Simplifying transducers

removing two-wayness and non-determinism

Pierre-Alain Reynier

LIF, Aix-Marseille Université & CNRS

Based on joint works with L Daviaud, E Filiot, O Gauwin, I Jecker, B Monmege, F Servais, JM Talbot and D Villevalois

Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie

Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
- PostDoc at ULB on controller synthesis with Jean-François Raskin

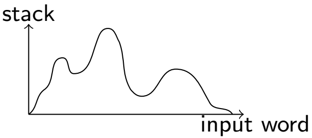
Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
- PostDoc at ULB on controller synthesis with Jean-François Raskin
- Ass Prof at Marseille
  ➜ Mid 2009: New collaboration between ULB and UMarseille: with JF Raskin, E Filliot, F Servais and JM Talbot
  Topic: streamability of XML transformations

Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
- PostDoc at ULB on controller synthesis with Jean-François Raskin
- Ass Prof at Marseille
  ➜ Mid 2009: New collaboration between ULB and UMarseille:
  with JF Raskin, E Filliot, F Servais and JM Talbot
  Topic: streamability of XML transformations

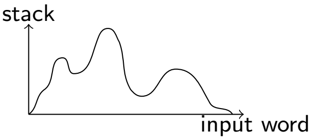
Determinization of visibly pushdown transducers



Genesis of this work

- PhD at LSV on timed systems with Patricia Bouyer and François Laroussinie
- PostDoc at ULB on controller synthesis with Jean-François Raskin
- Ass Prof at Marseille
  ➜ Mid 2009: New collaboration between ULB and UMarseille: with JF Raskin, E Filliot, F Servais and JM Talbot
  Topic: streamability of XML transformations

Determinization of visibly pushdown transducers



We did not succeed to solve this problem...
...but we found interesting open problems on word transducers

From Languages to Transductions

Languages	Transductions
Input $\rightarrow \{0,1\}$	*Input* $\rightarrow$ *Outputs*
automata	transducers
accept inputs	transform inputs

From Languages to Transductions

Languages	Transductions
Input $\to \{0,1\}$	*Input* $\to$ *Outputs*
automata	transducers
accept inputs	transform inputs

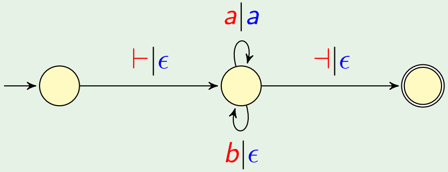
Applications:

- Word-to-word transducers:
  - language and speech processing
  - model-checking infinite state-space systems
  - verification of web sanitizers
  - string pattern matching
- Nested-word-to-word transducers:
  - XML transformations
  - model for recursive programs

(Two-way) finite state transducers

$=$ associate output words with transitions of a finite state automaton

Example (A transducer T)



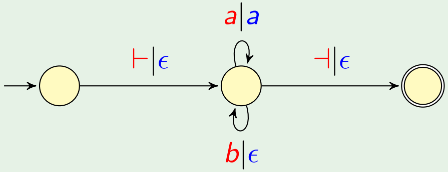
Semantics $[\![T]\!]$: $f : \vdash w \dashv \; \mapsto a^{\#_a(w)}$, with $w \in \{a, b\}^*$

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

Non-determinism: semantics is a relation

A transducer is:

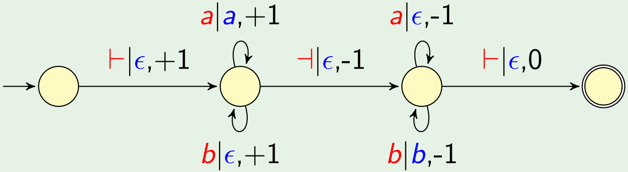
- functional if it realizes a function
- deterministic if the underlying automaton is deterministic

Classes: DFT, fNFT, NFT

(Two-way) finite state transducers

= associate output words with transitions of a finite state automaton

Example (A transducer T)



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

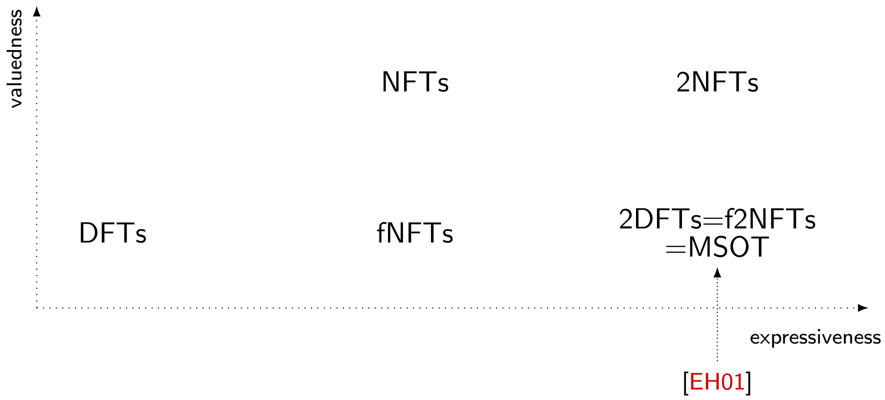
Non-determinism: semantics is a relation

A transducer is:

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

Classes: DFT, fNFT, NFT, 2DFT, f2NFT, 2NFT

Classes of Transductions



Classes of Transductions

$u \mapsto \mathit{mirror}(u)$

valuedness

NFTs $\subset$ 2NFTs

DFTs fNFTs $\subset$ 2DFTs=f2NFTs =MSOT

expressiveness

Classes of Transductions

valuedness

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

NFTs $\subset$ 2NFTs

$\cup$ $\cup$

DFTs fNFTs $\subset$ 2DFTs=f2NFTs =MSOT

expressiveness

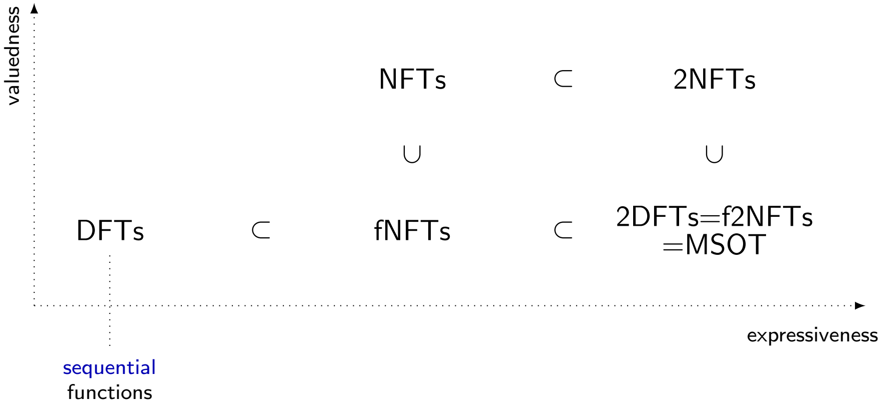
Classes of Transductions

valuedness

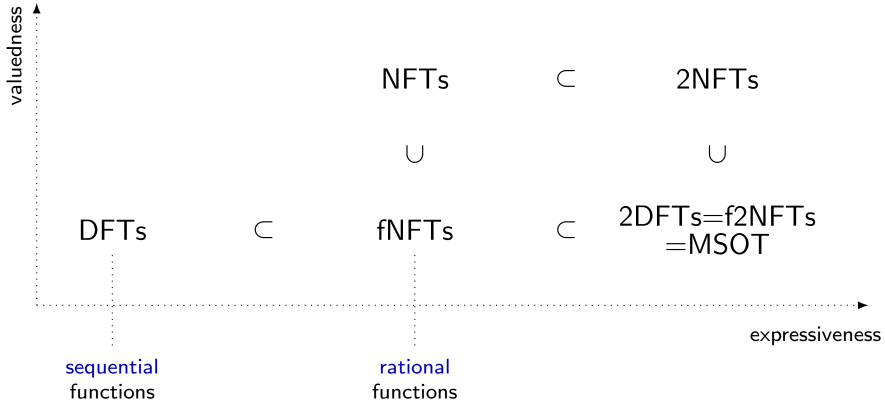
$$
\begin{array}{ccccccc}
 & & u \mapsto \mathit{last}(u)^{|u|} & & \text{NFTs} & \subset & \text{2NFTs} \\
 & & & & \cup & & \cup \\
\text{DFTs} & & \subset & & \text{fNFTs} & \subset & \begin{array}{c}\text{2DFTs}=\text{f2NFTs}\\ =\text{MSOT}\end{array}
\end{array}
$$

expressiveness

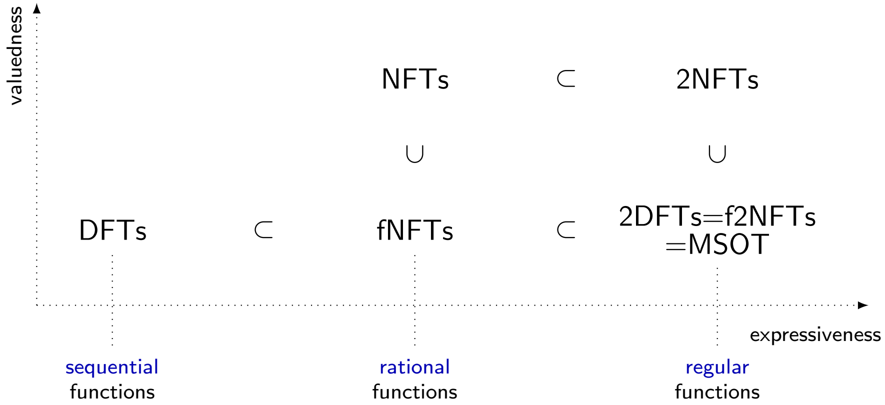
Classes of Transductions



Classes of Transductions

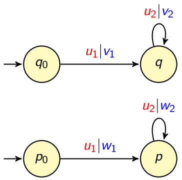


Classes of Transductions



Classes of Transductions

Twinning Property



$\mathsf{delay}(v_1, w_1) = \mathsf{delay}(v_1 v_2, w_1 w_2)$

valuedness

NFTs $\subset$ 2NFTs

$\cup$ $\cup$

DFTs $\subset$ fNFTs $\subset$ 2DFTs=f2NFTs =MSOT

PTIME

[Choffrut77,WK95,BCPS03]

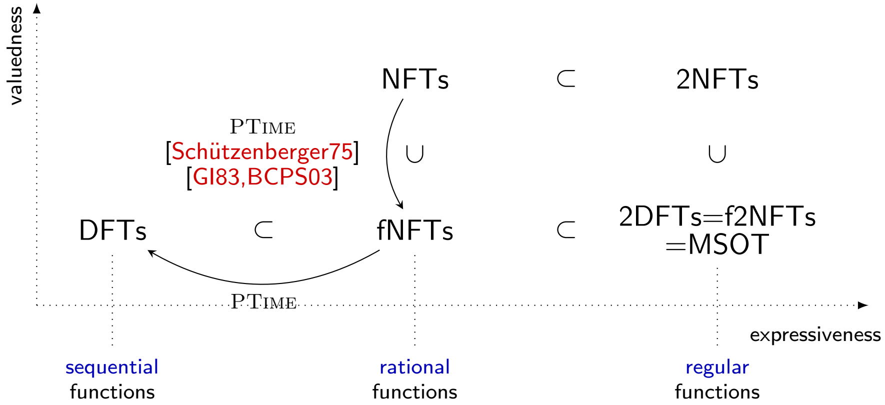
expressiveness

sequential functions

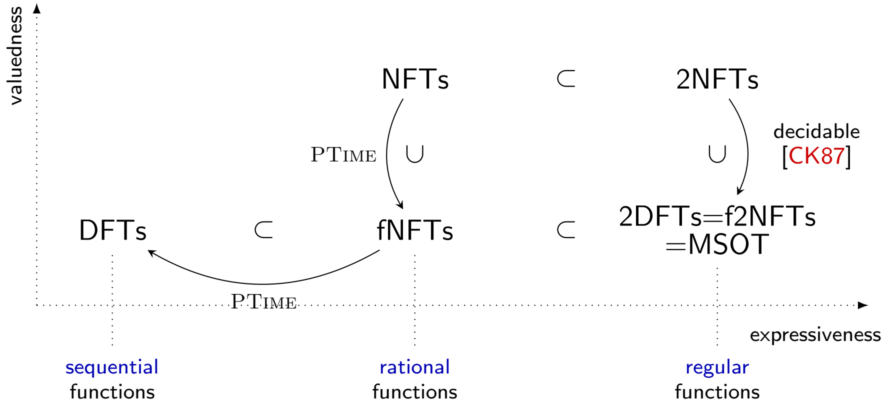
rational functions

regular functions

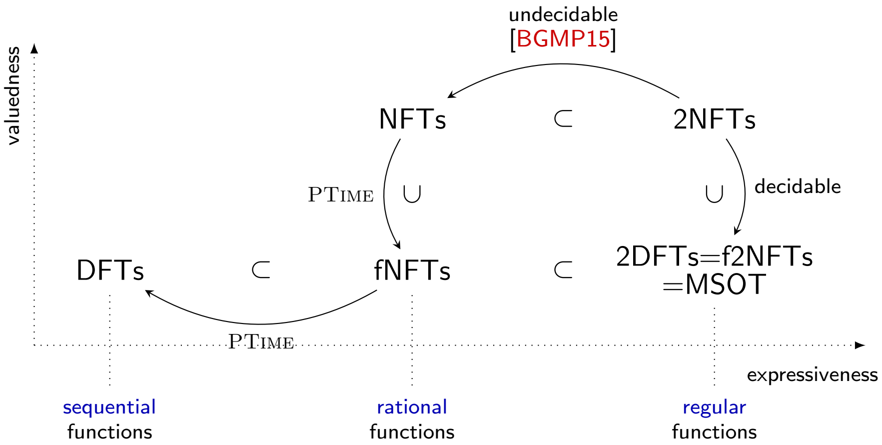
Classes of Transductions



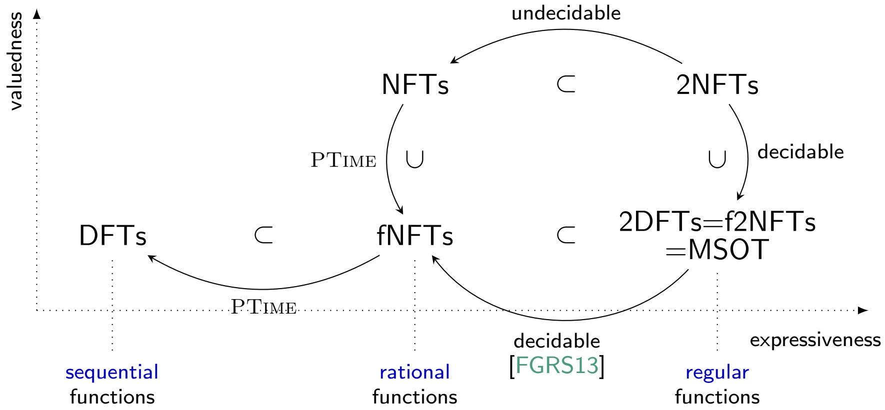
Classes of Transductions



Classes of Transductions



Classes of Transductions



Simplification of models

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

➜ Natural question:

- minimization of automata

Simplification of models

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

➜ Natural question:

- minimization of automata
- determinization

Simplification of models

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

➜ Natural question:

- minimization of automata
- determinization
- 2way: minimize number of passes

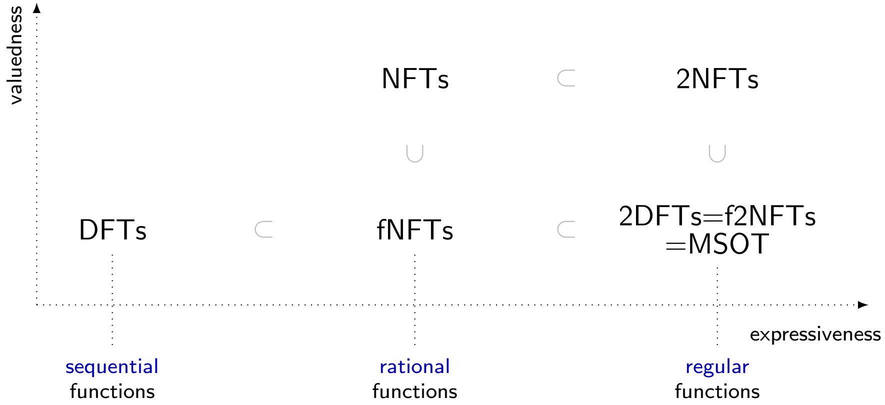
Simplification of models

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

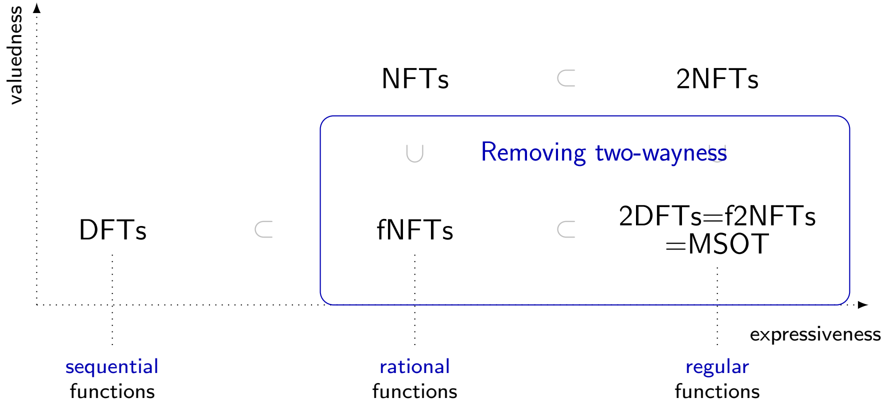
➜ Natural question:

- minimization of automata
- determinization
- 2way: minimize number of passes
- automata with registers: minimize number of registers
- ...

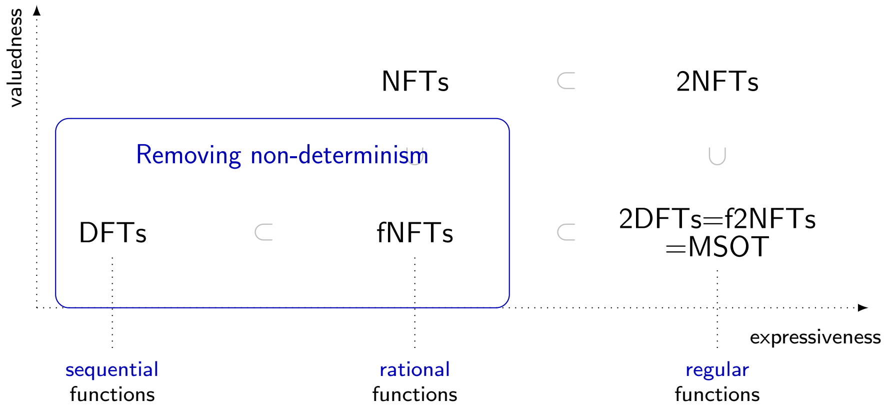
This talk


valuedness
NFTs ⊂ 2NFTs
∪ ∪
DFTs ⊂ fNFTs ⊂ 2DFTs=f2NFTs =MSOT
expressiveness
sequential functions
rational functions
regular functions

This talk


valuedness
NFTs
⊂
2NFTs
Removing two-wayness
∪
∪
DFTs
⊂
fNFTs
⊂
2DFTs=f2NFTs
=MSOT
expressiveness
sequential
functions
rational
functions
regular
functions

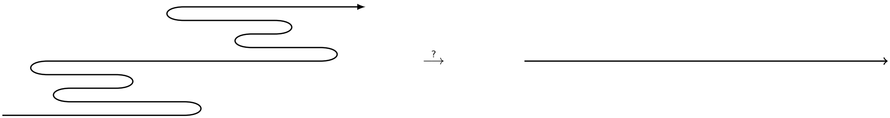
This talk


valuedness
NFTs
⊂
2NFTs
Removing non-determinism
∪
∪
DFTs
⊂
fNFTs
⊂
2DFTs=f2NFTs
=MSOT
expressiveness
sequential
functions
rational
functions
regular
functions

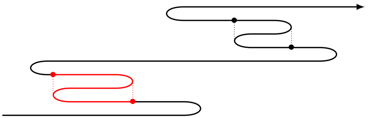
Overview

Overview

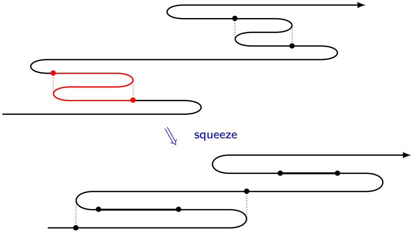
Rabin and Scott for transducers



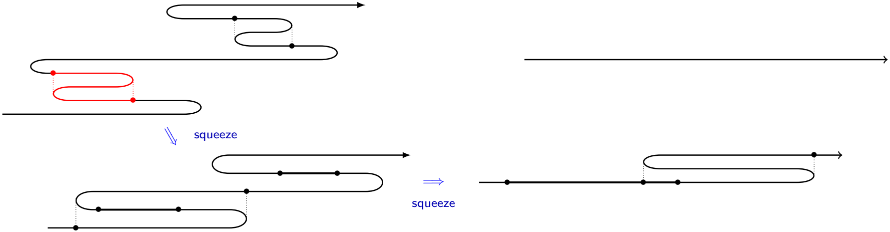
Rabin and Scott for transducers



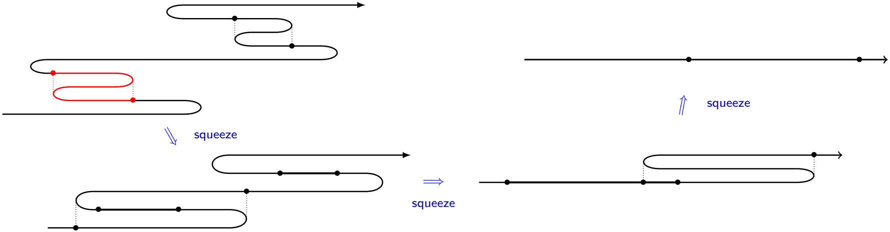
Rabin and Scott for transducers



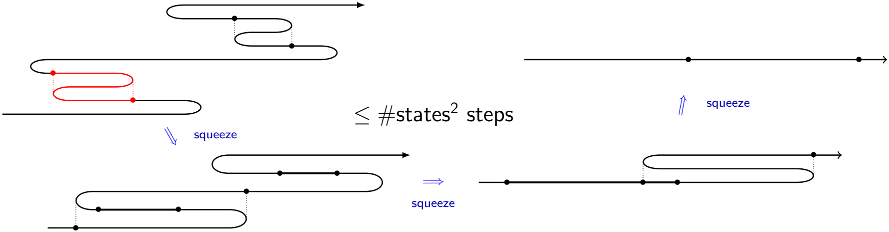
Rabin and Scott for transducers



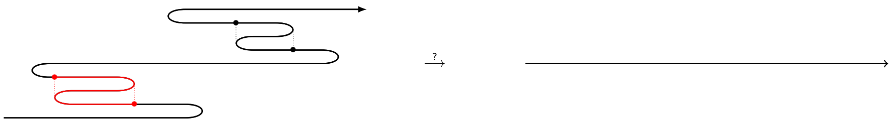
Rabin and Scott for transducers



Rabin and Scott for transducers



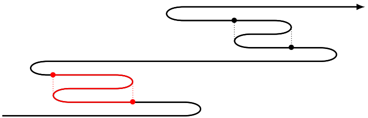
Rabin and Scott for transducers



Lemma

If T is one-way definable then every zigzag transducer of T is also one-way definable.

Rabin and Scott for transducers



$\xrightarrow{?}$

Lemma

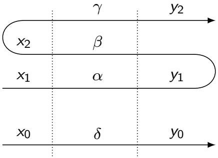
If T is one-way definable then every zigzag transducer of T is also one-way definable.

Decision Procedure:

1. repeat N^2 times:
   - are all zigzag transducers of T NFT-definable?
     - yes: $T \leftarrow \mathit{squeeze}(T)$
     - no: STOP: the initial 2NFTwas not NFT-definable!
2. remove backward transitions: you get an equivalent NFT

Rabin and Scott for transducers (2)

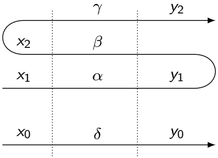
Consider a zigzag transducer Z



$$x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0$$

Rabin and Scott for transducers (2)

Consider a zigzag transducer Z

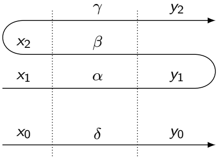


$$x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0$$

$\Longrightarrow$ α,β,γ,δ have conjugate primitive roots of length polynomial in $|Z|$

Rabin and Scott for transducers (2)

Consider a zigzag transducer Z



$$x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0$$

$\Longrightarrow$ $\alpha,\beta,\gamma,\delta$ have conjugate primitive roots of length polynomial in $|Z|$

Theorem

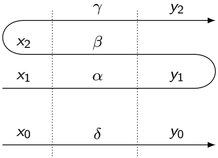
Given a zigzag transducer Z, we build a fNFT Z' s.t.:

1. $|Z'| = exp(O(|Z|))$
2. $R(Z') \subseteq R(Z)$
3. *Z is NFT-definable $\iff$ $dom(Z) \subseteq dom(Z')$*

3. is decidable.

Rabin and Scott for transducers (2)

Consider a zigzag transducer Z



$$x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0$$

$\implies$ α,β,γ,δ have conjugate primitive roots of length polynomial in $|Z|$

Theorem

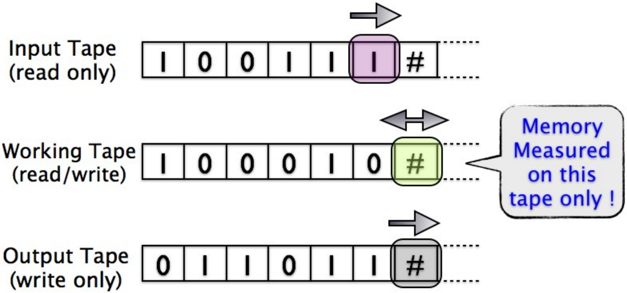
Given a zigzag transducer Z, we build a fNFT Z' s.t.:

1. $|Z'| = exp(O(|Z|))$
2. $R(Z') \subseteq R(Z)$
3. *Z is NFT-definable* $\iff$ $dom(Z) \subseteq dom(Z')$

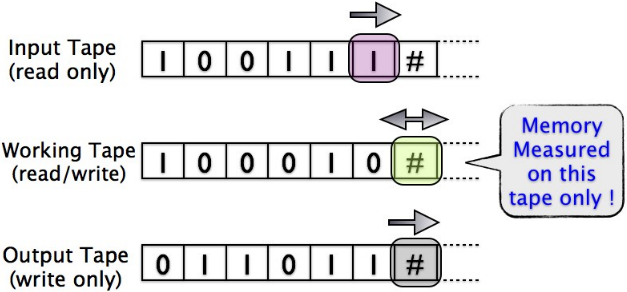
3 is decidable.

Yields a non-elementary decision procedure

Streaming model: Deterministic Turing Transducer



Streaming model: Deterministic Turing Transducer



Bounded Memory Problem

Input: a transformation T

Output: can T be realized with bounded memory?

$$\exists B \in \mathbb{N} \cdot \forall u \in dom(T)$$
$$T(u) \text{ can be computed with } B\text{-bounded memory?}$$

Streamability of word-to-word transformations

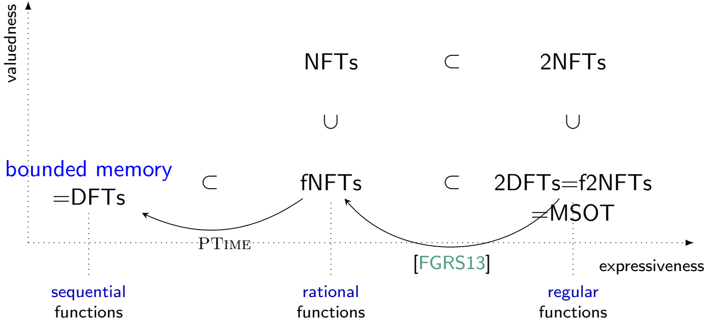
Observation:

$f : \Sigma^* \to \Sigma^*$ is bounded memory iff it is DFT-definable.

Streamability of word-to-word transformations

Observation:

$f : \Sigma^* \to \Sigma^*$ is bounded memory iff it is DFT-definable.



Corollary

Bounded memory is decidable for regular word functions.

Recent works on removing two-wayness

Two new techniques to obtain elementary complexity:

- [BGMP17] (Bordeaux)
  - careful analysis of loops
  - algebraic techniques (idempotent element, Simon)
  - decision in 2ExpSpace
  - also yields decidability results related to sweeping transducers

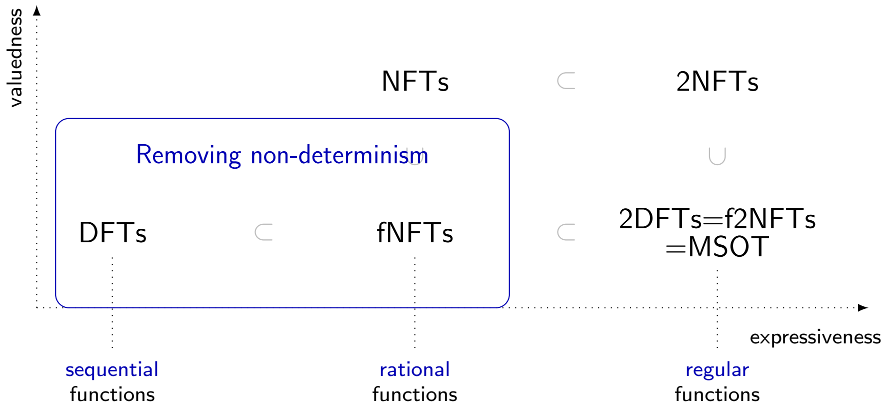
Recent works on removing two-wayness

Two new techniques to obtain elementary complexity:

- [BGMP17] (Bordeaux)
  - careful analysis of loops
  - algebraic techniques (idempotent element, Simon)
  - decision in 2ExpSpace
  - also yields decidability results related to sweeping transducers
- [MRT17] (Marseille, unpublished)
  - normalization procedure
  - more amenable to implementation
  - promising technique to address determinization of visibly pushdown transducers

Overview

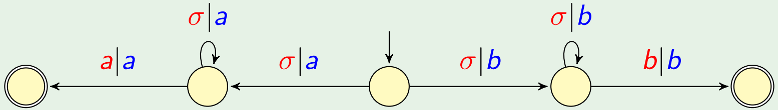
Removing non-determinism



Multi-sequential functions

Fact: Not every one-way transducer can be determinized

Example

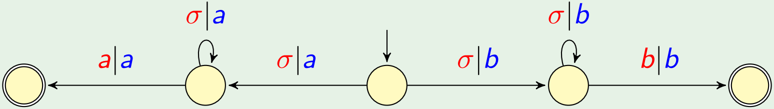


Semantics $[\![T]\!] = \textsc{Last}: \; w\sigma \mapsto \sigma^{|w|+1}$, with $\sigma \in \{a, b\}, w \in \{a, b\}^+$

Multi-sequential functions

Fact: Not every one-way transducer can be determinized

Example



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

But $\textsc{Last}$ can be written as the "sum" of two sequential functions

Multi-sequential functions

Fact: Not every one-way transducer can be determinized

Definition ([CS86])

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

Motivation: parallel evaluation of the function

Examples: ($\textsc{Last}$: $w\sigma \mapsto \sigma^{|w|+1}$, with $\sigma \in \{a, b\}, w \in \{a, b\}^+$)

- $\textsc{Last}$ is multi-sequential: split the domain as $\Sigma^* a \uplus \Sigma^* b$

Multi-sequential functions

Fact: Not every one-way transducer can be determinized

Definition ([CS86])

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

Motivation: parallel evaluation of the function

Examples: ($\textsc{Last}: \; w\sigma \; \mapsto \sigma^{|w|+1}$, with $\sigma \in \{a, b\}, w \in \{a, b\}^{+}$)

- $\textsc{Last}$ is multi-sequential: split the domain 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

Fact: Not every one-way transducer can be determinized

Definition ([CS86])

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

Motivation: parallel evaluation of the function

Examples: ($\textsc{Last}: \; w\sigma \; \mapsto \sigma^{|w|+1}$, with $\sigma \in \{a, b\}, w \in \{a, b\}^+$)

- $\textsc{Last}$ is multi-sequential: split the domain 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

Fact: Not every one-way transducer can be determinized

Definition ([CS86])

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

Motivation: parallel evaluation of the function

Theorem ([CS86,FJ15])

Multi-sequentiality can be decided in Ptime.

This work: minimization of the size of the union.

From rational to sequential functions [Choffrut77]

Sequentiality Problem

Input: a fNFT

Question: does there exist an equivalent DFT?

Theorem ([Choffrut77])

Let T be a fNFT. T.f.a.e:

- *there exists a DFT T' s.t. $T \equiv T'$*
- *T satisfies the twinning property*
- *$[\![T]\!]$ satisfies the Lipschitz property*
  $\exists L \in \mathbb{N} \mid \forall u, v \in dom(f), d(f(u), f(v)) \leq L.d(u, v)$

From rational to sequential functions [Choffrut77]

Sequentiality Problem

Input: a fNFT

Question: does there exist an equivalent DFT?

Theorem ([Choffrut77])

Let T be a fNFT. T.f.a.e:

- *there exists a DFT T' s.t. $T \equiv T'$*
- *T satisfies the twinning property*
- *$[\![T]\!]$ satisfies the Lipschitz property*
  $\exists L \in \mathbb{N} \mid \forall u, v \in dom(f), d(f(u), f(v)) \leq L.d(u, v)$

Twinning Property can be decided in PTime ([WK95])

Corollary

The Sequentiality Problem is decidable in PTime.

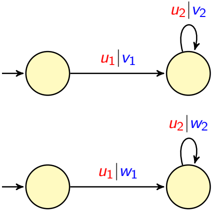
Twinning Property [Choffrut77]

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 $\mathrm{delay}(v_1, w_1) = \mathrm{delay}(v_1 v_2, w_1 w_2)$

Twinning Property [Choffrut77]

Define:

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

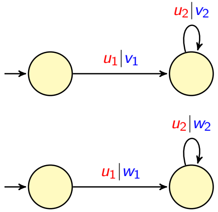
Example:
$\text{lcp}(aaa, aab) = aa$
$\text{delay}(aaa, aab) = (a, b)$

Determinization procedure:

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

States are sets $\{(p, a), (q, \varepsilon), (s, bb)\}$

For all situations like:



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

Twinning Property [Choffrut77]

Define:

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

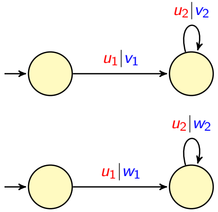
Example:
$\mathrm{lcp}(aaa, aab) = aa$
$\mathrm{delay}(aaa, aab) = (a, b)$

Determinization procedure:

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

States are sets $\{(p, a), (q, \varepsilon), (s, bb)\}$

For all situations like:



we have $\mathrm{delay}(v_1, w_1) = \mathrm{delay}(v_1 v_2, w_1 w_2)$

Lemma

$T \models$ Twinning Property
$\Longrightarrow$ *bounded delays*
$\Longrightarrow$ *termination of subset constr.*

Characterization of k-sequential functions

k-sequentiality Problem

Input: a fNFT T and $k \in \mathbb{N}$

Question: does there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$?

Characterization of k-sequential functions

k-sequentiality Problem

Input: a fNFT T and $k \in \mathbb{N}$
Question: does there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$?

Theorem

Let T be a fNFT and $k \in \mathbb{N}$. T.f.a.e:

- *there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$*
- *T satisfies the branching twinning property of order k*
- *$[\![T]\!]$ satisfies the k-Lipschitz property*

Characterization of k-sequential functions

k-sequentiality Problem

Input: a fNFT T and $k \in \mathbb{N}$
Question: does there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$?

Theorem

Let T be a fNFT and $k \in \mathbb{N}$. T.f.a.e:

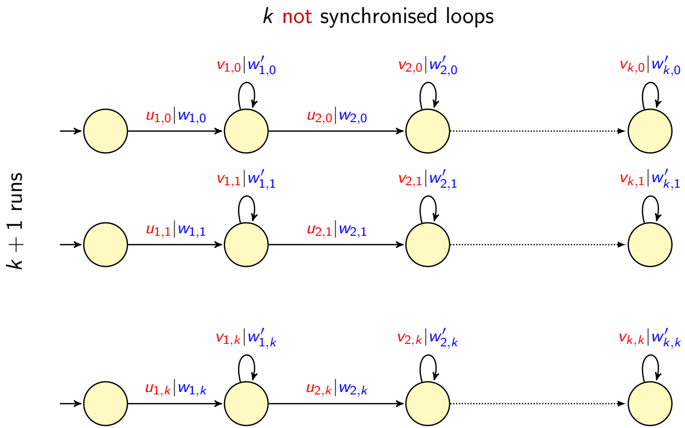
- *there exist DFTs $T_1, \ldots, T_k$ s.t. $T \equiv \bigcup_i T_i$*
- *T satisfies the branching twinning property of order k*
- *$[\![T]\!]$ satisfies the k-Lipschitz property*

Theorem

The k-sequentiality Problem is PSpace-complete (k given in unary).

Branching twinning property of order k

For all situations like:

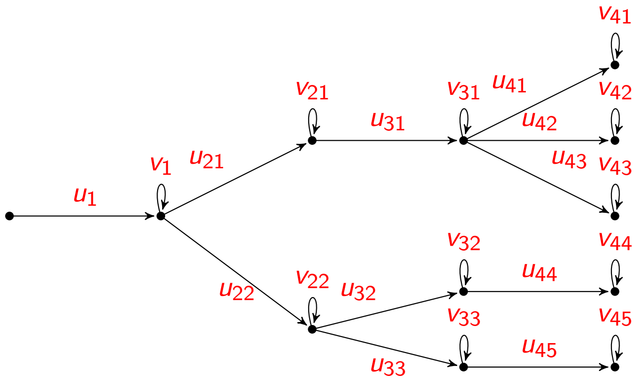


there are two runs $0 \leq i < j \leq k$ s.t. for every loop ℓ with same input words,

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 fNFT is definable by a union of k DFT iff it satisfies the BTP of order k.

Branching twinning property of order k

Theorem

A fNFT is definable by a union of k DFT iff it satisfies the BTP of order k.

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 union of k DFT iff it satisfies the BTP of order k.

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

Overview

Summary

- Streamability/efficient evaluation of transformations
- Open problems on word transducers
- Removing two-wayness: decidable, now in 2ExpSpace
- Removing non-determinism: k-sequentiality PSpace-complete

Summary

- Streamability/efficient evaluation of transformations
- Open problems on word transducers
- Removing two-wayness: decidable, now in 2ExpSpace
- Removing non-determinism: k-sequentiality PSpace-complete

Extensions:

- logical presentation of transductions
- functional $\rightsquigarrow$ finite-valued
- transducers $\rightsquigarrow$ weighted automata on semigroups ($+$ hypotheses)

Summary

- Streamability/efficient evaluation of transformations
- Open problems on word transducers
- Removing two-wayness: decidable, now in 2ExpSpace
- Removing non-determinism: k-sequentiality PSpace-complete

Extensions:

- logical presentation of transductions
- functional $\rightsquigarrow$ finite-valued
- transducers $\rightsquigarrow$ weighted automata on semigroups ($+$ hypotheses)

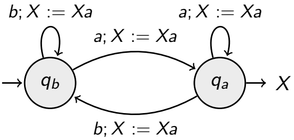
Growing interest for transducers:

- ~~LIF~~ LIS
- LaBRI (2Way to 1Way)
- LSV!
- Warsaw (origin semantics)
- Microsoft (symbolic transducers)
- UPenn (streaming string transducers)

Streaming String Transducers

Definition ([AC10])

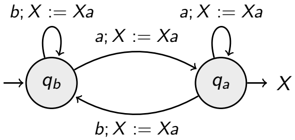
Streaming String Transducer = DFA
+ finite set of registers X, Y, Z



Streaming String Transducers

Definition ([AC10])

Streaming String Transducer = DFA
+ finite set of registers X, Y, Z



➜ allows to recover previously defined classes of functions:

k-sequential functions

k registers $X_1, \ldots, X_k$
$X_i := X_i u$

Rational functions

$X := Yu$

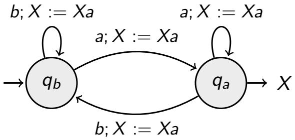
Regular functions

$X := uYv$
$X := YZ$
linear updates

Streaming String Transducers

Definition ([AC10])

Streaming String Transducer = DFA
+ finite set of registers X, Y, Z



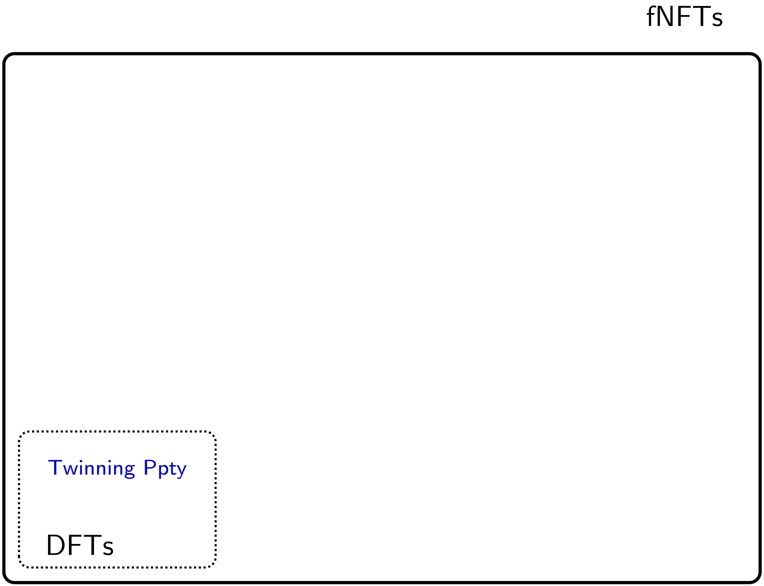
➜ allows to recover previously defined classes of functions:

k-sequential functions	Rational functions	Regular functions
k registers $X_1, \ldots, X_k$ $X_i := X_i u$	$X := Yu$	$X := uYv$ $X := YZ$ linear updates

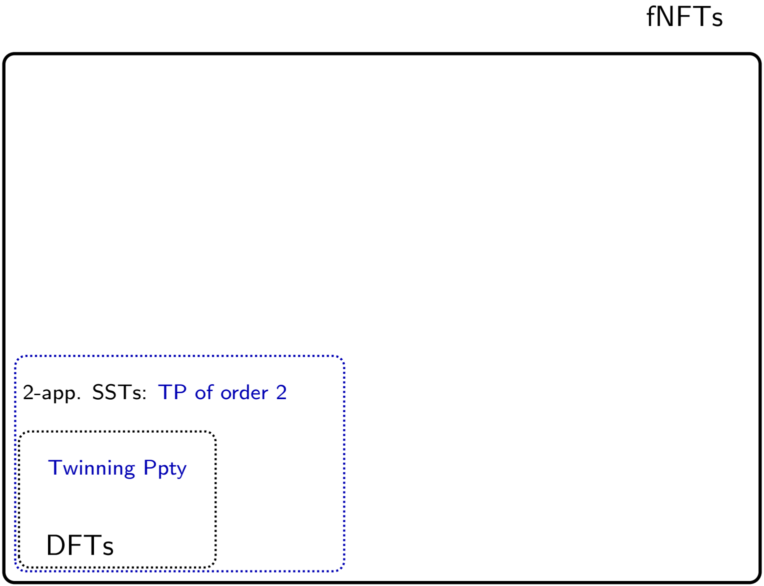
Theorem

One can minimize the number of registers in the class of rational functions.

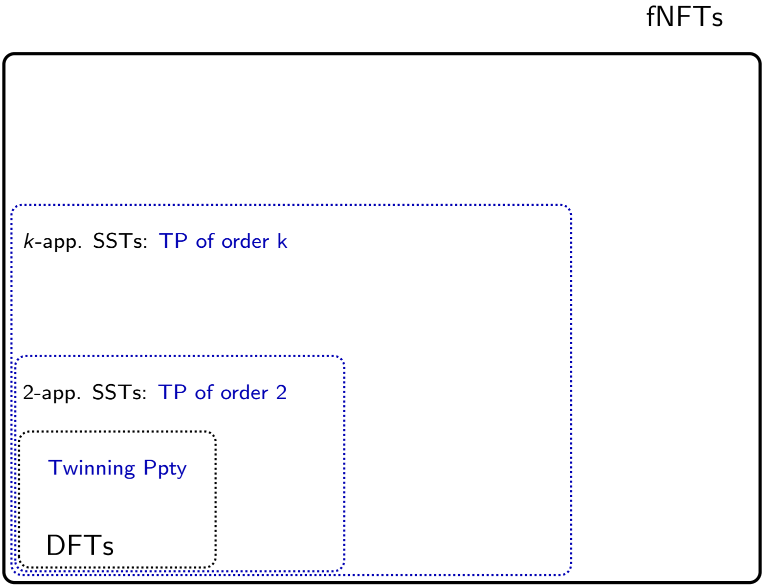
Landscape of rational functions



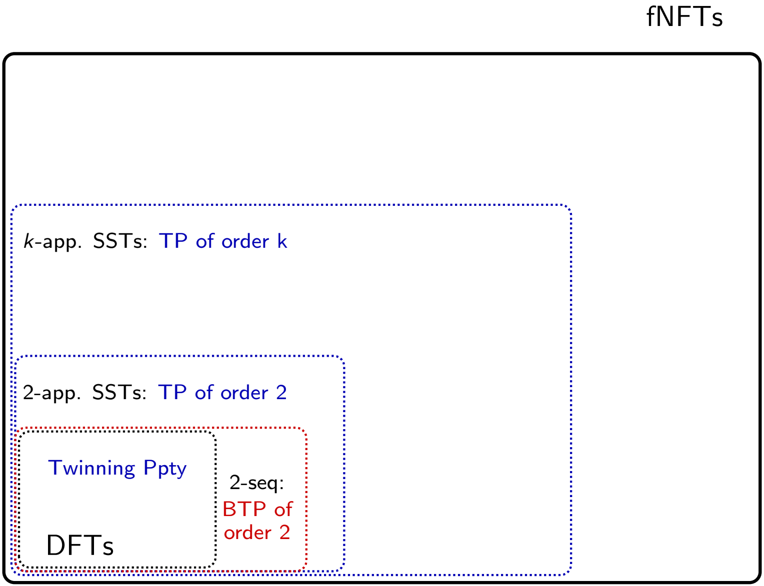
Landscape of rational functions



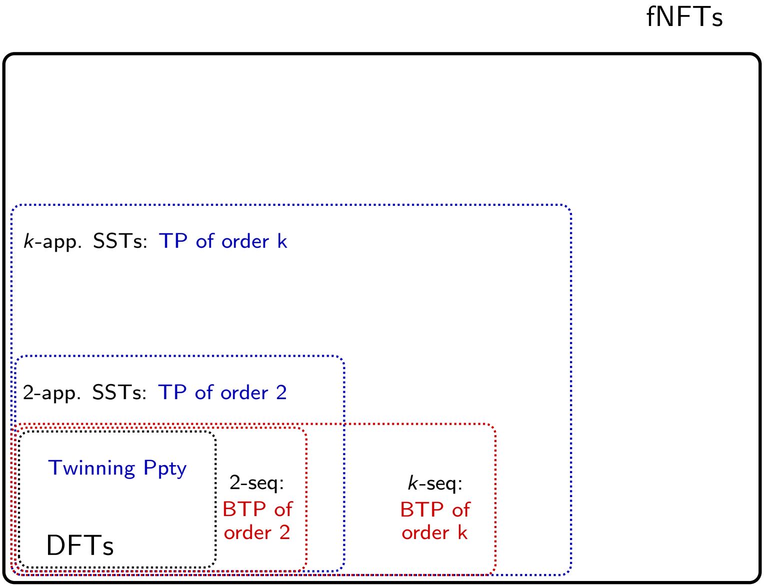
Landscape of rational functions



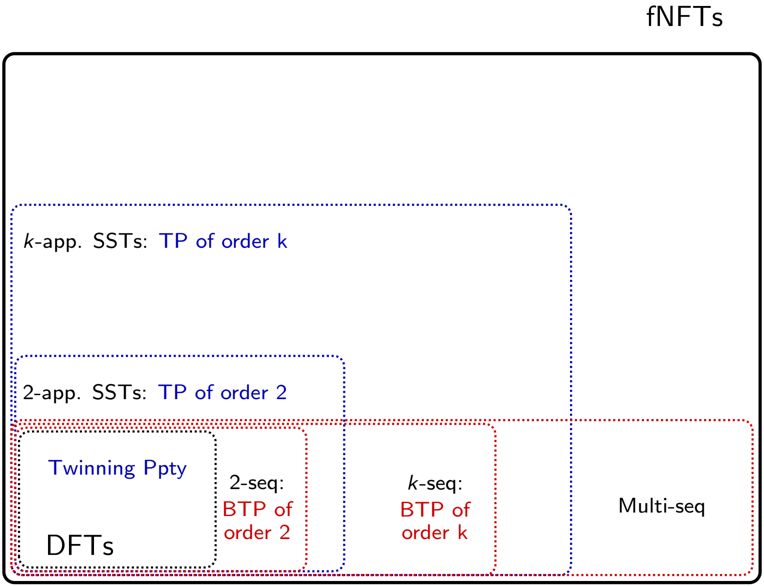
Landscape of rational functions



Landscape of rational functions



Landscape of rational functions



Some perspectives

- Register minimization for regular functions
  ➜ deal with prepending of words ($X := uYv$)
  ➜ deal with concatenation of registers ($X := YZ$)
- Algebraic characterizations (bimachines [RS91])
- Specification formalism
- Tool development
- Symbolic transducers
- Back to nested words!

Some perspectives

- Register minimization for regular functions
  ➜ deal with prepending of words ($X := uYv$)
  ➜ deal with concatenation of registers ($X := YZ$)
- Algebraic characterizations (bimachines [RS91])
- Specification formalism
- Tool development
- Symbolic transducers
- Back to nested words!

Thanks!