a fVPT
Output: can $T$ be realized with current-height-bounded memory?

Characterization: Matched Twinning Property

- ensures quadratic dependence in the current height
- Decidable in Co-NPTIME
Logical characterization for VPTs [LICS'16]

**Theorem**

*MSO definable nested word-to-word transformations are expressively equivalent to $2DVPT_{su}$.*

- Two-way VPTs (stack operation exchanged when reading backwards)
- $su$: single-use

**Key result:** $2DVPT^{LA} \equiv 2DVPT$ (LA: Look Around)
Logical characterization for VPTs [LICS’16]

**Theorem**

*MSO* definable nested word-to-word transformations are expressively equivalent to $2DVPT_{su}$.

- Two-way VPTs (stack operation exchanged when reading backwards)
- su : single-use

**Key result:** $2DVPT^{LA} \equiv 2DVPT$ (LA:Look Around)

**Bonus:**

**Theorem**

*Order-preserving* *MSO* definable nested word-to-word transformations are expressively equivalent to functional VPTs.
Summary for nested words transductions

DVPTs \subset\subset OBM \subset\subset HBM \subset\subset fVPTs

valuedness

expressiveness

Co-NP

open
Summary for nested words transductions

\[ \text{DVPTs} \subset \text{OBM} \subset \text{HBM} \subset \text{fVPTs} \]

\[ \text{VPTs} \]

\[ \text{finite-valued VPTs} \]

\[ \text{k-valued VPTs} \]

\[ \text{PTime} \]

\[ \text{Co-NP} \]

\[ \text{open} \]

Pierre-Alain Reynier (LIF, AMU & CNRS)
Summary for nested words transductions

\[ \text{DVPTs} \subset \text{OBM} \subset \text{HBM} \subset \text{fVPTs} \]

\[ \text{VPTs} \]

\[ \text{finite-valued VPTs} \]

\[ \text{k-valued VPTs} \]

\[ \text{PTime} \]

\[ \cup \quad \text{Co-NP} \]

\[ \text{open} \]

\[ \text{MSOT} = 2\text{DVPT}_{su}^{LA} \]

\[ \text{Co-NP} \]

\[ \text{open} \]

\[ \text{open} \]

\[ \text{expressiveness} \]

\[ \text{valuedness} \]
Overview

1. Introduction
2. Models of transducers
3. Decidability results (based on patterns)
4. Connections with Logic and Algebra
5. Nested words
6. Conclusion
Summary

- Decidability results based on patterns
  - Shift from NFT to raSST
  - Allows to derive efficient decision procedures

- Logic and Algebra connections
  - Minimization for sequential functions
  - Decidability of FOT for rational functions
  - Characterization only for regular functions

- Nested words
  - Most of the decidability results can be lifted
  - Logic can be lifted too
  - Algebra is missing
I did not present...

Some of these results are valid not only for transducers.

Recent results on model simplification:
- minimize the number of passes of a 2DFT
- minimize the number of registers of non-det. SST

Important results about relations:
- $k$-valuedness is decidable
- $k$-valued relations can be decomposed into functions

Recent trend about an alternative semantics (see next talk)
Perspectives

Shift from rational to regular functions
- register minimization
- algebraic presentation, canonical object

Specification languages for transformations

Alternative semantics to break undecidability/high complexity
Perspectives

Shift from rational to regular functions
- register minimization
- algebraic presentation, canonical object

Specification languages for transformations

Alternative semantics to break undecidability/high complexity

Thanks!