Two "new" twinning properties for transducers

Pierre-Alain Reynier

LIF, Aix-Marseille University & CNRS

Based on:

A Generalized twinning property for minimisation of cost register automata [LICS'16]
with Laure Daviaud and Jean-Marc Talbot

Degree of sequentiality of weighted automata [FoSSaCS'17]
with Laure Daviaud, Ismaël Jecker and Didier Villevalois
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
- ...

Overview

1. Models of transducers

2. What can we do with $k$ registers?

3. Independent registers

4. Conclusion
Overview

1. Models of transducers

2. What can we do with $k$ registers?

3. Independent registers

4. Conclusion
Finite state transducers

= associate output words with transitions of a finite state automaton

Example (A transducer $T$)

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

Non-determinism: semantics may be a relation
Finite state transducers

associate output words with transitions of a finite state automaton

Example (A transducer $T$)

Semantics $[T] = \text{LAST} : \; w\sigma \mapsto \sigma|w|+1$, with $\sigma \in \{a, b\}$, $w \in \{a, b\}^+$

Non-determinism: semantics may be a relation

A transducer is:

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

Classes: DFT, fNFT, NFT
Streaming String Transducers (right-appending) [AC10]

1-Way DFA + registers

Register updates:
  \[ X := Y \cdot u \]

\( X, Y \): registers
\( u \): output word

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

\( X_a, X_b \)

\( \Rightarrow \) realizes \text{LAST}
Streaming String Transducers (right-appending) [AC10]

1-Way DFA + registers

Register updates:

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

\( X, Y \): registers
\( u \): output word

\[ \uparrow : \ X_a := X_a a \quad X_b := X_b b \]

→ realizes \( \text{LAST} \)

---

**Theorem**

\( fNFT \equiv \text{right-appending SST} \)
\( DFT \equiv \text{right-appending SST with 1 register} \)
fNFT and right-appending SST

\[ fNFT \equiv \text{right-appending SST} \]

\( \subseteq \): determinization, one register per state of the fNFT. \( X_q \) stores the output produced along any run from an initial state to \( q \)

\( \supseteq \): guess which register will be produced
**Theorem**

\[ fNFT \equiv \text{right-appending SST} \]

\( \subseteq \): determinization, one register per state of the fNFT. \( X_q \) stores the output produced along any run from an initial state to \( q \).

\( \supseteq \): guess which register will be produced.

**Register Complexity Problem for fNFT**

Input: A fNFT \( T \)

Question: Minimal \( k \) s.t. there exists a \( k \)-raSST \( T' \) with \( T \equiv T' \)
Overview

1. Models of transducers

2. What can we do with $k$ registers?

3. Independent registers

4. Conclusion
From rational to sequential functions [Choffrut77]

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

**Issue:** termination (bound the size of unproduced outputs)
An example

\textbf{LAST} on $\Sigma^3$
An example

LAST on $\Sigma^3$

Pierre-Alain Reynier (LIF, AMU & CNRS) Two new twinning properties for transducers Kickoff ANR Delta
Twinning Property [Choffrut77]

We define:

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

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

For all situations like:

we have \(\text{delay}(w_0, w_1) = \text{delay}(w_0 w_0', w_1 w_1')\)
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:

\[ \text{we have delay}(w_0, w_1) = \text{delay}(w_0w'_0, w_1w'_1) \]

Lemma

\[ T \models \text{Twinning Property} \iff \text{Termination of subset construction} \]

\[ T \models \text{Twinning Property} \implies \forall (p, x) \in \text{subset constr.}, |x| \leq n^2 M \]
Twinning Property [Choffrut77]

We define:

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

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

For all situations like:

\[
\begin{array}{c}
\text{we have delay}(w_0, w_1) = \text{delay}(w_0w'_0, w_1w'_1)
\end{array}
\]

Lemma

\[
T \models \text{Twinning Property} \iff \text{Termination of subset construction}
\]

\[
T \models \text{Twinning Property} \implies \forall (p, x) \in \text{subset constr.}, |x| \leq n^2 M
\]

Theorem ([WK95])

\[
\text{Twinning Property can be decided in PTime.}
\]
Register complexity using Twinning Property

Twinning property characterizes the fact that runs (on the same input) remain close.

Intuition:
Need for 2 reg. if there are 2 runs with arbitrarily large delays

Need for $k + 1$ reg. if there are $k + 1$ runs with pairwise arb. large delays
Twinning Property of order $k$

For all situations like:

$k$ synchronised loops

\[ u_1 | w_{1,0} \rightarrow v_1 | w_{1,0} \rightarrow u_2 | w_{2,0} \rightarrow v_2 | w_{2,0} \rightarrow v_k | w_{k,0} \]

$k + 1$ runs

\[ u_1 | w_{1,1} \rightarrow v_1 | w_{1,1} \rightarrow u_2 | w_{2,1} \rightarrow v_2 | w_{2,1} \rightarrow v_k | w_{k,1} \]

\[ u_1 | w_{1,k} \rightarrow v_1 | w_{1,k} \rightarrow u_2 | w_{2,k} \rightarrow v_2 | w_{2,k} \rightarrow v_k | w_{k,k} \]

there are two runs $0 \leq i < j \leq k$ s.t. for every loop $\ell$,

we have $\text{delay}(w_{1,i} \ldots w_{\ell,i}, w_{1,j} \ldots w_{\ell,j}) = \text{delay}(w_{1,i} \ldots w_{\ell,i} w'_{\ell,i}, w_{1,j} \ldots w_{\ell,j} w'_{\ell,j})$
Register complexity using Twinning Property

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.

"close": $(p, x)$ with $|x| \leq n^{k+1} M$

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.
Example

How many registers for the following function?

\[ \text{LAST}^2 : u_1 \# u_2 \mapsto \text{LAST}(u_1) \# \text{LAST}(u_2) \]
Example

How many registers for the following function?

\[
\text{LAST}^2 : u_1 \# u_2 \mapsto \text{LAST}(u_1) \# \text{LAST}(u_2)
\]

Only 2 registers!
Example

How many registers for the following function?

\[ \text{LAST}^2 : u_1 \# u_2 \mapsto \text{LAST}(u_1) \# \text{LAST}(u_2) \]
Overview

1. Models of transducers

2. What can we do with $k$ registers?

3. Independent registers

4. Conclusion
Independent registers

Only updates $X := Xu$: "No communication between threads"
Independent registers

Only updates $X := Xu$: "No communication between threads"

$$\text{LAST}^2 : u_1 \# u_2 \mapsto \text{LAST}(u_1) \# \text{LAST}(u_2)$$

→ Need 4 independent registers
Branching twinning property of order $k$

For all situations like:

$k$ not synchronised loops

there are two runs $0 \leq i < j \leq k$ 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 $k$-raSST with independent registers iff it satisfies the BTP of order $k$. This is decidable in Pspace ($k$ in unary).

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 $S = S' \sqcup S''$, 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 $k' + k'' \leq k$ apply induction hypothesis
Branching twinning property of order $k$

**Theorem**

A fNFT is definable by a $k$-raSST with independent registers iff it satisfies the BTP of order $k$. This is decidable in Pspace ($k$ in unary).

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 $k$-raSST with independent registers iff it satisfies the BTP of order $k$. This is decidable in Pspace ($k$ in unary).

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 $S' \uplus S''$, 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
Application to multi-sequential functions

**Definition ( [CS86] )**

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

**Examples:**

- \( \text{LAST} \) and \( \text{LAST}^2 \) are multi-sequential
- \( \text{LAST}^* : u_1 \# \ldots \# u_n \mapsto \text{LAST}(u_1)\# \ldots \# \text{LAST}(u_n) \) is not multi-seq.
Application to multi-sequential functions

Definition ([CS86])

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

Observations:

- Multi-sequential functions $\equiv$ raSST with independent registers
- size of the union $=$ number of registers

→ Register minimization in this class $\equiv$ Minimization of size of the union
Application to multi-sequential functions

Definition ( [CS86])
Multi-sequential functions are defined as functions that can be realized as finite union of sequential transducers.

Observations:
- Multi-sequential functions $\equiv$ raSST with independent registers
- size of the union $=$ number of registers

$\Rightarrow$ Register minimization in this class $\equiv$ Minimization of size of the union

Corollary
Given a fNFT $T$ and $k$ (in unary), deciding whether there exist $T_1, \ldots, T_k \in DFT$ s.t. $T \equiv \bigcup_{i=1}^{k} T_i$ is PSpace-complete.
Overview

1. Models of transducers
2. What can we do with $k$ registers?
3. Independent registers
4. Conclusion
Alternative characterizations:

- bounded variation property (for $k$ registers)
- Lipschitz property (for $k$ independent registers)
I did not present...

Alternative characterizations:
- bounded variation property (for $k$ registers)
- Lipschitz property (for $k$ independent registers)

Functional $\sim$ finite-valued

Extension of the SST model: final states are mapped to sets of registers
Alternative characterizations:

- bounded variation property (for $k$ registers)
- Lipschitz property (for $k$ independent registers)

Functional $\sim$ finite-valued

Extension of the SST model: final states are mapped to sets of registers

Weighted automata on semigroups, with:

- set semantics
- infinitary semigroup ($\alpha \beta \gamma \neq \beta \implies |\{\alpha^n \beta \gamma^n | n \in \mathbb{N}\}| = +\infty$)
- finitely generated semigroup
Perspectives

Shift from rational to regular functions
➡ deal with concatenation of registers
➡ compare runs of two-way machines

Algebraic characterizations

From set semantics to other operations

Standard extensions (infinite words, nested words…)}
Perspectives

Shift from rational to regular functions
- deal with concatenation of registers
- compare runs of two-way machines

Algebraic characterizations

From set semantics to other operations

Standard extensions (infinite words, nested words...)

Thanks!
Alternative characterizations

\[ f : \Sigma^* \to \Gamma^* \]

<table>
<thead>
<tr>
<th></th>
<th>bounded variation</th>
<th>Lipschitz property</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DFT</strong></td>
<td>( \forall n \exists N \forall u, v \in \text{dom}(f), ) ( d(u, v) \leq n \Rightarrow d(f(u), f(v)) \leq N )</td>
<td>( \exists L \forall u, v \in \text{dom}(f), ) ( d(f(u), f(v)) \leq L.(d(u, v) + 1) )</td>
</tr>
<tr>
<td><strong>k registers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>k independent registers</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Alternative characterizations

\[ f : \Sigma^* \rightarrow \Gamma^* \]

<table>
<thead>
<tr>
<th></th>
<th>bounded variation</th>
<th>Lipschitz property</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DFT</strong></td>
<td>( \forall n \exists N \forall u, v \in \text{dom}(f), ) ( d(u, v) \leq n \Rightarrow d(f(u), f(v)) \leq N )</td>
<td>( \exists L \forall u, v \in \text{dom}(f), ) ( d(f(u), f(v)) \leq L.(d(u, v) + 1) )</td>
</tr>
<tr>
<td>( k ) registers</td>
<td>( \forall n \exists N \forall u_0 \ldots u_k \in \text{dom}(f), ) ( (\forall i \neq j, d(u_i, u_j) \leq n) \Rightarrow \exists i \neq j.d(f(u_i), f(u_j)) \leq N )</td>
<td>?</td>
</tr>
<tr>
<td>( k ) independent registers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Alternative characterizations

\[ f : \Sigma^* \rightarrow \Gamma^* \]

<table>
<thead>
<tr>
<th></th>
<th>bounded variation</th>
<th>Lipschitz property</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DFT</strong></td>
<td>( \forall n \exists N \forall u, v \in \text{dom}(f), ) (d(u, v) \leq n \Rightarrow d(f(u), f(v)) \leq N)</td>
<td>( \exists L \forall u, v \in \text{dom}(f), ) (d(f(u), f(v)) \leq L(d(u, v) + 1))</td>
</tr>
<tr>
<td><strong>k registers</strong></td>
<td>( \forall n \exists N \forall u_0 \ldots u_k \in \text{dom}(f), ) [(\forall i \neq j, d(u_i, u_j) \leq n) \Rightarrow \exists i \neq j. d(f(u_i), f(u_j)) \leq N]</td>
<td>?</td>
</tr>
<tr>
<td><strong>k independent registers</strong></td>
<td>?</td>
<td>( \exists L \forall u_0 \ldots u_k \in \text{dom}(f), ) (\exists i \neq j \text{ s.t.} d(f(u_i), f(u_j)) \leq L(d(u_i, u_j) + 1))</td>
</tr>
</tbody>
</table>