Decomposition of finite-valued streaming string transducers
Transformations of objects, here \textit{words}

\textit{transduction} = function or relation between words
Transductions

Transformations of objects, here \textit{words}

\texttt{transduction} = function or relation between words

\begin{align*}
\text{hannover} & \rightarrow \{\text{hannover}\}^* \quad \text{Kleene iteration} \\
\text{hannover} & \rightarrow \text{revonnah} \quad \text{mirror} \\
\text{hannover} & \rightarrow \text{hannover hannover} \quad \text{duplicate} \\
\text{hannover} & \rightarrow \text{over hann} \quad \text{split \& swap}
\end{align*}
Transductions defined by formulas

**MSOT** = monadic second-order transductions \[\text{[Courcelle '95]}\]

Logically define the output inside copies of the input:

- **domain**: unary formula selecting positions in each copy
- **order**: binary formula defining an order on the domain
- **letters**: unary formulas partitioning the domain

\[
\varphi_{<}(x,y) = "x > y" \quad // \quad x < y \text{ in the output iff } x > y \text{ in the input}
\]
### Finite-state Transducers

Finite-state Transducers = automata with outputs on transitions

<table>
<thead>
<tr>
<th>One-way</th>
<th>Deterministic</th>
<th>Non-deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1DFT</td>
<td></td>
<td>1NFT</td>
</tr>
<tr>
<td><a href="#">Diagram</a></td>
<td><a href="#">Diagram</a></td>
<td><a href="#">Diagram</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Two-way</th>
<th>Deterministic</th>
<th>Non-deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>2DFT</td>
<td></td>
<td>2NFT</td>
</tr>
<tr>
<td><a href="#">Diagram</a></td>
<td><a href="#">Diagram</a></td>
<td><a href="#">Diagram</a></td>
</tr>
</tbody>
</table>
Transductions defined by automata

**SST** = Streaming String Transducers

- deterministic / non-deterministic
- 1-way
- write-only registers to store partial outputs
  + copyless restriction = each register used at most once

E.g. split & swap

```
x = a
y = a
```

```
a | x := x.a
   |
   |
   | y := y.a
   |
   |
   out := y.x
```
Transductions defined by automata

\textbf{SST} = Streaming String Transducers \quad [\text{Alur, Cerny '10}]

- deterministic / non-deterministic
- 1-way
- write-only registers to store partial outputs
  + copyless restriction = each register used at most once

\[
\begin{align*}
x &= \\
y &= 
\end{align*}
\]

E.g. split & swap

\[
\begin{align*}
a \mid x &:= x.a \\
y &:= y & a \mid x &:= x \\
y &:= y.a & \text{out} &:= y.x
\end{align*}
\]
Transductions defined by automata

\[ \text{SST} = \text{Streaming String Transducers} \]

- deterministic / non-deterministic
- 1-way
- write-only registers to store partial outputs
  + copyless restriction = each register used at most once

\[ x = \]
\[ y = \]

E.g. split & swap

\[ a \mid x := x.a \]
\[ y := y \]

\[ a \mid x := x \]
\[ y := y.a \]

out := y.x

E.g. mirror

\[ a \mid x := a.x \]

out := x

[Alur, Cerny ’10]
Relational transductions

$1DFT \quad a \cdot w \mapsto w \cdot a$

$2DFT = DSST = MSOT \quad w \mapsto w \cdot w$

$1NFT \quad w \mapsto \Sigma |w|$

$2NFT \quad w \mapsto w^*$

$NSST = NMSOT \quad uv \mapsto vu$
Relational transductions

1DFT
\[ a \cdot w \mapsto w \cdot a \]

2DFT = DSST = MSOT
\[ w \mapsto w \cdot w \]

1NFT
\[ w \mapsto \sum_{|w|} \]

NSST = NMSOT
\[ uv \mapsto vu \]

decidable equivalence
undecidable equivalence

2NFT
\[ w \mapsto w^* \]
Relational transductions

1DFT
\[ a \cdot w \mapsto w \cdot a \]

2DFT = DSST = MSOT
\[ w \mapsto w \cdot w \]

1NFT
\[ w \cdot a \mapsto a \cdot w \]

NSST = NMSOT = 2NFT

decidable equivalence

undecidable equivalence
Relational transductions

\[1DFT \quad 2DFT = DSST = MSOT\]

\[1NFT \quad NSST = NMSOT = 2NFT\]

decidable equivalence

undecidable equivalence

Anything interesting beyond functional transductions?
Finite valuedness

\[ k \text{-valued transductions} = \text{at most } k \text{ outputs for each input} \]

- decidable equivalence?
- correspondence with logic (e.g. MSO) ?
- equivalent models (e.g. 2-way vs SSTs) ?
- effective characterisations (e.g. 1-way definability) ?
$k$-valued transductions $= \text{at most } k \text{ outputs for each input}$

- decidable equivalence?
- correspondence with logic (e.g. MSO)?
- equivalent models (e.g. 2-way vs SSTs)?
- effective characterisations (e.g. 1-way definability)?

a unifying approach: Decomposition Theorem

"Every $k$-valued transducer $\in \mathcal{C}$ can be decomposed into a finite union of functional transducers $\in \mathcal{C}$"
The decomposition theorem for $\mathcal{C} = \{ 1\text{NFTs} \}$:

Every $k$-valued 1NFT is a finite union of functional 1NFTs.

[Weber ’96, Sakarovitch - de Souza ‘08]
The decomposition theorem for $\mathcal{C} = \{ \text{1NFTs} \}$:

Every $k$-valued 1NFT is a finite union of functional 1NFTs.

[Weber ’96, Sakarovitch - de Souza ‘08]

Corollaries:

- decidable equivalence of $k$-valued 1NFTs
- $k$-valued 1NFTs = $k$-valued order-preserving MSO transductions
Decomposition of 1NFTs

Unboundedly many runs with same output…

Follow lexico.-least run for each output

Subproblem: compare runs by their outputs
In a 1NFT, outputs are formed by appending only to the right.
In a 1NFT, outputs are formed by appending only to the right.

\[
align(\rho, \rho') = \begin{cases} 
(u, \varepsilon) & : \text{out}(\rho) . u = \text{out}(\rho') \\
(\varepsilon, v) & : \text{out}(\rho) = \text{out}(\rho') . v
\end{cases}
\]
Decomposition of 1NFTs

In a 1NFT, outputs are formed by appending only to the right

\[
\rho \quad \text{align}(\rho, \rho') = u \quad \text{out} \quad \rho' \quad \text{lag}(\rho, \rho') = \lvert \text{align}(\rho, \rho') \rvert
\]
Decomposition of 1NFTs

In a 1NFT, outputs are formed by appending only to the right

\[ \text{align} (\rho, \rho') = \left\{ \begin{array}{l}
(u, \varepsilon) : \text{out} (\rho) \cdot u = \text{out} (\rho') \\
(\varepsilon, v) : \text{out} (\rho) = \text{out} (\rho') \cdot v
\end{array} \right\} 
\]

\[ \text{lag} (\rho, \rho') = | \text{align} (\rho, \rho') | \]

\[ \text{maxlag} (\rho, \rho') = \text{MAX} \{ \text{lag} (\rho_{\leq t}, \rho'_{\leq t}) : t \leq |\rho| \} \]
Decomposition of 1NFTs

It may happen that $out(\rho) = out(\rho')$ yet with large $maxlag(\rho, \rho')$

\[
\begin{align*}
\text{input} & = a a \ldots a a \ldots & \# & \ldots a a \ldots a a \\
out(\rho) & = a a \ldots a a \ldots \\
out(\rho') & = \ldots a a \ldots a a
\end{align*}
\]
Decomposition of 1NFTs

It may happen that $\text{out} (\rho) = \text{out} (\rho')$ yet with large $\text{maxlag} (\rho, \rho')$

\[
\begin{align*}
\text{input} &= \text{a a ... a a ...} \# \ldots \text{a a ... a a} \\
\text{out} (\rho) &= \text{a a ... a a ...} \\
\text{out} (\rho') &= \ldots \text{a a ... a a}
\end{align*}
\]
It may happen that $\text{out}(\rho) = \text{out}(\rho')$ yet with large $\text{maxlag}(\rho,\rho')$.

It is not thanks to $\text{out}(\rho) = \text{out}(\rho')$ that the transducer is $k$-valued!

\[
\begin{align*}
\text{input} & = \ a \ a \ \cdots \ a \ a \ \cdots \ # \ \cdots \ a \ a \ \cdots \ a \ a \\
\text{out}(\rho) & = \ a \ a \ \cdots \ a \ a \ \cdots \\
\text{out}(\rho') & = \ a \ a \ \cdots \ a \ a \ \cdots \ a \ a
\end{align*}
\]
Key combinatorial property:

\[ \rho_1, \ldots, \rho_{k+1} \text{ runs of } k\text{-valued 1NFT} \implies \exists i \neq j \text{ out}(\rho_i) = \text{out}(\rho_j) \land \text{maxlag}(\rho_i, \rho_j) \text{ small} \]
Decomposition of 1NFTs

**Key combinatorial property:**

\[ \rho_1, \ldots, \rho_{k+1} \text{ runs of } k\text{-valued 1NFT} \quad \Rightarrow \quad \exists \ i \neq j \quad \text{out}(\rho_i) = \text{out}(\rho_j) \quad \& \quad \text{maxlag}(\rho_i, \rho_j) \text{ small} \]

Moreover, if \( \text{maxlag}(\rho_i, \rho_j) \) is small one can maintain \( \text{align}(\rho_i, \rho_j) \) in bounded memory — in particular, one knows whether \( \text{out}(\rho_i) = \text{out}(\rho_j) \)
One can simulate only the witness runs, namely, the $\rho$’s that are

- successful

- lexico.-least among all other runs $\rho'$ with $\{\text{out}(\rho) = \text{out}(\rho') \& \text{maxlag}(\rho, \rho') \text{ small}\}$ at most $k$ witnesses!
Conjecture: every $k$-valued SST is a finite union of functional SSTs
Conjecture: every $k$-valued SST is a finite union of functional SSTs

Some corollaries:

- decidable equivalence of $k$-valued SSTs
  \[\text{Alur, Deshmukh '11}\]

- $k$-valued SSTs $= k$-valued MSO transductions $= k$-valued 2NFTs
  \[\text{Alur, Cerny '10, Engelfriet, Hoogeboom '01}\]

- effective characterisation of $k$-valued SSTs definable by $k$-valued 1NFTs
  \[\text{Filiot, Gauwin, Reynier, Servais '13}\]
Conjecture: every $k$-valued SST is a finite union of functional SSTs

Some corollaries:

- Decidable equivalence of $k$-valued SSTs
  - $T \subseteq T_1 \cup \ldots \cup T_k$ decidable for functional SSTs $T$, $T_1$, ..., $T_k$
  - [Alur, Deshmukh '11]

- $k$-valued SSTs = $k$-valued MSO transductions
  - = $k$-valued 2NFTs
  - [Alur, Cerny '10]
  - [Engelfriet, Hoogeboom '01]

- Effective characterisation of $k$-valued SSTs
  - definable by $k$-valued 1NFTs
  - [Filiot, Gauwin, Reynier, Servais '13]
Conjecture: every $k$-valued SST is a finite union of functional SSTs

Some corollaries:

- Decidable equivalence of $k$-valued SSTs

- $k$-valued SSTs $=$ $k$-valued MSO transductions $=$ $k$-valued 2NFTs

- Effective characterisation of $k$-valued SSTs definable by $k$-valued 1NFTs

$DSSTs = MSO = 2NFTs$ in the functional case

[Alur, Deshmukh ’11]

[Alur, Cerny ’10]

[Engelfriet, Hoogeboom ’01]

[Filiot, Gauwin, Reynier, Servais ’13]
Conjecture: every $k$-valued SST is a finite union of functional SSTs

Some corollaries:

- Decidable equivalence of $k$-valued SSTs
  - [Alur, Deshmukh ’11]

- $k$-valued SSTs $= k$-valued MSO transductions $= k$-valued 2NFTs
  - [Alur, Cerny ’10] [Engelfriet, Hoogeboom ’01]

- Effective characterisation of $k$-valued SSTs definable by $k$-valued 1NFTs
  - [Filiot, Gauwin, Reynier, Servais ’13]

Our contribution: we proved the conjecture for SSTs with 1 register
First difficulty: letters added to left and right of register ⇒ symmetric alignments on registers

\[
\text{align} (\rho, \rho') = \left\{ \lambda = (u, v, w, z) : u \cdot \text{reg}(\rho) \cdot v = w \cdot \text{reg}(\rho') \cdot z \right\}
\]
Decomposition of 1-register SSTs

First difficulty: letters added to left and right of register \( \Rightarrow \) symmetric alignments on registers

\[
\text{align} (\rho, \rho') = \left\{ \lambda = (u, v, w, z) : u \cdot \text{reg} (\rho) \cdot v = w \cdot \text{reg} (\rho') \cdot z \right\}
\]

\[
\text{lag} (\rho, \rho') = \text{MIN} \left\{ |\lambda| : \lambda \in \text{align} (\rho, \rho') \right\}
\]
Decomposition of 1-register SSTs

**First difficulty:** letters added to left and right of register \( \Rightarrow \) symmetric alignments on registers

\[
\begin{align*}
\rho & \quad \text{align} (\rho, \rho') = \left\{ \lambda = (u, v, w, z) : u \cdot \text{reg} (\rho) \cdot v = w \cdot \text{reg} (\rho') \cdot z \right\} \\
\rho' & \quad \text{lag} (\rho, \rho') = \text{MIN} \left\{ |\lambda| : \lambda \in \text{align} (\rho, \rho') \right\} \\
\maxlag (\rho, \rho') & = \text{MAX} \left\{ \text{lag} (\rho_{\leq t}, \rho'_{\leq t}) : t \leq |\rho| \right\}
\end{align*}
\]
Decomposition of 1-register SSTs

Second difficulty: combinatorial property

\[ k+1 \text{ runs } \bar{\rho} = \rho_1, \ldots, \rho_{k+1} \]

of \( k \)-valued SST \[ \Rightarrow \exists \ i \neq j \quad \text{reg} (\rho_i) = \text{reg} (\rho_j) \quad \& \]

\[ \text{maxlag} (\rho_i, \rho_j) \quad \text{small} \]

Some equalities hold just by chance...
Decomposition of 1-register SSTs

Second difficulty: combinatorial property

\( k+1 \) runs \( \bar{\rho} = \rho_1, \ldots, \rho_{k+1} \) of \( k \)-valued SST \( \implies \exists i \neq j \) \( \text{reg}(\rho_i) = \text{reg}(\rho_j) \) \& maxlag(\rho_i, \rho_j) \) small

Some equalities hold just by chance…

\((i,j) \in \text{Equals}(\bar{\rho})\)

\((i,j) \in \text{SmallLags}(\bar{\rho})\)
Decomposition of 1-register SSTs

Second difficulty: combinatorial property

\[ k+1 \text{ runs } \bar{\rho} = \rho_1, \ldots, \rho_{k+1} \] of \( k \)-valued SST

\[ \Rightarrow \exists i \neq j \quad \text{reg} (\rho_i) = \text{reg} (\rho_j) \land \text{maxlag} (\rho_i, \rho_j) \text{ small} \]

Some equalities hold just by chance...

\[ (i,j) \in \text{Equals} (\bar{\rho}) \]

\[ (i,j) \in \text{SmallLags} (\bar{\rho}) \]

\( (i,j) \in \text{Equals} (\bar{\rho}) \) is not robust to pumping
Take \((i,j) \in \text{Equals}(\bar{\rho})\)
Decomposition of 1-register SSTs

Take \((i, j) \in \text{Equals}(\overline{\rho})\)

Let \(\overline{\rho}_n = \text{pump}_n^{L_1,\ldots,L_k}(\overline{\rho})\)

\[\text{maxlag}(\rho_i, \rho_j)\]

\[\text{lag}(\rho_i, \rho_j)\]
Decomposition of 1-register SSTs

Take \((i,j) \in \text{Equals}(\bar{\rho})\)

Let \(\bar{\rho}_n = \text{pump}^n_{L_1,\ldots,L_k}(\bar{\rho})\)

2 cases

\((i,j) \in \text{Equals}(\bar{\rho}_n)\) holds for only finitely many \(n\)'s

\((i,j) \in \text{Equals}(\bar{\rho}_n)\) holds for infinitely many \(n\)'s

\(\text{maxlag}(\rho_i, \rho_j)\)

\(\text{lag}(\rho_i, \rho_j)\)
Decomposition of 1-register SSTs

Take \((i,j) \in \text{Equals}(\bar{\rho})\)

Let \(\bar{\rho}_n = \text{pump}^n_{L_1,\ldots,L_k}(\bar{\rho})\)

2 cases

\((i,j) \in \text{Equals}(\bar{\rho}_n)\) holds for only finitely many \(n\)'s

\(\Rightarrow\) replace \(\bar{\rho}\) by \(\bar{\rho}_n\) for large enough \(n\)

\((i,j) \in \text{Equals}(\bar{\rho}_n)\) holds for infinitely many \(n\)'s
Decomposition of 1-register SSTs

Take \((i,j) \in \text{Equals}(\overline{\rho})\)

Let \(\overline{\rho}_n = \text{pump}^n_{L_1,...,L_k}(\overline{\rho})\)

2 cases

\((i,j) \in \text{Equals}(\overline{\rho}_n)\) holds for only finitely many \(n\)'s

\(\Rightarrow\) replace \(\overline{\rho}\) by \(\overline{\rho}_n\) for large enough \(n\)

\((i,j) \in \text{Equals}(\overline{\rho}_n)\) holds for infinitely many \(n\)'s

\(\Rightarrow\) it holds for all \(n\)'s, including \(n=0\) …
Decomposition of 1-register SSTs

Take \((i, j) \in \text{Equals}(\bar{\rho})\)

Let \(\bar{\rho}_n = \text{pump}^n_{L_1, \ldots, L_k}(\bar{\rho})\)

Word equations of the form

\[
u_0 (v_1)^n u_1 \ldots (v_h)^n u_h = u'_0 (v'_1)^n u'_1 \ldots (v'_h)^n u'_h\]

2 cases

\((i, j) \in \text{Equals}(\bar{\rho})\)

\(\text{maxlag}(\rho_i, \rho_j)\)

\(\text{lag}(\rho_i, \rho_j)\)

\(\bar{\rho}_n\) for large enough \(n\)

[Kortelainen '98, Saarela '15] ⇒ it holds for all \(n\)'s, including \(n=0\)…
Third difficulty: maintain alignments in bounded memory
Decomposition of 1-register SSTs

Third difficulty: maintain alignments in bounded memory

After updating

\[ \text{reg}(\rho) := \text{reg}(\rho) . s \]
\[ \text{reg}(\rho') := a . \text{reg}(\rho') \]
Third difficulty: maintain alignments in bounded memory

Decomposition of 1-register SSTs

After updating

$$reg(\rho) := reg(\rho) \cdot s$$
$$reg(\rho') := a \cdot reg(\rho')$$

+ knowledge on periodicity:

$$reg(\rho) \in \{ stac \}^*$$
$$reg(\rho') \in \{ csta \}^*$$
Decomposition of 1-register SSTs

**Third difficulty:** maintain alignments in bounded memory

After updating

\[
\begin{align*}
\text{reg}(\rho) & := \text{reg}(\rho) \cdot s \\
\text{reg}(\rho') & := a \cdot \text{reg}(\rho')
\end{align*}
\]

+ knowledge on periodicity:

\[
\begin{align*}
\text{reg}(\rho) & \in \{\text{stac}\}^* \\
\text{reg}(\rho') & \in \{\text{csta}\}^*
\end{align*}
\]
Theorem
Every $k$-valued SST with 1 register is a union of $k$ functional SSTs.

Corollary
Equivalence problem for $k$-valued SSTs with 1 register is decidable.

A first steps towards a decomposition theorem for SSTs with many registers…
Beyond the 1-register case

Managed to prove the **combinatorial property** with many registers:

\[ \rho_1, \ldots, \rho_{k+1} \text{ runs of } k\text{-valued SST} \implies \exists i \neq j \quad \text{out}(\rho_i) = \text{out}(\rho_j) \land \maxlag(\rho_i, \rho_j) \text{ small} \]
Managed to prove the **combinatorial property** with many registers:

\[
\rho_1, \ldots, \rho_{k+1} \text{ runs of } k\text{-valued SST} \quad \Rightarrow \quad \exists \ i \neq j \quad \text{out}(\rho_i) = \text{out}(\rho_j) \ \& \ \text{maxlag}(\rho_i, \rho_j) \ \text{small}
\]

**Idea:**

1. not all loops induce repetitions of factors in the registers
2. those that do not induce repetitions can be simulated with less registers
3. word equations + induction on number of registers…
maxlag(\rho_i, \rho_j) \text{ small} \quad \Rightarrow \quad \text{align}(\rho_i, \rho_j) \text{ maintainable in bounded memory}
Beyond the 1-register case

\[ \text{maxlag}(\rho_i, \rho_j) \text{ small } \quad \text{align}(\rho_i, \rho_j) \text{ maintainable in bounded memory} \]
Beyond the 1-register case

\[ \text{maxlag}(\rho_i, \rho_j) \text{ small } \rightarrow \text{align}(\rho_i, \rho_j) \text{ maintainable in bounded memory} \]
Beyond the 1-register case

$maxlag(\rho_i, \rho_j)$ small $\rightarrow$ $align(\rho_i, \rho_j)$ maintainable in bounded memory
Beyond the 1-register case

$\maxlag(\rho_i, \rho_j)$ small $\Rightarrow$ $\align(\rho_i, \rho_j)$ maintainable in bounded memory