Towards Register Minimisation of Streaming String Transducers

Pierre-Alain Reynier

LIS, Aix-Marseille Université & CNRS

Transducers

Automata accept objects / Transducers transform objects

A transduction is a function (or even a relation) from words to words
➜ In this talk, we focus on functions

Examples:

- ➜ ERASE: "Oxford" $\mapsto$ "xfrd"
- ➜ LAST: "Oxford" $\mapsto$ "dddddd"
- ➜ REVERSE: "Oxford" $\mapsto$ "drofxO"
- ➜ COPY: "Oxford" $\mapsto$ "OxfordOxford"
- ➜ REPLACE: "Oxford#I love $1" $\mapsto$ "I love Oxford"
- ➜ SORT: "Oxford" $\mapsto$ "dfoOrx"

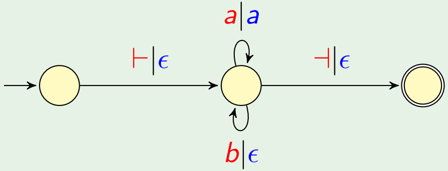
Transducers

Some applications:

- language and speech processing
- model-checking infinite state-space systems
- verification of web sanitizers
- string pattern matching
- XML transformations (nested word)
- model for recursive programs (nested word)

(One/Two-way) finite state transducers

Example (A transducer T)

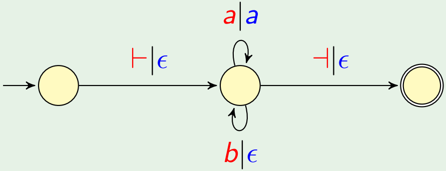


Semantics $[\![T]\!]$: $\textsc{Erase}: \vdash w \dashv \ \mapsto a^{\#_a(w)}$, with $w \in \{a, b\}^*$

Non-determinism: semantics is a relation

(One/Two-way) finite state transducers

Example (A transducer T)



Semantics $[\![T]\!]$: $\textsc{Erase}$: $\vdash w \dashv \;\mapsto a^{\#_a(w)}$, with $w \in \{a, b\}^*$

Non-determinism: semantics is a relation

A transducer is:

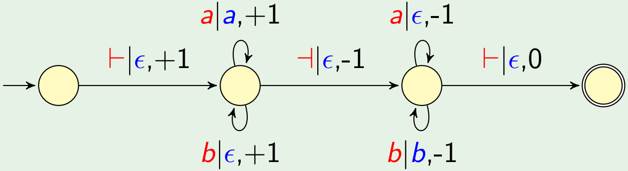
- functional if it realizes a function
- deterministic if the underlying automaton is deterministic

Classes: det1W, fun1W, 1W

➜ Too low expressive power ($\textsc{Reverse}$, $\textsc{Copy}$, $\textsc{Replace}$, $\textsc{Sort}$)

(One/Two-way) finite state transducers

Example (A transducer T)



Semantics $[\![T]\!]$: $\textsc{Sort}$: $\vdash w \dashv \ \mapsto a^{\#_a(w)} b^{\#_b(w)}$, 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: det1W, fun1W, 1W, det2W, fun2W, 2W

Regular Word Functions

[EH01] $\begin{array}{r} \text{fun2W} \\ = \text{det2W} \end{array}$

Regular Word Functions

MSO-definable Transducers (à la Courcelle)

[EH01]

[EH01] fun2W =det2W

Regular Word Functions

MSO-definable Transducers

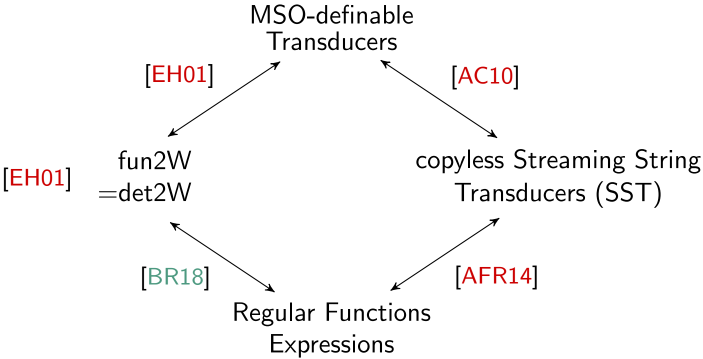
[EH01]

[AC10]

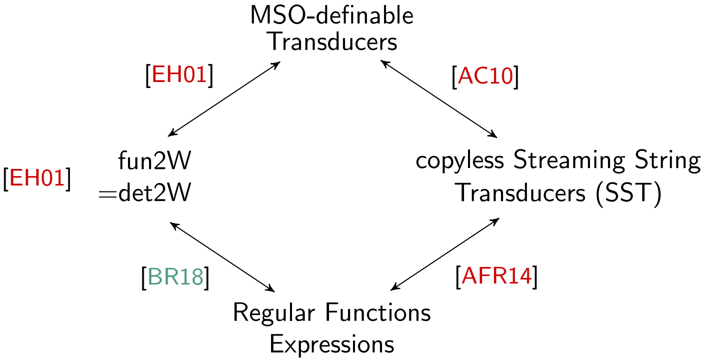
[EH01] fun2W
=det2W

copyless Streaming String Transducers (SST)

Regular Word Functions



Regular Word Functions



- closed under composition
- regular languages are preserved by inverse image
- functionality and equivalence are decidable

Streaming String Transducers [AC10]

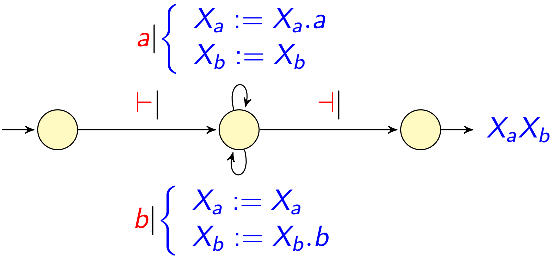
1W deterministic autom.
+ registers

Register updates:

- X:=u.Y.v
- X:=Y.Z

X,Y,Z: registers
u,v: words in Σ^*

$\vdash w \dashv \;\mapsto\; a^{\#_a(w)} b^{\#_b(w)}$



Streaming String Transducers [AC10]

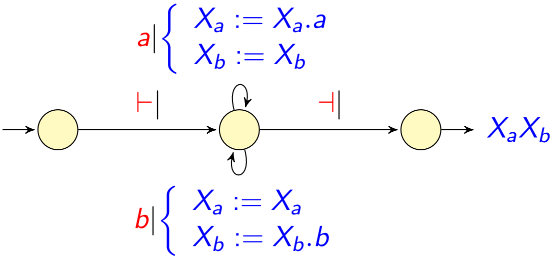
1W deterministic autom.
+ registers

Register updates:

- X:=u.Y.v
- X:=Y.Z

X,Y,Z: registers
u,v: words in Σ^*

$\vdash w \dashv \;\mapsto a^{\#_a(w)} b^{\#_b(w)}$



Expressiveness results :

- det1W $\equiv$ 1-register appending SST X:=X.a

Streaming String Transducers [AC10]

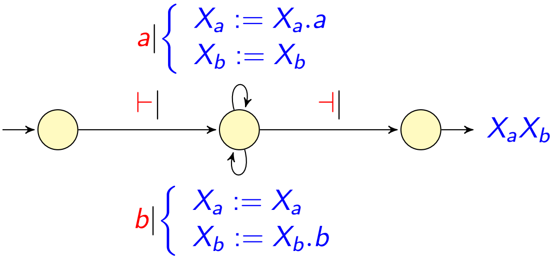
1W deterministic autom.
+ registers

Register updates:

- X:=u.Y.v
- X:=Y.Z

X,Y,Z: registers
u,v: words in Σ^*

$\vdash w \dashv \;\mapsto\; a^{\#_a(w)} b^{\#_b(w)}$



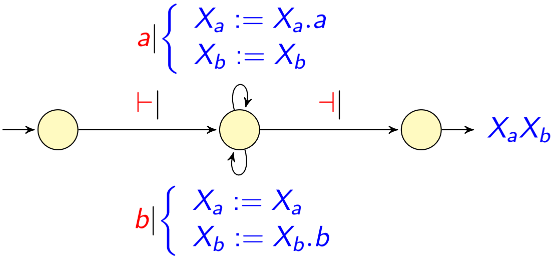
Expressiveness results :

- det1W $\equiv$ 1-register appending SST — X:=X.a
- fun1W $\equiv$ appending SST — X:=Y.a

Streaming String Transducers [AC10]

$\vdash w \dashv \;\mapsto a^{\#_a(w)} b^{\#_b(w)}$

1W deterministic autom.
+ registers



Register updates:

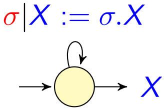
- X:=u.Y.v
- X:=Y.Z

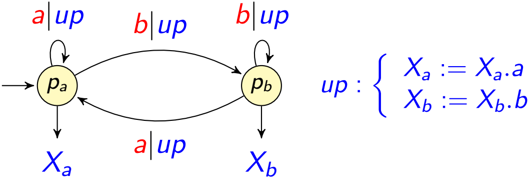
X,Y,Z: registers
u,v: words in Σ^*

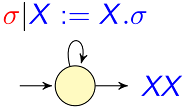
Expressiveness results :

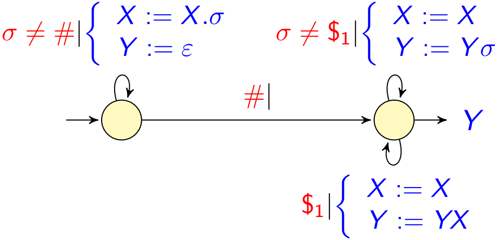
- det1W $\equiv$ 1-register appending SST X:=X.a
- fun1W $\equiv$ appending SST X:=Y.a
- fun2W $\equiv$ copyless SST (X,Y):=(X,X) is forbidden

Examples of SST









Register Minimisation Problem for SST

Motivations: Streaming and simplification of models

- minimisation/determinisation of automata
- normal form $\rightsquigarrow$ learning
- 2way: reduce number of passes

Register Minimisation Problem for class $\mathcal{S}$ of SST

Input: $T \in \mathcal{S}$ and $k \in \mathbb{N}$
Question: Does there exist $T' \in \mathcal{S}$ with k registers s.t. $T \equiv T'$?

Related works

- [AR13] Additive Cost Register Automata — X:=Y+c, c$\in \mathbb{Z}$
- [BGMP16] concatenation-free funNSST — X:=uYv

Classes of Functions

Regular functions det2W=copyless SST=MSOT

REVERSE COPY

Classes of Functions

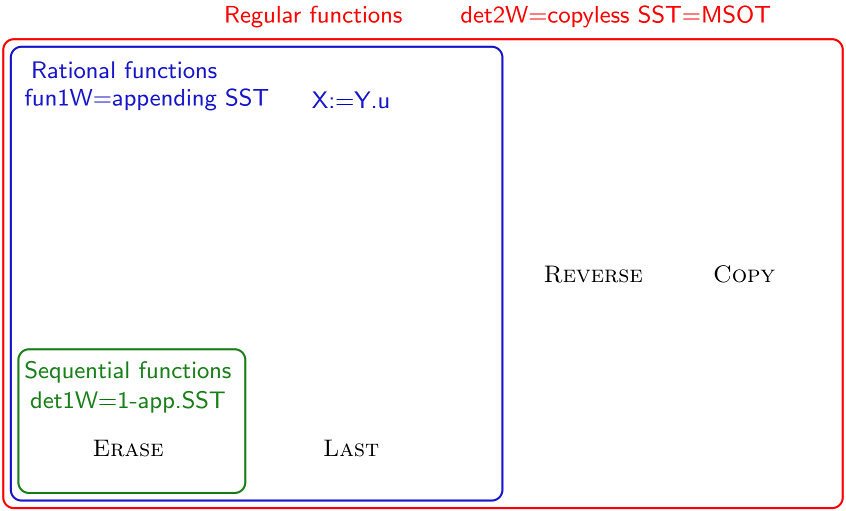
Regular functions det2W=copyless SST=MSOT

Rational functions
fun1W=appending SST X:=Y.u

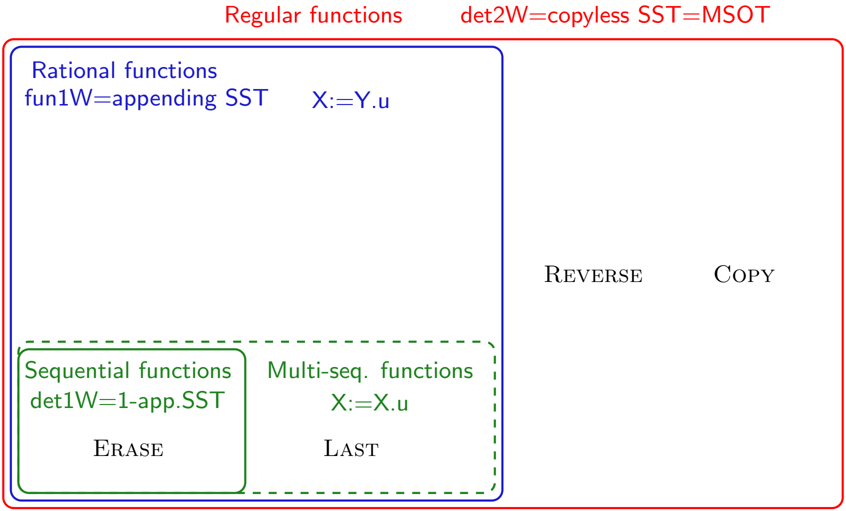
Reverse Copy

Last

Classes of Functions



Classes of Functions



In this talk

- Rational functions (X:=Y.u)
  ➜ [LICS16] with L. Daviaud and J.M. Talbot

- Multi-sequential functions (X:=X.u)
  ➜ [FoSSaCS17] with L. Daviaud, I. Jecker and D. Villevalois

Overview

Overview

Rational functions and appending SST

Appending SST: only updates X:=Y.u

Facts:

- appending SST = fun1W
- appending SST $\rightsquigarrow$ fun1W is polynomial (guess the register)
- appending SST with 1 register = det1W

Register minimisation for appending SST

Input: an appending SST T and $k \in \mathbb{N}$
Question: does there exist an app. SST T' with k registers s.t. $T \equiv T'$?

➜ for $k = 1$, our problem is the det1W-definability of fun1W

From rational functions to sequential ones

Sequentiality Problem [Choffrut77]

Input: a fun1W T
Question: does there exist an equivalent det1W?

Standard technique:

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

Configurations of the form $\{(p, a), (q, \varepsilon), (s, bb)\}$

From rational functions to sequential ones

Sequentiality Problem [Choffrut77]

Input: a fun1WT

Question: does there exist an equivalent det1W?

Standard technique:

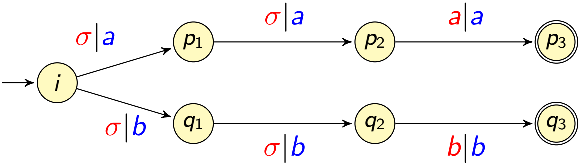
- subset construction starting from the set of initial states.
- output longest common prefix
- store the unproduced outputs in the configuration

Configurations of the form $\{(p, a), (q, \varepsilon), (s, bb)\}$

Issue: termination (bound the size of unproduced outputs)

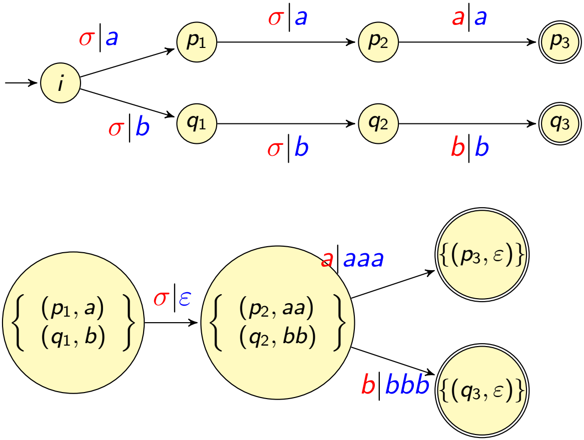
An example

$\textsc{Last}$ on Σ^3



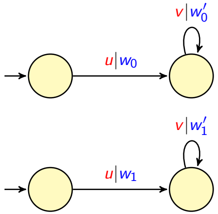
An example

Last on Σ^3



Twinning Property [Choffrut77]

For all situations like:



we have $\mathrm{delay}(w_0, w_1) = \mathrm{delay}(w_0 w_0', w_1 w_1')$

We define:

$$\mathrm{delay}(u, v) = \mathrm{lcp}(u, v)^{-1}.(u, v)$$

Example:

$\mathrm{lcp}(aaa, aab) = aa$

$\mathrm{delay}(aaa, aab) = (a, b)$

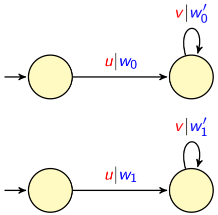
Twinning Property [Choffrut77]

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:



we have $\text{delay}(w_0, w_1) = \text{delay}(w_0 w'_0, w_1 w'_1)$

$T \models$ Twinning Property $\implies \forall (p, x) \in$ subset constr., $|x| \leq n^2 M$

Theorem ([Choffrut77])

$T \models$ Twinning Property $\iff$ There exists an equivalent det1W

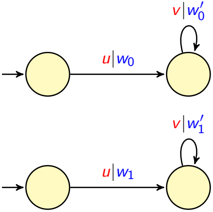
Twinning Property [Choffrut77]

We define:

$$\mathrm{delay}(u, v) = \mathrm{lcp}(u, v)^{-1}.(u, v)$$

Example:
$\mathrm{lcp}(aaa, aab) = aa$
$\mathrm{delay}(aaa, aab) = (a, b)$

For all situations like:



we have $\mathrm{delay}(w_0, w_1) = \mathrm{delay}(w_0 w_0', w_1 w_1')$

$T \models$ Twinning Property $\implies \forall (p, x) \in$ subset constr., $|x| \leq n^2 M$

Theorem ([Choffrut77])

$T \models$ Twinning Property $\iff$ There exists an equivalent det1W

Theorem ([WK95])

Twinning Property can be decided in PTime.

Register minimisation using Twinning Property

Our objective: Characterize when a fun1W can be expressed by an appending SST with k registers.

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

Intuition:

2 reg. needed if there are 2 runs with arbitrarily large delays

$k+1$ reg. needed if there are $k+1$ runs with pairwise arb. large delays

k registers are sufficient if for every $k+1$ runs, 2 of them remain close

Register minimisation using Twinning Property

Our objective: Characterize when a fun1W can be expressed by an appending SST with k registers.

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

Intuition:

2 reg. needed if there are 2 runs with arbitrarily large delays

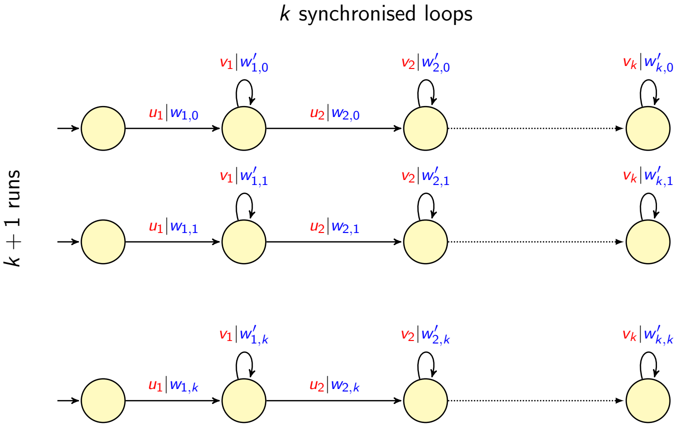
$k+1$ reg. needed if there are $k+1$ runs with pairwise arb. large delays

k registers are sufficient if for every $k+1$ runs, 2 of them remain close

For every $k+1$ runs, 2 of them remain close

Twinning Property of order k

For all situations like:



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

we have $\mathrm{delay}(w_{1,i} \dots w_{\ell,i}, w_{1,j} \dots w_{\ell,j}) = \mathrm{delay}(w_{1,i} \dots w_{\ell,i} w'_{\ell,i}, w_{1,j} \dots w_{\ell,j} w'_{\ell,j})$

Register minimisation using Twinning Property

Lemma

If a fun1W 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" to one of these k runs.

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

Register minimisation using Twinning Property

Lemma

If a fun1W 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" to one of these k runs.

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

Theorem

- *A fun1W is definable by a k-app. SST iff it satisfies the TP of order k*
- *TP of order k can be decided in PSpace (k given in unary)*

Register minimisation using Twinning Property

Lemma

If a fun1W 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" to one of these k runs.

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

Theorem

- *A fun1W is definable by a k-app. SST iff it satisfies the TP of order k*
- *TP of order k can be decided in PSpace (k given in unary)*

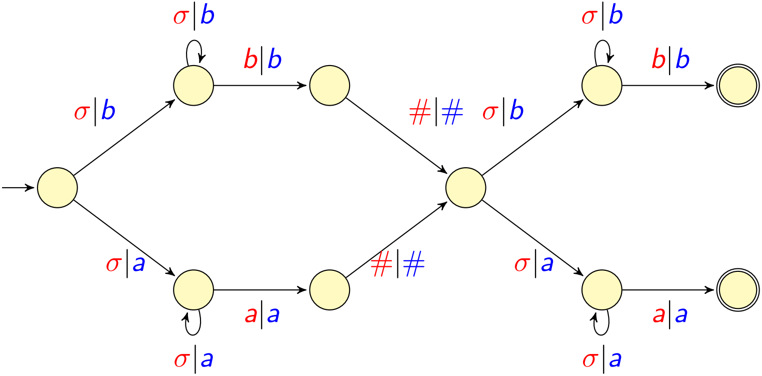
Corollary

The register minimisation problem for appending SST is PSpace-complete.

Example

How many registers for the following function?

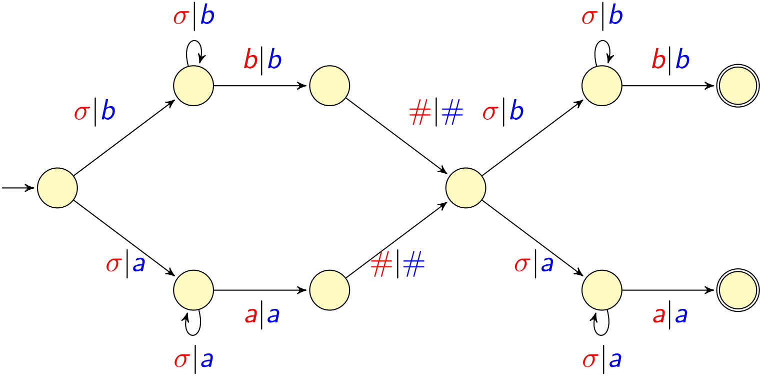
$$\textsc{Last}^2 : u_1\#u_2 \mapsto \textsc{Last}(u_1)\#\textsc{Last}(u_2)$$



Example

How many registers for the following function?

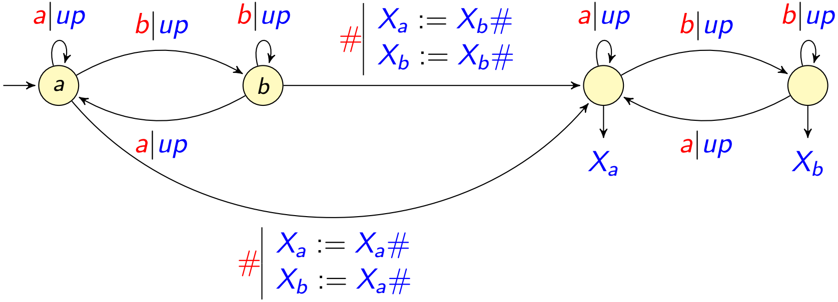
$$\textsc{Last}^2 : u_1\#u_2 \mapsto \textsc{Last}(u_1)\#\textsc{Last}(u_2)$$



Only 2 registers!

Example

$$\textsc{Last}^2 : u_1\#u_2 \mapsto \textsc{Last}(u_1)\#\textsc{Last}(u_2)$$



Overview

Multi-sequential functions

Definition ([CS86])

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

➜ allows a parallel evaluation in a streaming scenario

Examples:

- $\textsc{Last}$ on $\Sigma = \{a, b\}$ is multi-sequential: split Σ^+ as $\Sigma^* a \uplus \Sigma^* b$

Multi-sequential functions

Definition ([CS86])

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

➜ allows a parallel evaluation in a streaming scenario

Examples:

- $\textsc{Last}$ on $\Sigma = \{a, b\}$ is multi-sequential: split Σ^+ as $\Sigma^* a \uplus \Sigma^* b$
- $\textsc{Last}^2 : u_1 \# u_2 \mapsto \textsc{Last}(u_1)\#\textsc{Last}(u_2)$ is multi-sequential: split the domain according to $last(u_1), last(u_2) \in \{a, b\}$

Multi-sequential functions

Definition ([CS86])

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

➜ allows a parallel evaluation in a streaming scenario

Examples:

- $\textsc{Last}$ on $\Sigma = \{a, b\}$ is multi-sequential: split Σ^+ as $\Sigma^* a \uplus \Sigma^* b$
- $\textsc{Last}^2 : u_1 \# u_2 \mapsto \textsc{Last}(u_1)\#\textsc{Last}(u_2)$ is multi-sequential: split the domain according to $last(u_1), last(u_2) \in \{a, b\}$
- $\textsc{Last}^* : u_1 \# \ldots \# u_n \mapsto \textsc{Last}(u_1)\# \ldots \#\textsc{Last}(u_n)$ is not multi-seq.

Multi-sequential functions

Definition ([CS86])

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

Definition (Appending SST with independent registers)

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

Multi-sequential functions

Definition ([CS86])

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

Definition (Appending SST with independent registers)

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

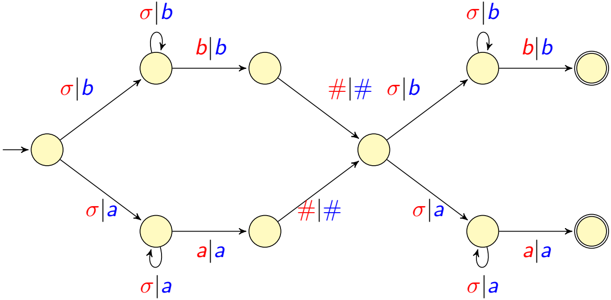
Observations:

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

➜ Register minimisation in this class $\equiv$ Minimisation of size of the union

Example

$$\mathrm{L{\scriptstyle AST}}^2 : u_1\#u_2 \mapsto \mathrm{L{\scriptstyle AST}}(u_1)\#\mathrm{L{\scriptstyle AST}}(u_2)$$

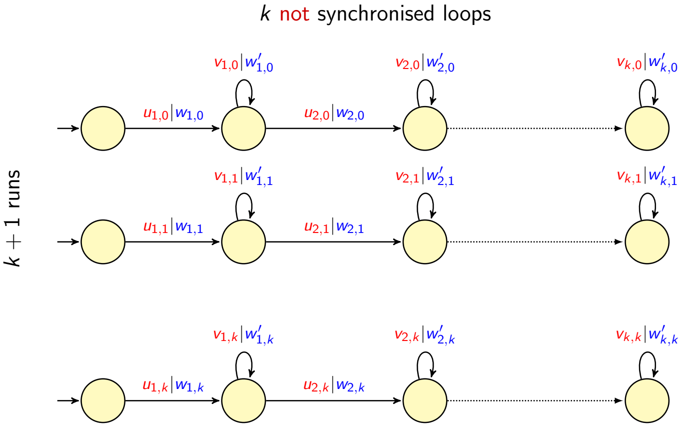


➜ Requires 4 independent registers

Registers cannot be reset!

Branching twinning property of order k

For all situations like:

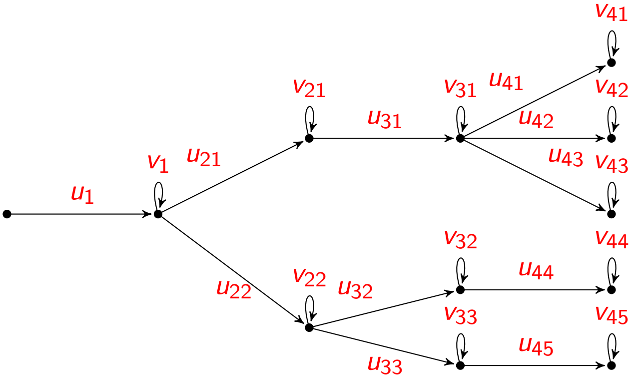


there are two runs $0 \leq i < j \leq k$ s.t. for every loop ℓ with same input words,

we have $\text{delay}(w_{1,i} \dots w_{\ell,i}, w_{1,j} \dots w_{\ell,j}) = \text{delay}(w_{1,i} \dots w_{\ell,i} w'_{\ell,i}, w_{1,j} \dots w_{\ell,j} w'_{\ell,j})$

Branching twinning property of order k

Tree representation of input words:



Branching twinning property of order k

Theorem

- *A fun1W is definable by a k-app. SST with independent registers iff it satisfies the BTP of order k.*
- *The BTP of order k is decidable in PSpace (k in unary).*

Branching twinning property of order k

Theorem

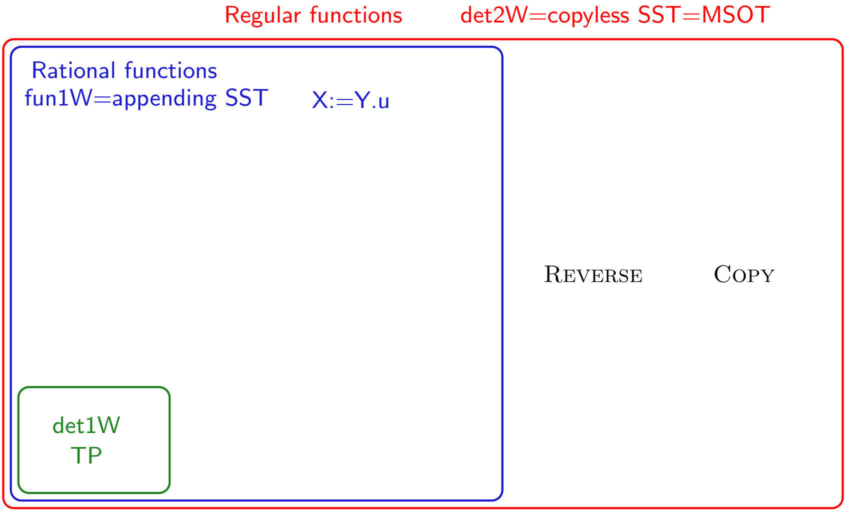
- *A fun1W is definable by a k-app. SST with independent registers iff it satisfies the BTP of order k.*
- *The BTP of order k is decidable in PSpace (k in unary).*

Theorem

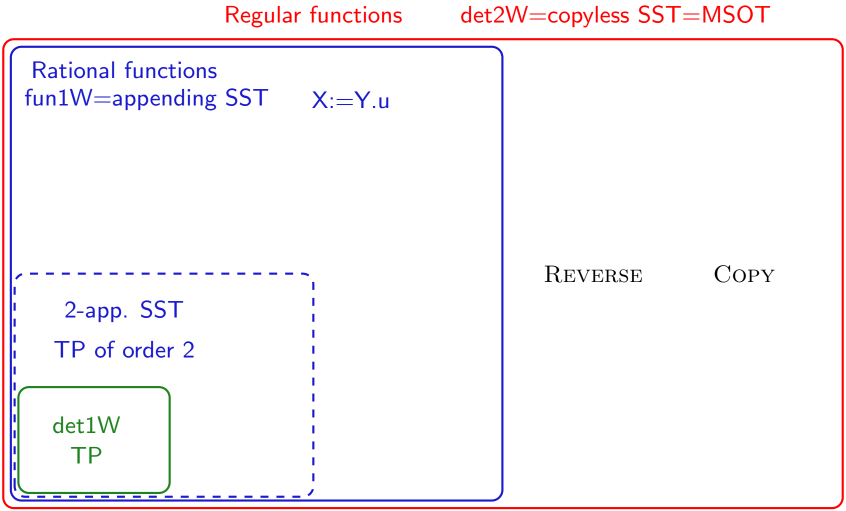
The register minimisation problem for appending SST with independent registers is PSpace-complete.

Overview

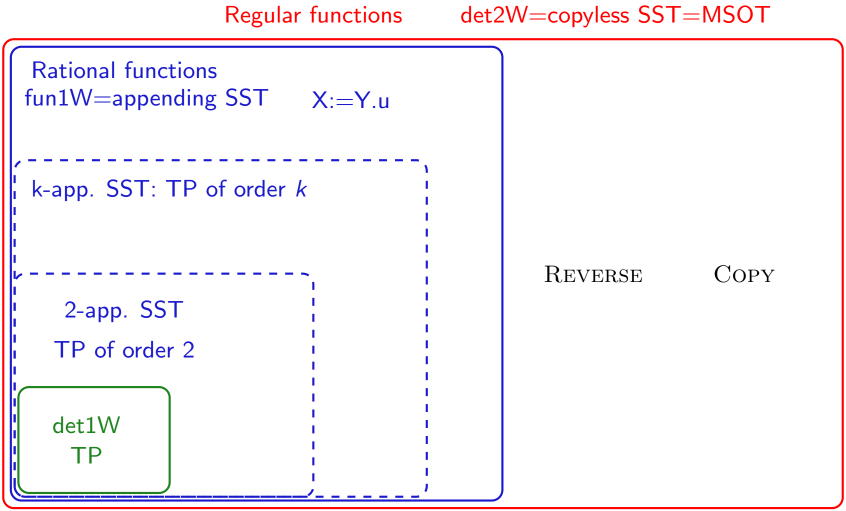
Summary


Regular functions
det2W=copyless SST=MSOT
Rational functions
fun1W=appending SST
X:=Y.u
Reverse
Copy
det1W
TP

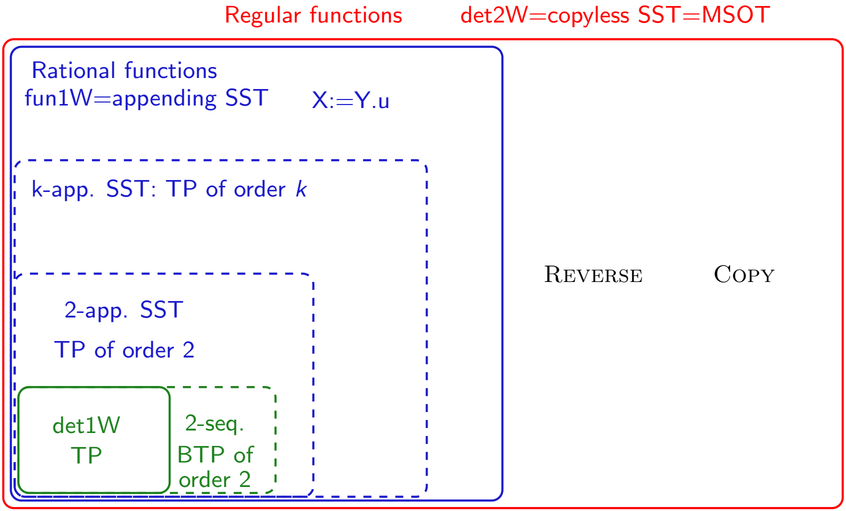
Summary


Regular functions
det2W=copyless SST=MSOT
Rational functions
fun1W=appending SST
X:=Y.u
2-app. SST
TP of order 2
det1W
TP
Reverse
Copy

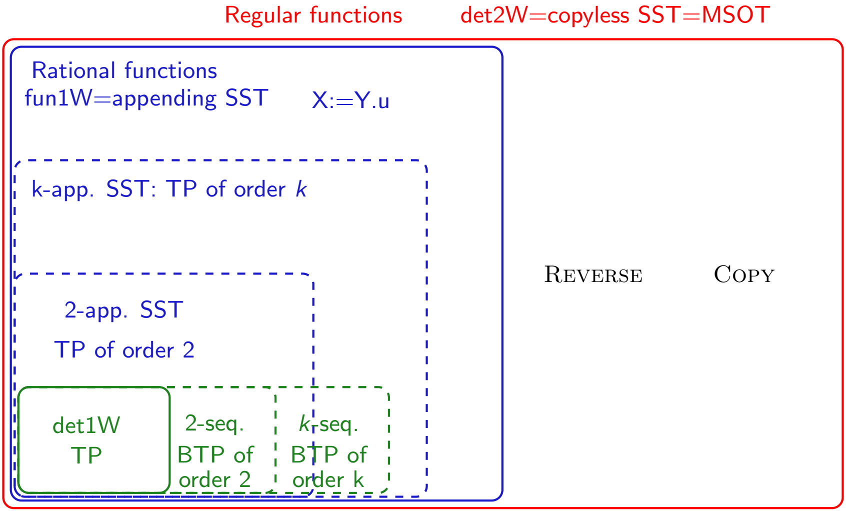
Summary


Regular functions
det2W=copyless SST=MSOT
Rational functions
fun1W=appending SST
X:=Y.u
k-app. SST: TP of order k
2-app. SST
TP of order 2
det1W
TP
REVERSE
COPY

Summary

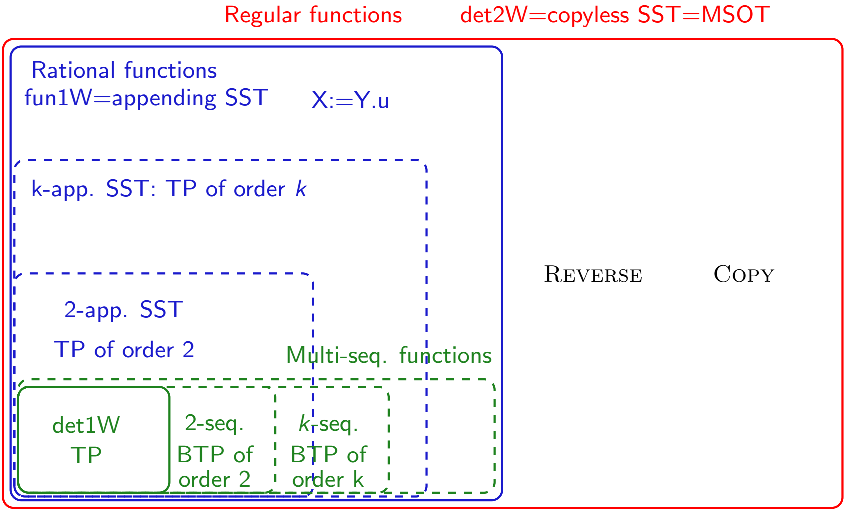

Regular functions
det2W=copyless SST=MSOT
Rational functions
fun1W=appending SST
X:=Y.u
k-app. SST: TP of order k
2-app. SST
TP of order 2
det1W
TP
2-seq.
BTP of
order 2
Reverse
Copy

Summary



Regular functions
det2W=copyless SST=MSOT
Rational functions
fun1W=appending SST
X:=Y.u
k-app. SST: TP of order k
2-app. SST
TP of order 2
det1W
TP
2-seq.
BTP of
order 2
k-seq.
BTP of
order k
Reverse
Copy

Summary


Regular functions
det2W=copyless SST=MSOT
Rational functions
fun1W=appending SST
X:=Y.u
k-app. SST: TP of order k
2-app. SST
TP of order 2
Multi-seq. functions
det1W
TP
2-seq.
BTP of
order 2
k-seq.
BTP of
order k
Reverse
Copy

I did not present...

Alternative characterizations:

- bounded variation property
- Lipschitz property

I did not present...

Alternative characterizations:

- bounded variation property
- Lipschitz property

Functional $\rightsquigarrow$ finite-valued

I did not present...

Alternative characterizations:

- bounded variation property
- Lipschitz property

Functional $\rightsquigarrow$ finite-valued

Extension to "weak" weighted automata on semigroups:

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

Perspectives

Shift from rational to regular functions
➜ deal with both prepending and appending: X:=u.Y.v (on-going)
➜ deal with concatenation of registers

Weighted automata: replace set semantics with other aggregations

Extensions to infinite words, nested words

Perspectives

Shift from rational to regular functions

➜ deal with both prepending and appending: X:=u.Y.v (on-going)

➜ deal with concatenation of registers

Weighted automata: replace set semantics with other aggregations

Extensions to infinite words, nested words

Thanks!

Classes of Transductions

Regular functions
det2W=copyless SST
=MSOT

Copy

Reverse

Classes of Transductions

Rational functions
fun1W=appending SST
(X:=Y.u)

LAST

Regular functions
det2W=copyless SST
=MSOT

COPY

REVERSE

Classes of Transductions

Rational relations
1W=appending NSST

SUBWORD $u \mapsto \{u' | u' \preceq u\}$

Rational functions
fun1W=appending SST
(X:=Y.u)

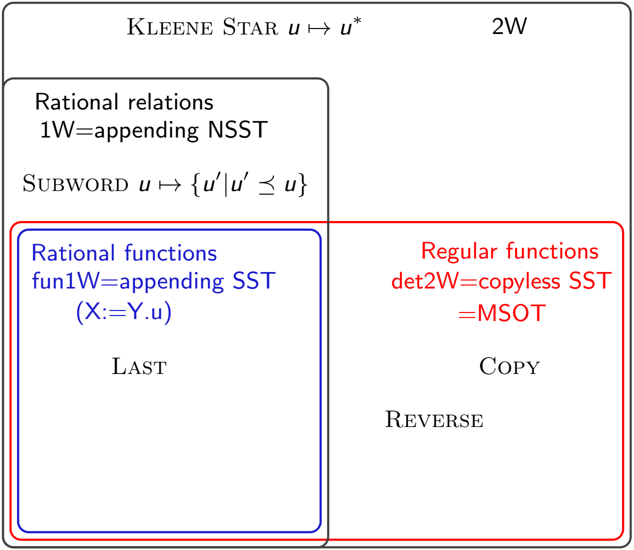
LAST

Regular functions
det2W=copyless SST
=MSOT

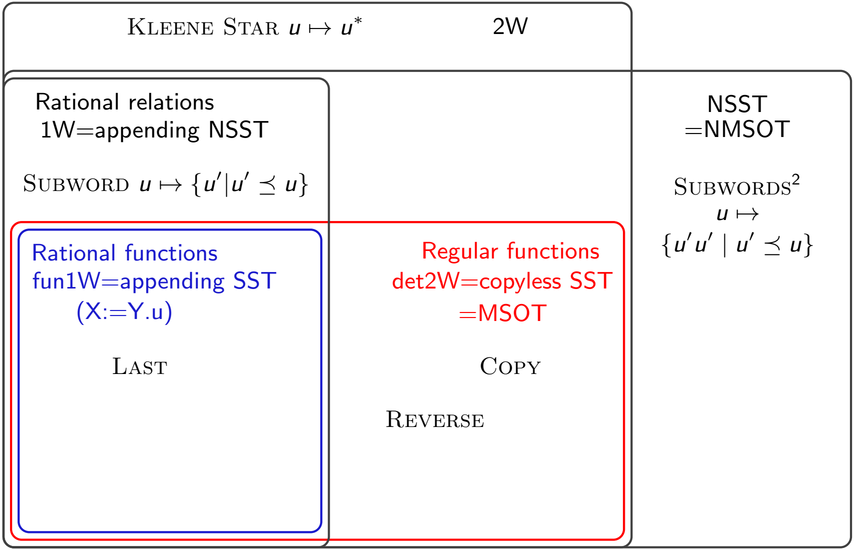
COPY

REVERSE

Classes of Transductions



Classes of Transductions



Alternative characterizations

$f : \Sigma^* \mapsto \Gamma^*$

	bounded variation	Lipschitz property
det1W	$\forall n\ \exists N\ \forall u, v \in dom(f)$, $d(u,v) \leq n \Rightarrow d(f(u), f(v)) \leq N$	$\exists L\ \forall u, v \in dom(f)$, $d(f(u), f(v)) \leq L.(d(u,v) + 1)$
k registers		
k independent registers		

Alternative characterizations

$f : \Sigma^* \mapsto \Gamma^*$

	bounded variation	Lipschitz property
det1W	$\forall n \; \exists N \; \forall u, v \in dom(f)$, $d(u, v) \leq n \Rightarrow d(f(u), f(v)) \leq N$	$\exists L \; \forall u, v \in dom(f)$, $d(f(u), f(v)) \leq L.(d(u, v) + 1)$
k registers	$\forall n \; \exists N \; \forall u_0 \dots u_k \in dom(f)$, $(\forall i \neq j, d(u_i, u_j) \leq n)$ $\Rightarrow \exists i \neq j. d(f(u_i), f(u_j)) \leq N$	?
k independent registers		

Alternative characterizations

$f : \Sigma^* \mapsto \Gamma^*$

	bounded variation	Lipschitz property
det1W	$\forall n\ \exists N\ \forall u, v \in dom(f)$, $d(u,v) \leq n \Rightarrow d(f(u), f(v)) \leq N$	$\exists L\ \forall u, v \in dom(f)$, $d(f(u), f(v)) \leq L.(d(u,v)+1)$
k registers	$\forall n\ \exists N\ \forall u_0 \dots u_k \in dom(f)$, $(\forall i \neq j, d(u_i, u_j) \leq n)$ $\Rightarrow \exists i \neq j . d(f(u_i), f(u_j)) \leq N$	?
k independent registers	?	$\exists L\ \forall u_0 \dots u_k \in dom(f)$, $\exists i \neq j$ s.t. $d(f(u_i), f(u_j)) \leq L.(d(u_i, u_j)+1)$