Word transducers:
from 2-way to 1-way

Gabriele Puppis
joint works with Felix Baschenis
Olivier Gauwin
Anca Muscholl
Transform objects, here: **words**

**transduction** = mapping (or relation) from words to words

- santiago $\rightarrow$ sntg (erase vowels)
- santiago $\rightarrow$ ogaitnas (reverse)
- santiago $\rightarrow$ santiagosantiago (duplicate)
- santiago $\rightarrow$ antiagos (rotate)
$1\text{DFT} = \text{1-way deterministic finite transducers}$

santiago $\rightarrow$ sntg

erase vowels
Transducers

$1\text{DFT} = 1\text{-way deterministic finite transducers}$

santiago $\rightarrow$ sntg  
erase vowels

$2\text{DFT} = 2\text{-way deterministic finite transducers}$

santiago $\rightarrow$ ogaitnas  
reverse

santiago $\rightarrow$ santiagosantiago  
duplicate
1DFT = 1-way deterministic finite transducers
1NFT = ... non-deterministic ...

santiago  ➔  sntg  erase vowels

2DFT = 2-way deterministic finite transducers
2NFT = ... non-deterministic ...

santiago  ➔  ogaitnas  reverse

santiago  ➔  santiagosantiago  duplicate
SST = streaming string transducers

- deterministic or non-deterministic
- 1-way
- write-only registers to store partial outputs

[Alur, Cerny ’10]
SST = streaming string transducers

- deterministic or non-deterministic
- 1-way
- write-only registers to store partial outputs

Santiago \rightarrow ogaitnas

\texttt{a}_x := a.x

\texttt{out}(x)
Transducers

\[ \text{SST} = \text{streaming string transducers} \]

- deterministic or non-deterministic
- 1-way
- write-only registers to store partial outputs

\[
\begin{align*}
\text{Santiago} & \quad \rightarrow \quad \text{Ogaitnas} \\
\text{Santiago} & \quad \rightarrow \quad \text{SantiagoSantiago} \\
\end{align*}
\]

\[
\begin{align*}
a \mid x := a.x & \quad \rightarrow \quad \text{out}(x) \\
a \mid x := x.a \quad y := y.a & \quad \rightarrow \quad \text{out}(x.y)
\end{align*}
\]
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
MSOT = monadic second-order transductions  [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

φ_<(x,y) = “x, y in the same copy and x < y
or x in the first copy and y in the second copy”
Automata = logic?

\[ w \mapsto ww \]

\[
1\text{DFT} \preceq 2\text{DFT} = \text{DSST} = \text{MSOT}
\]
Automata = logic?

\[
\begin{align*}
1DFT & \ni \ 2DFT & = & DSST & = & MSOT \\
1NFT & \ni \ 2NFT & \ni \ NSST & \ni \ NMSOT
\end{align*}
\]

\[
\begin{align*}
w & \mapsto \, w^2 \\
w & \mapsto \Sigma^{|w|}
\end{align*}
\]
Automata = logic?

\[ \begin{align*}
1\text{DFT} & \preceq 2\text{DFT} & = & \text{DSST} & = & \text{MSOT} \\
1\text{NFT} & \preceq 2\text{NFT} & \neq & \text{NSST} & = & \text{NMSOT} \\
\end{align*} \]
Automata = logic?

$1DFT \subsetneq 2DFT = DSST = MSOT$

$1NFT \subsetneq 2NFT \neq NSST = NMSOT$

$w \mapsto w \cdot w$

$w \mapsto \Sigma^{|w|}$

$w \mapsto w^*$

$uv \mapsto vu$

[= if functional]
First part

2NFT vs 1NFT

- characterisation of 1-way definability
- undecidability in the non-functional case

Second part

Minimising resources

- sweeps of 2NFT vs registers of NSST
- characterisation of k-sweep definability
1-way definability

Problem:

given a 2NFT, is it 1-way definable (equivalent to some 1NFT)?
Problem:

given a 2NFT, is it 1-way definable (equivalent to some 1NFT) ?

The above problem is decidable, with non-elementary complexity.

[Filiot, Gauwin, Reynier, Servais ’13]
Our result:

Given a functional $2\text{NFT}^\star T$,

- we can construct a $1\text{NFT} \ T' \subset T$ \hspace{1cm} (2ExpTime)
- $T$ is 1-way definable \iff $T' = T$
- we can decide the latter \hspace{1cm} (ExpSpace)

$\star$ sweeping for simplicity
Fix a regular language $R$.

$$T(w) = \begin{cases} 
  w.w & \text{if } w \in R \\
  \bot & \text{otherwise}
\end{cases}$$
Fix a regular language $R$.

$$ T(w) = \begin{cases} w.w & \text{if } w \in R \\ \bot & \text{otherwise} \end{cases} $$

- $R = \Sigma^*$  $\longrightarrow$  $T$ is not 1-way definable
Fix a regular language $R$.

$$T(w) = \begin{cases} 
w.w & \text{if } w \in R \\
\bot & \text{otherwise} \end{cases}$$

- $R = \Sigma^*$  $\rightarrow$  $T$ is not 1-way definable
- $R = [0]\Sigma[1]\Sigma\ldots\Sigma[2^n-1]\Sigma$  $\rightarrow$  $T$ has size $n$
  
  equivalent 1-way $T'$ has size $\geq 2^{2^n}$
Fix a regular language $R$. 

$$T(w) = \begin{cases} w.w & \text{if } w \in R \\ \bot & \text{otherwise} \end{cases}$$

- $R = \Sigma^*$ $\longrightarrow$ $T$ is not 1-way definable
- $R = [0]\Sigma[1]\Sigma \ldots \Sigma[2^n-1]\Sigma$ $\longrightarrow$ $T$ has size $n$
  equivalent 1-way $T'$ has size $\geq 2^{2n}$
- $R = \{ \text{abc} \}^*$ $\longrightarrow$ $T$ is 1-way definable
  (output “abc” twice every 3 input letters)
2NFT
Inversion

- two loops
- non-empty outputs produced by the intercepted factors
- output order ≠ input order
Inversion

- two loops
- non-empty outputs produced by the intercepted factors
- output order ≠ input order
Inversion

- two loops
- non-empty outputs produced by the intercepted factors
- output order ≠ input order
Pumping inversions

2NFT

equivalent 1NFT
Pumping inversions

2NFT  equivalent 1NFT

output produced between the inversion is periodic
T is 1-way definable

every inversion produces an output of bounded period
T.F.A.E.:

T is 1-way definable

every inversion produces an output of bounded period

every run admits a stair-like decomposition
T.F.A.E.:

- T is 1-way definable
- every inversion produces an output of bounded period
- every run admits a stair-like decomposition
T.F.A.E.:

- T is 1-way definable
- Every inversion produces an output of bounded period
- Every run admits a stair-like decomposition
The characterisation
The characterisation
The characterisation
Outputs entirely covered by inversions are periodic...
T.F.A.E.:

- T is 1-way definable
- Every inversion produces an output of bounded period
- Every run admits a stair-like decomposition
T.F.A.E.:

- T is 1-way definable
- every inversion produces an output of bounded period
- every run admits a stair-like decomposition

The characterisation can be guessed in ExpSpace
The non-functional case

Whether a non-functional 2NFT is 1-way definable is undecidable.

Reduction from PCP — given morphisms $f, g : \Sigma^* \rightarrow \Delta^*$

does $\exists w \in \Sigma^+ f(w) = g(w)$ ?
The non-functional case

Whether a non-functional 2NFT is 1-way definable is undecidable.

Reduction from PCP — given morphisms $f, g : \Sigma^* \rightarrow \Delta^*$

does $\exists w \in \Sigma^+ \ f(w) = g(w)$ ?

❖ Encodings: $w.u \mapsto w.\$^m$
Whether a non-functional 2NFT is 1-way definable is undecidable.

Reduction from PCP — given morphisms \( f, g : \Sigma^* \to \Delta^* \)
\[
\text{does } \exists w \in \Sigma^+ \quad f(w) = g(w) \quad ?
\]

- Encodings:
  \[ w.u \mapsto w.u^m \]
  good if \( m = |u| \)
  and \( u = f(w) \)
  and \( u = g(w) \)
Whether a non-functional 2NFT is 1-way definable is undecidable.

Reduction from PCP — given morphisms \( f, g : \Sigma^* \rightarrow \Delta^* \)

\[ \exists w \in \Sigma^+ \quad f(w) = g(w) \]

❖ Encodings: \( w.u \mapsto w.m \)

bad if \( m \neq |u| \)

or \( u \neq f(w) \)

or \( u \neq g(w) \)
The non-functional case

Whether a non-functional 2NFT is 1-way definable is undecidable.

Reduction from PCP — given morphisms $f, g : \Sigma^* \rightarrow \Delta^*$ does $\exists w \in \Sigma^+ \; f(w) = g(w)$?

- Encodings: $w.u \mapsto w.$^m$ bad if $m \neq |u|$ or $u \neq f(w)$ or $u \neq g(w)$

All encodings are bad iff $\; T$ is 1-way definable
The non-functional case

Whether a non-functional 2NFT is 1-way definable is undecidable.

Reduction from PCP — given morphisms \( f, g : \Sigma^* \to \Delta^* \)
does \( \exists w \in \Sigma^+ \quad f(w) = g(w) \)?

- **Encodings**: \( w.u \mapsto w.m \) bad if \( m \neq |u| \)
or \( u \neq f(w) \)
or \( u \neq g(w) \)

All encodings are bad iff
\[ T \text{ is 1-way definable} \]
The non-functional case

Whether a non-functional 2NFT is 1-way definable is undecidable.

Reduction from PCP — given morphisms \( f, g : \Sigma^* \rightarrow \Delta^* \)

does \( \exists w \in \Sigma^+ \ f(w) = g(w) \) ?

- Encodings: \( w.u \mapsto w.m \)
  - bad if \( m \neq |u| \)
  - or \( u \neq f(w) \)
  - or \( u \neq g(w) \)

guess \( w = w_1.a.w_2, u = u_1.u_2 \)
check \( f(a) \) not a prefix of \( u_2 \)
output \( |f(w_1)| $|u_2| \)

read \( w.u \) output \( w \)

All encodings are bad iff

T is 1-way definable
Part 2: minimising resources

What do we mean by resource?

- number of control states
- amount of non-determinism
- number of sweeps
- number of registers
- ...

Part 2: minimising resources

What do we mean by resource?

- number of control states
- amount of non-determinism
- number of sweeps
- number of registers
- ...

... interesting… but poorly understood
What do we mean by resource?

- number of control states
- amount of non-determinism
- number of sweeps
- number of registers

...
Given a deterministic SST over a unary output alphabet, one can compute the minimum number of registers in EXP\textsc{TIME}.

[Alur, Raghothaman ’13]
A previous result

Given a deterministic SST over a unary output alphabet, one can compute the minimum number of registers in \( \text{EXPTIME} \).

[Alur, Raghothaman ’13]

Our setting:

- arbitrary alphabet
- weak restriction on updates…
- non-deterministic (but still functional) SST
Recall $2\text{NFT} \approx \text{SST}$ in the functional case
Recall $2\text{NFT} \approx \text{SST}$ in the functional case

$$w_1 \# w_2 \# \ldots \# w_n \mapsto \text{rev}(w_1) \# \text{rev}(w_2) \# \ldots \# \text{rev}(w_n)$$
Recall \( 2 \text{NFT} \approx \text{SST} \) in the functional case

\[ w_1 \# w_2 \# \ldots \# w_n \mapsto \text{rev}(w_1) \# \text{rev}(w_2) \# \ldots \# \text{rev}(w_n) \]
The following are also equally expressive:

- concatenation-free SST
  \[ x := a.y.b \quad \checkmark \]
  \[ x := y.z \quad \times \]
- sweeping 2NFT
- bounded reversal 2NFT
The following are also equally expressive:

- concatenation-free SST
- sweeping 2NFT
- bounded reversal 2NFT

\[ u \# v \mapsto v \# u \]
| 2\(k\)-sweep 2NFT | can be transformed into | \(k\)-register SST |
| 2\(k\)-sweep 2NFT | can be transformed into | 2\(k\)-sweep 2NFT |
\[ \begin{align*}
2k\text{-sweep} \ 2\text{NFT} & \quad \text{can be transformed into} \quad k\text{-register} \ \text{SST} \\
k\text{-register} \ \text{SST} & \quad \text{can be transformed into} \quad 2k\text{-sweep} \ 2\text{NFT}
\end{align*} \]
Sweeps vs registers

$2k$-sweep 2NFT can be transformed\* into $k$-register SST

$k$-register SST can be transformed\* into $2k$-sweep 2NFT

\* in $2\text{ExpTime}$

\* in $\text{ExpTime}$
A characterization similar to 1-way definability:

Given a functional sweeping 2NFT $T$ and a number $k$

- we can construct a $k$-sweep NFT $T' \subsetneq T$ \hfill (2ExpTime)
- $T$ is $k$-sweep definable \iff $T' = T$
- we can decide the latter \hfill (ExpSpace)
Given a sweeping 2NFT, we can compute:

- the minimum # of sweeps \((\text{ExpSpace})\)
- a sweeping 2NFT with the min. # of sweeps \((2\text{ExpTime})\)
Minimisation results

Given a sweeping 2NFT, we can compute:

- the minimum # of sweeps \((\text{ExpSpace})\)
- a sweeping 2NFT with the min. # of sweeps \((2\text{ExpTime})\)

Given a concatenation-free SST, we can compute:

- the minimum # of registers \((2\text{ExpSpace})\)
- a concatenation-free SST with the min. # of registers \((3\text{ExpTime})\)
Conclusions… what next?

- Formalise the results for 2NFT (non-sweeping)
- Characterise sweepingness with unknown # of passes
- Minimise # of registers of SST (non concatenation-free)
- Find decidable non-functional cases ($k$-valuedness ?)
Conclusions… what next?

- Formalise the results for 2NFT (non-sweeping)
- Characterise *sweepingness* with unknown # of passes
- Minimise # of registers of SST (non concatenation-free)
- Find decidable non-functional cases (*k*-valuedness ?)