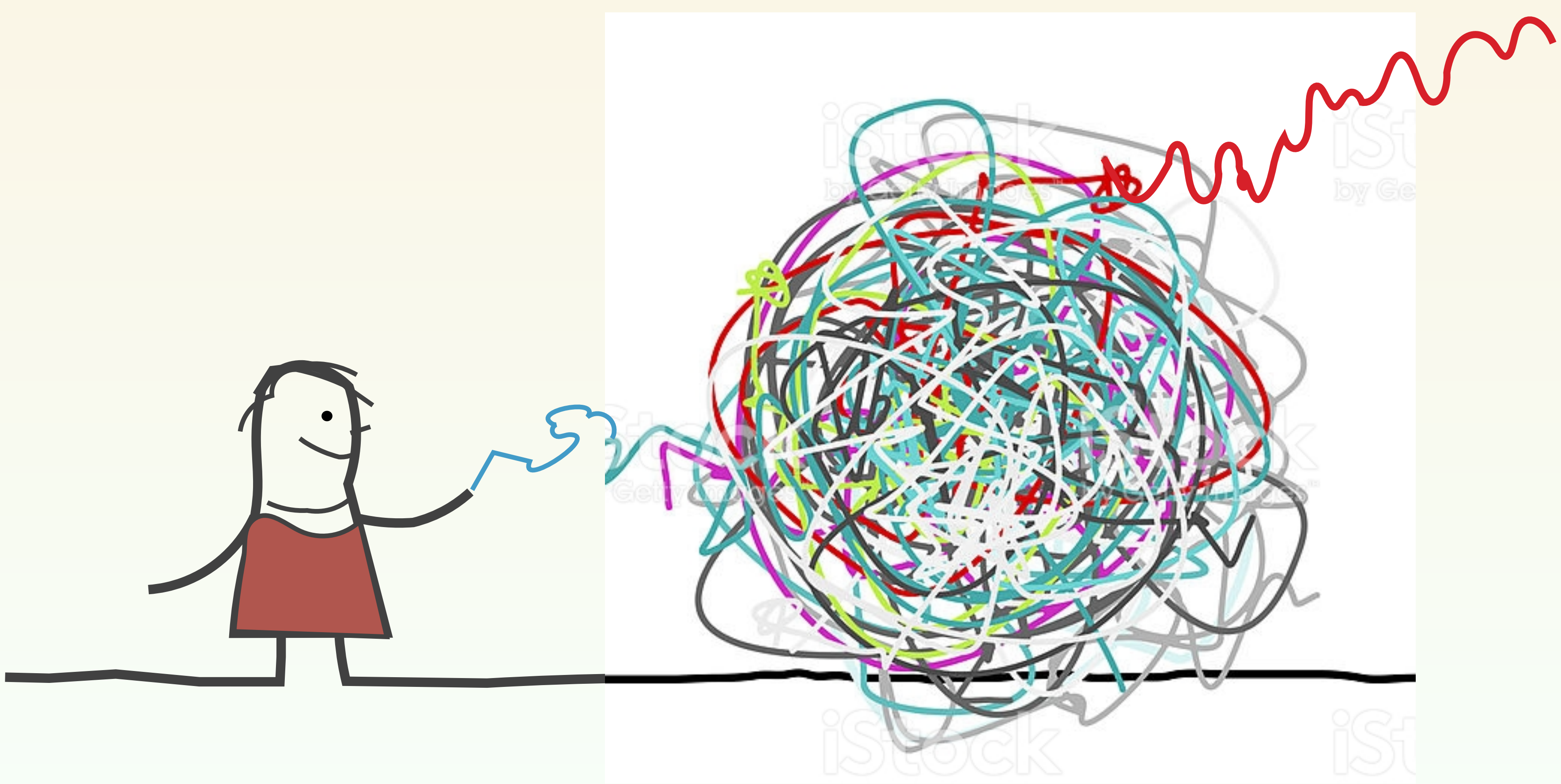
A View on String Transducers

Anca Muscholl



Transductions: some history

Early notion in formal language theory, motivated by coding theory, compiling, linguistics…

Moore 1956 "Gedankenexperiments on sequential machines"

Church 1957, Schützenberger 1961, Ginsburg-Rose 1966, Nivat 1968, Aho-Hopcroft-Ullman 1969, Engelfriet 1972, Eilenberg 1976, Choffrut 1977, Berstel 1979.

Word transducers are weighted automata over the language semi-ring (sum=union, product=concatenation)

Overview

- Word transductions
  - Büchi
  - Kleene
  - Schützenberger
- Equivalence problem
  - Culik-Karkumäki
  - Ehrenfeucht & Hilbert
  - Bojańczyk

Transducers

transform objects - here: words

transduction: mapping (or relation) from words to words

metamorphosis → mtmrphss erase vowels

metamorphosis → sisohpromatem mirror

metamorphosis → metamorphosismetamorphosis duplicate

metamorphosis → phosismetamor permute circularly

Overview

Word transductions

- automata = logic
- translations between models
- expressions
- algebra?

Equivalence problem

- finitely valued transducers
- Ehrenfeucht & Hilbert
- origin equivalence

Transducers

one-way (non-)deterministic finite-state transducers

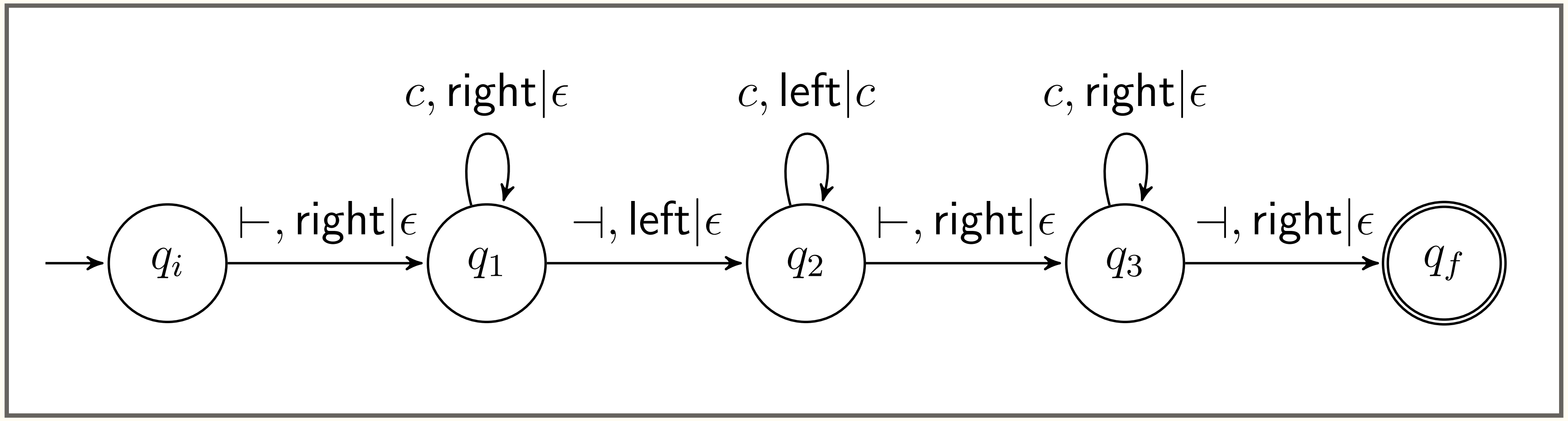
metamorphosis → mtmrphss   erase vowels

two-way (non-)deterministic finite-state transducers

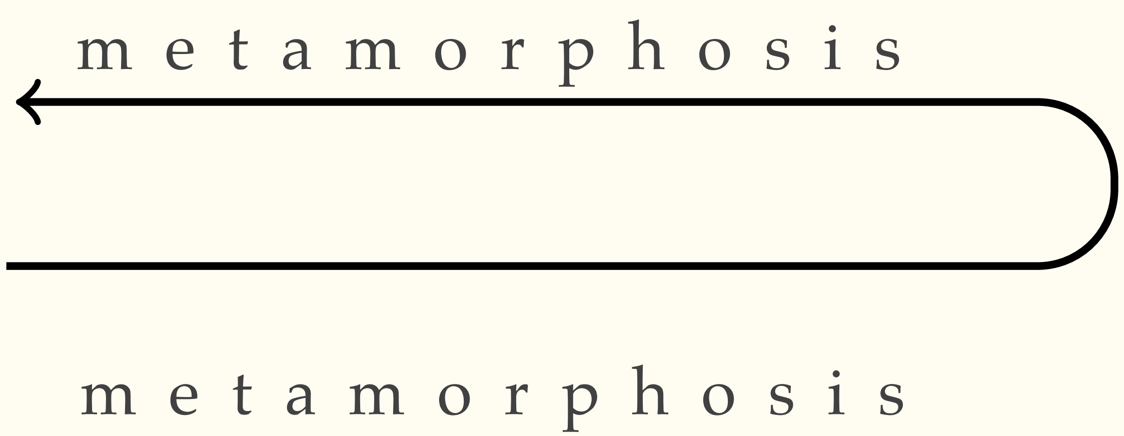
metamorphosis → sisohpromatem   mirror

metamorphosis → metamorphosismetamorphosis   doubling

Transduction: binary relation over words



deterministic, 2-way transducer computing the mirror



metamorphosis ⟶ sisohpromatem

Logic

MSOT: monadic second-order transductions

[Courcelle, Engelfriet]

maps structures into structures

- output consists of fixed number of copies of input positions

- domain formula: unary MSO formula

  "*c*-th copy of input position **x** occurs in the output and is labeled by symbol *a*"

- order formula: binary MSO formula

  "*c*-th copy of input position **x** precedes *d*-th copy of input position **y** in the output"

Logic

MSOT: monadic second-order transductions

Example: $w \mapsto w\,w$

- 2 copies
- domain formula: $\mathrm{dom}_{a,1}(x) = \mathrm{dom}_{a,2}(x) \equiv a(x)$
- order formula: $\mathrm{before}_{1,1}(x,y) = \mathrm{before}_{2,2}(x,y) \equiv (x < y)$

$$\mathrm{before}_{1,2}(x,y) \equiv \mathrm{true}$$

Logic

MSOT: monadic second-order transductions

Example: $w \mapsto w\,w$

- 2 copies
- domain formula: $\mathrm{dom}_{a,1}(x) = \mathrm{dom}_{a,2}(x) \equiv a(x)$
- order formula: $\mathrm{before}_{1,1}(x,y) = \mathrm{before}_{2,2}(x,y) \equiv (x < y)$

  $\mathrm{before}_{1,2}(x,y) \equiv \mathrm{true}$

MSOT = deterministic, two-way transducers (2DFT)

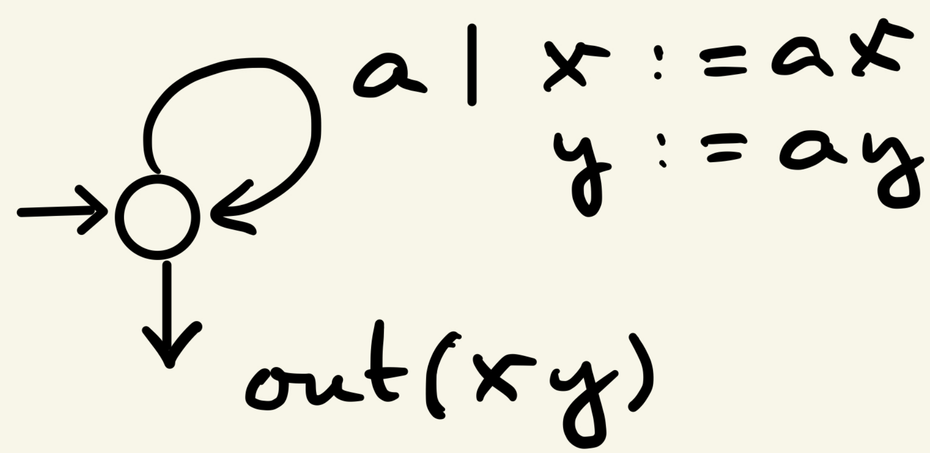
[Engelfriet-Hoogeboom 2001]

Transducers with registers

SST: streaming string transducers [Alur-Cerny 2010]

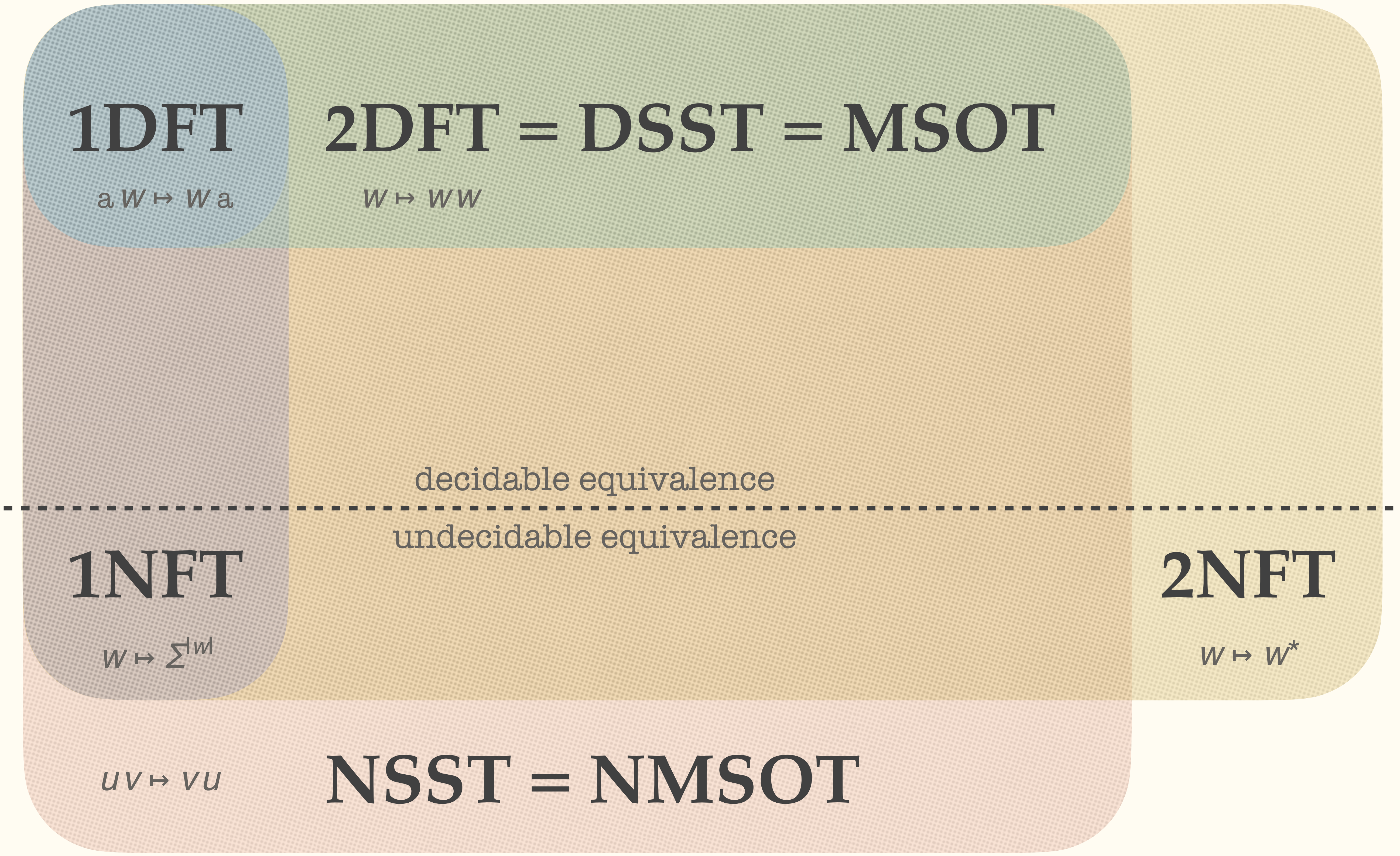
- one-way automata +
- finite number of registers: output can be appended left or right, registers can be concatenated

doubling



DSST: deterministic copyless streaming string transducers = MSOT

Landscape with transducers



Functions

A transducer is single-valued if there is at most one output per input word

here: transductions are functions from words to words

2DFT = DSST = NSST = MSOT

regular word functions

[Engelfriet-Hoogeboom 2001]

[Alur-Cerny 2010]

Overview

- Word transductions
  - automata = logic
  - translations between models
  - expressions
  - algebra?
- Equivalence problem
  - finitely valued transducers
  - Ehrenfeucht & Hilbert
  - origin equivalence

Word functions

poly

sequential word functions: 1DFT (one-way, deterministic)

rational word functions: 1NFT (one-way, non-deterministic)

regular word functions: 2DFT = 2NFT = DSST = NSST = MSOT

non-elementary

[Filiot, Gauwin, Reynier, Servais 2013]

Regular vs rational

Given a single-valued two-way transducer T:

- it is decidable in ExpSPACE whether an equivalent one-way transducer exists
- if "yes": construction of 2-exp size equivalent one-way transducer

[Baschenis, Gauwin, M., Puppis 2017]

Lower bounds

- PSPACE for the decision procedure
- the size of the one-way transducer is optimal

Remark: undecidable for relations

Given a single-valued two-way transducer T, the existence of an equivalent one-way transducer is decidable in ExpSPACE.

Example: $w \mapsto w\ w$ with $w \in R$

- if $R = (a+b)^*$: no equivalent one-way transducer
- if $R = (ab)^*$: an equivalent one-way transducer exists

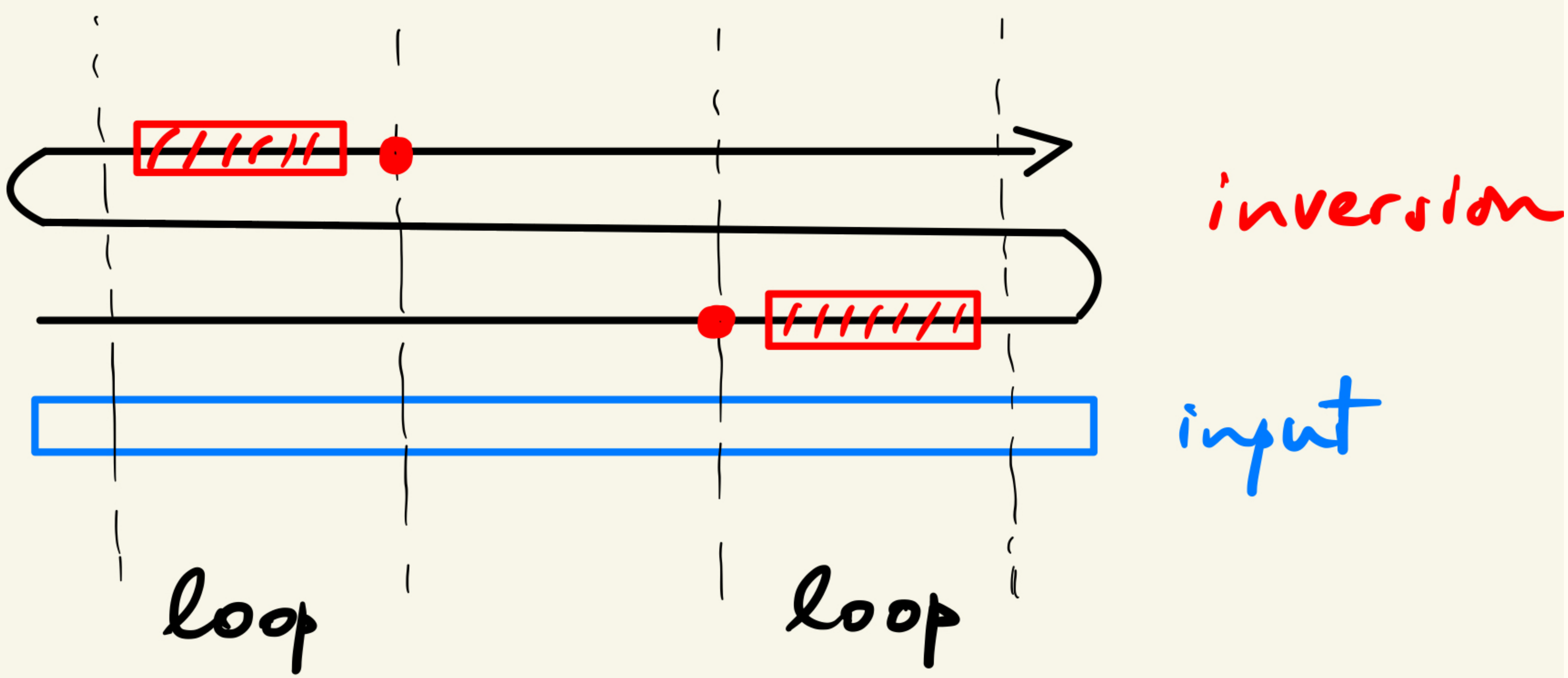
Remark: doubling and mirror are typically two-way

Given a single-valued two-way transducer T, the existence of an equivalent one-way transducer is decidable in ExpSPACE.

Key tool: inversions + word combinatorics

Given a single-valued two-way transducer T, the existence of an equivalent one-way transducer is decidable in ExpSPACE.

Key tool: inversions + word combinatorics

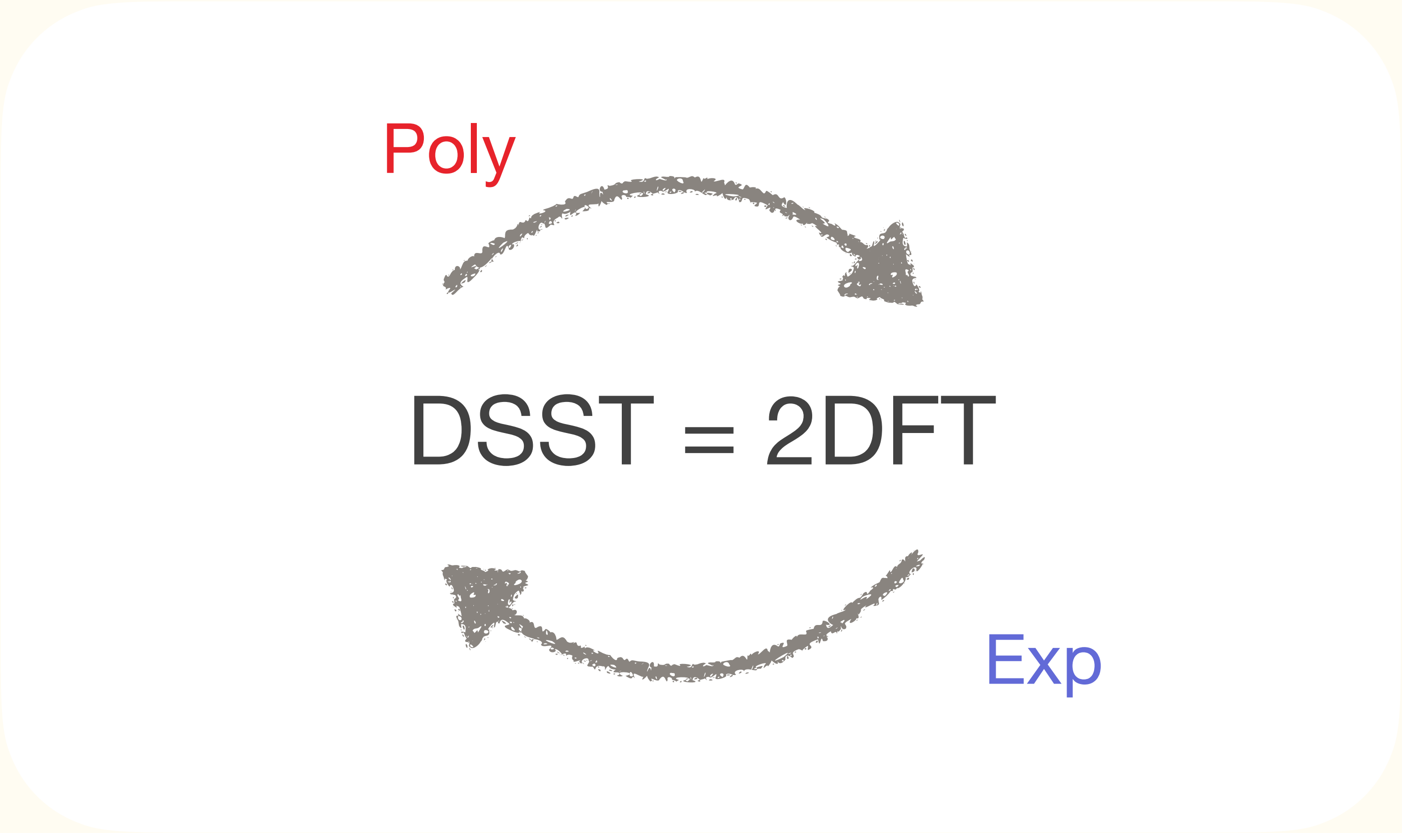


The output between the red dots has exponentially-bounded period

Weighted automata: One-way vs Two-way

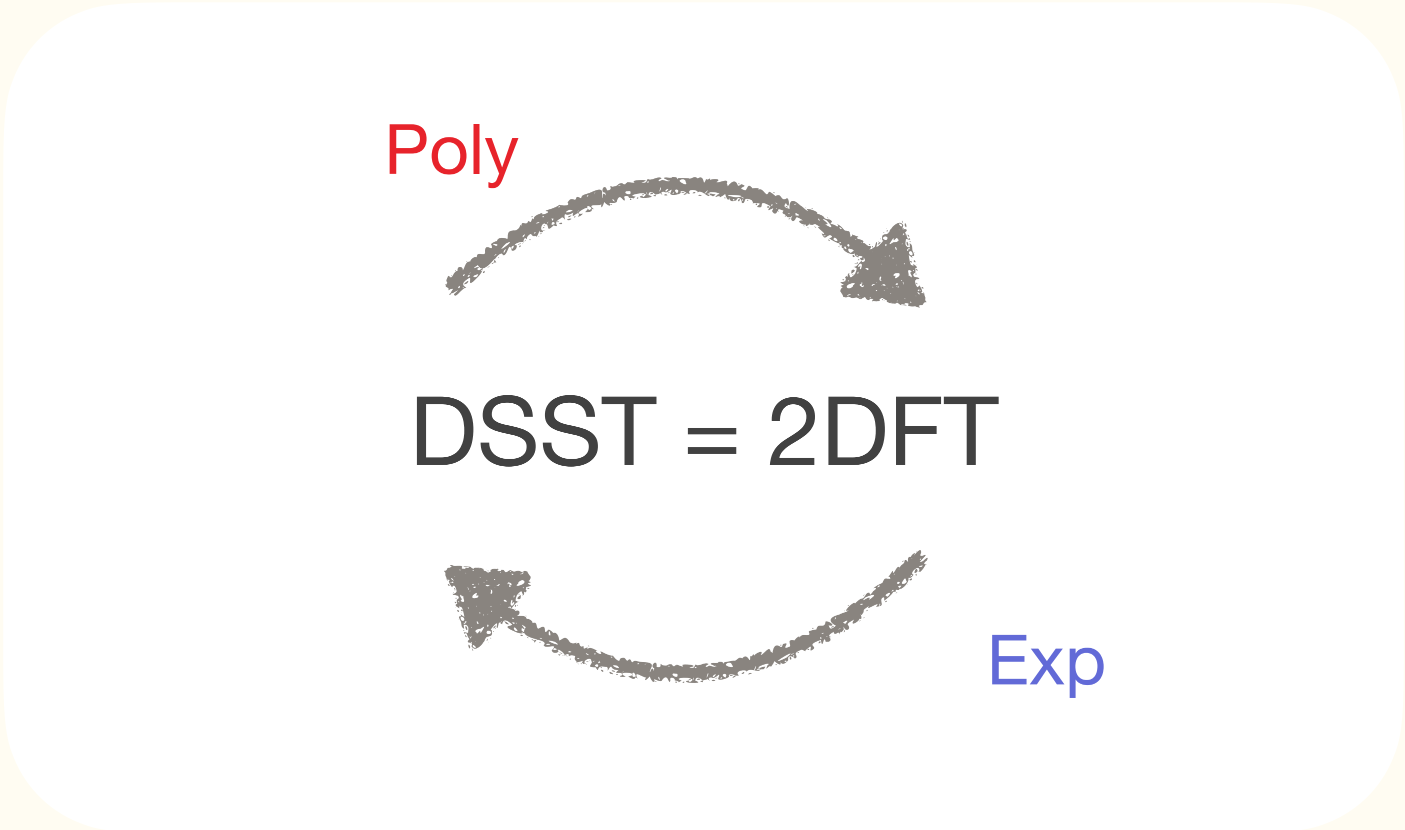
- Two-way computation adds power
- It can be decided if a two-way WA is equivalent to a one-way WA (over commutative semiring) [Anselmo 1990]
- One-way WA = restricted weighted MSO logic [Droste, Gastin 2009]
- Two-way WA with pebbles = weighted FO logic + transitive closure [Bollig, Gastin, Monmege, Zeitoun 2014]

Translations



- a one-way transducer **T** labels the input word by the accepting run of the DSST
- a 2DFT can build the output from the annotated input …

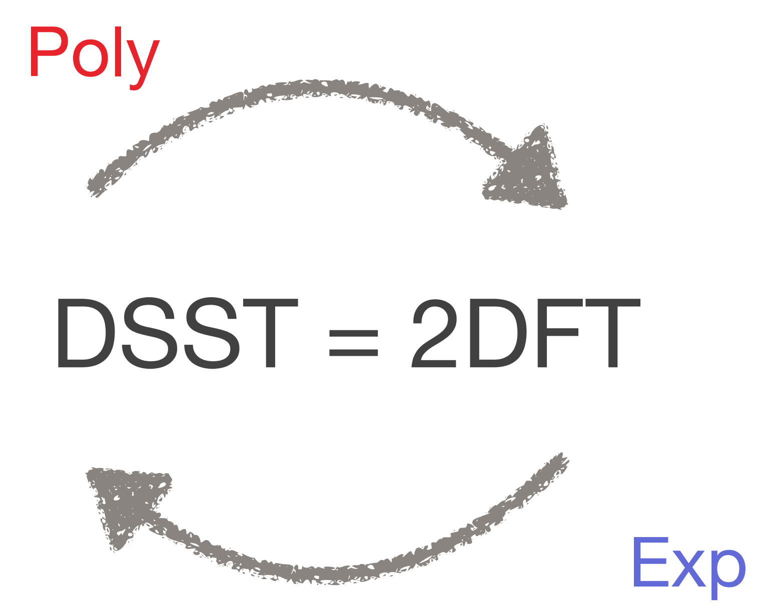
Translations



- a one-way transducer **T** labels the input word by the accepting run of the DSST
- a 2DFT can build the output from the annotated input …

… if **T** is reversible, so co-deterministic (and deterministic)

Translations



Deterministic one-way transducers can be simulated by reversible two-way transducers with quadratic blow-up.

[Dartois, Fournier, Jecker, Lhote 2017]

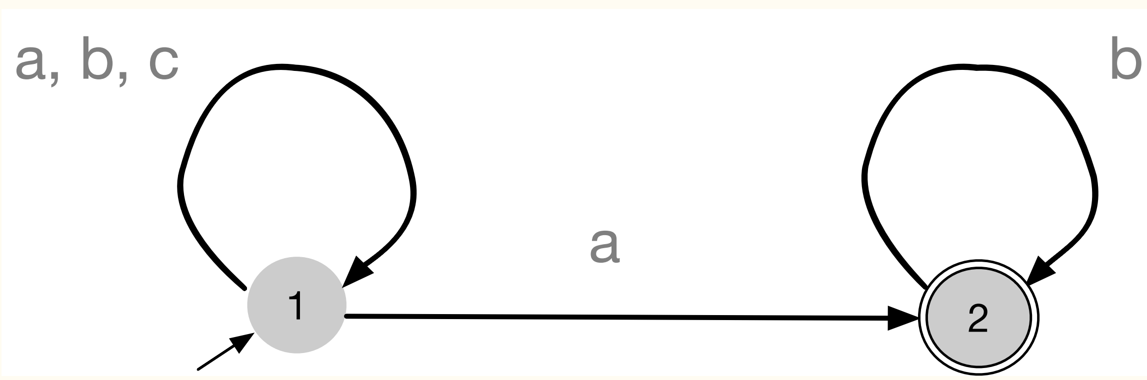
reversible = deterministic and co-deterministic

Reversible computations

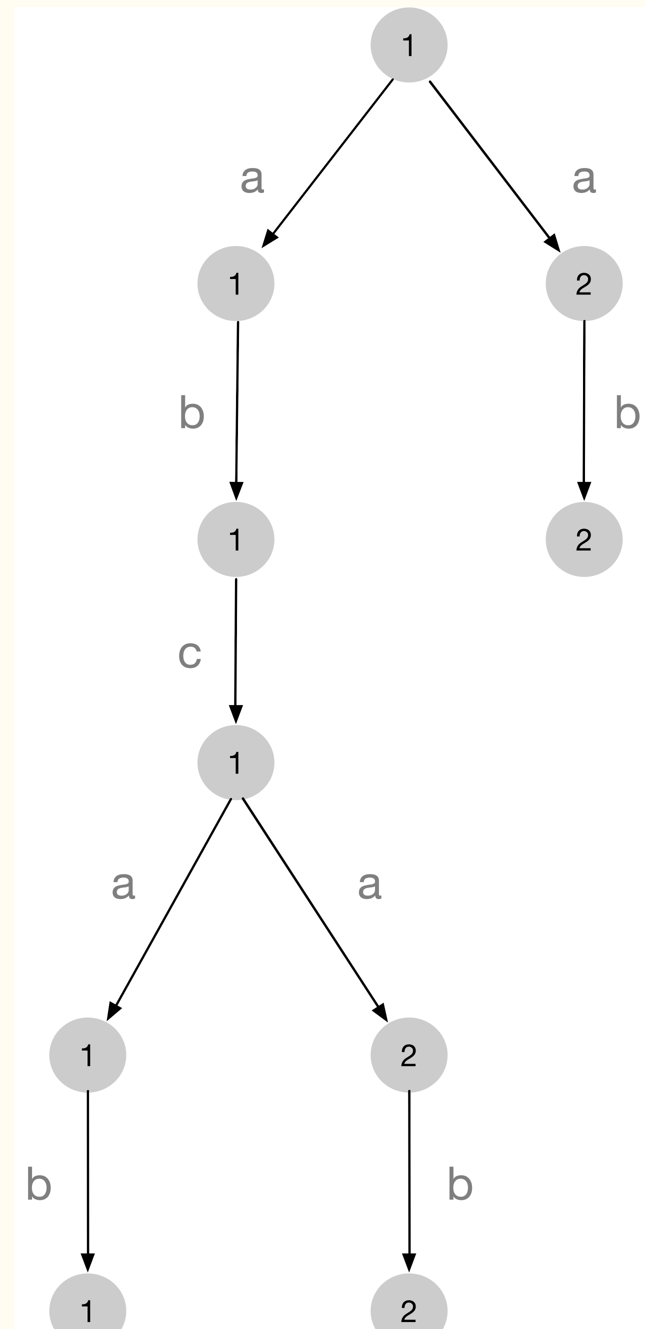
reversible: deterministic and co-deterministic

DFS of computation tree of co-deterministic one-way automata

[Hopcroft-Ullman'67, Sipser'78]



a b c a b

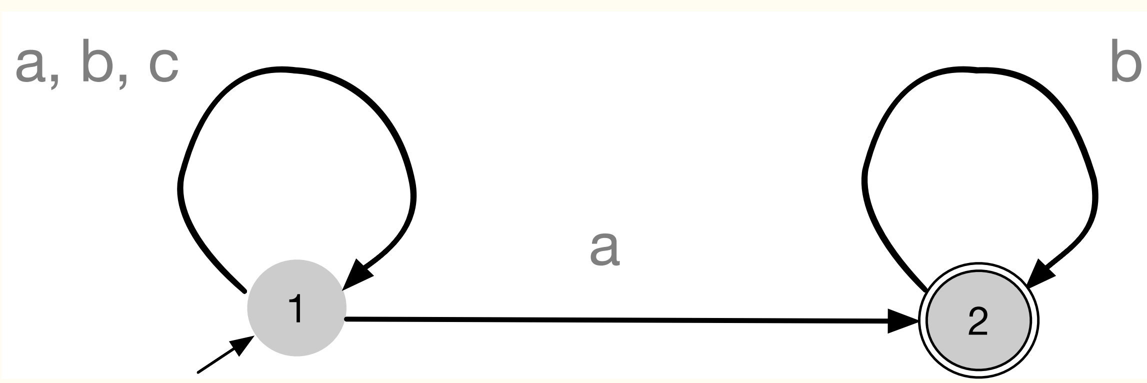


Reversible computations

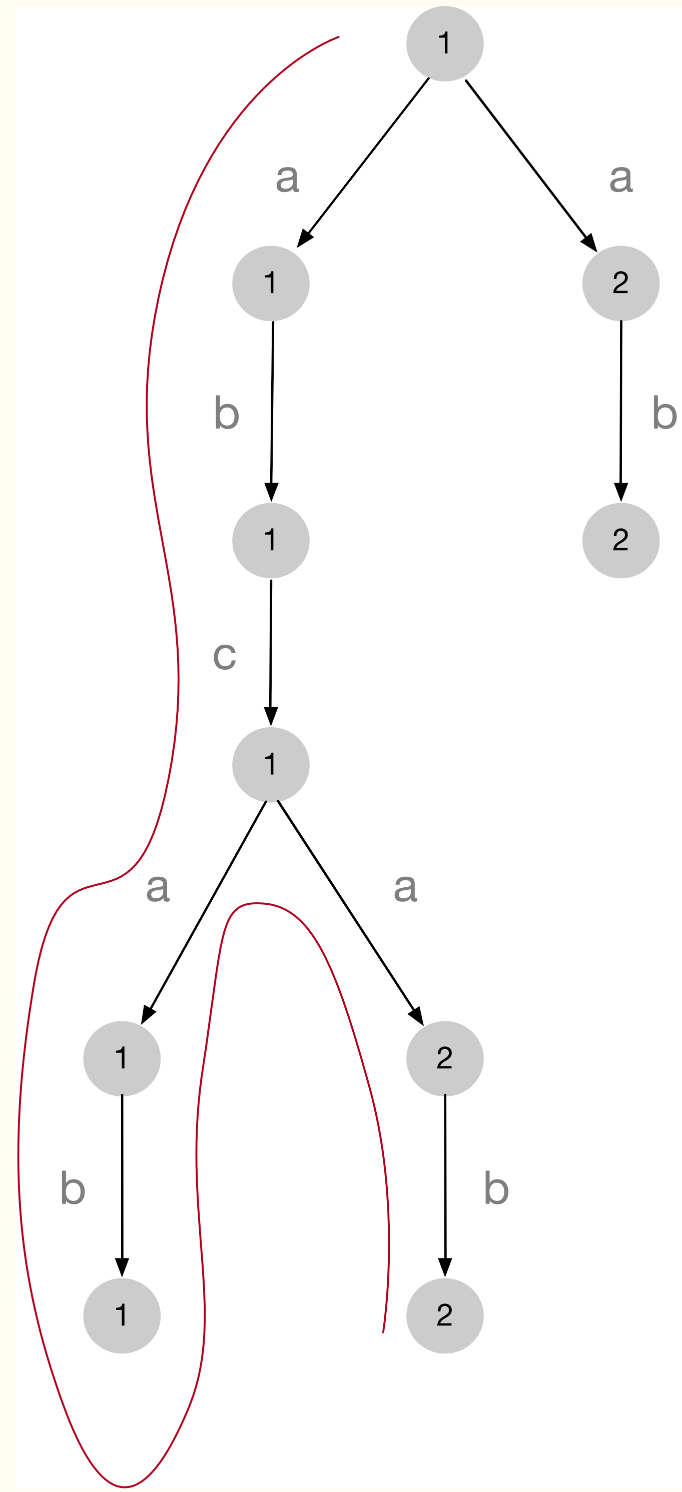
reversible: deterministic and co-deterministic

DFS of computation tree of co-deterministic one-way automata

reversible



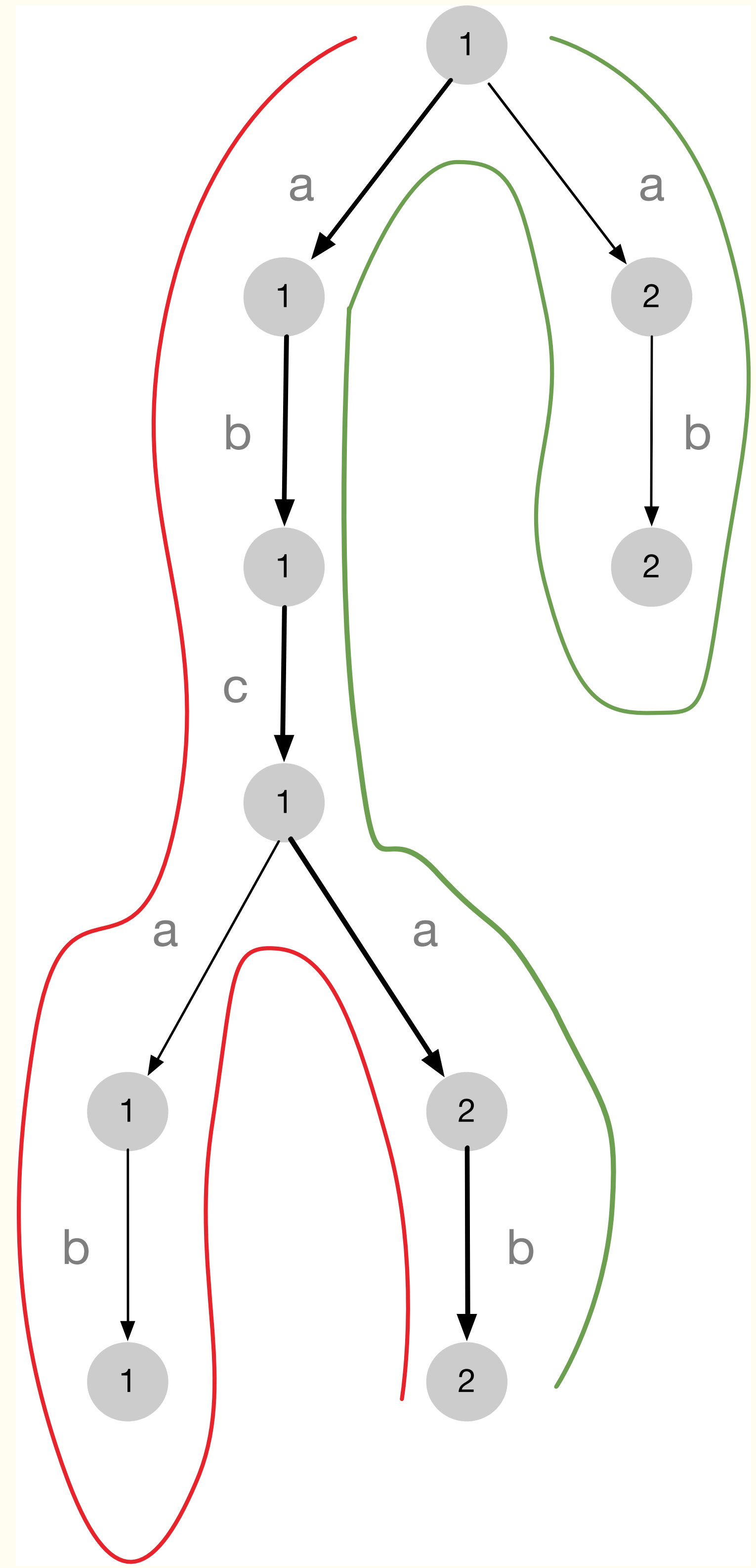
a b c a b



Reversible transducers

reversible: deterministic and co-deterministic

Computation tree of co-deterministic transducers

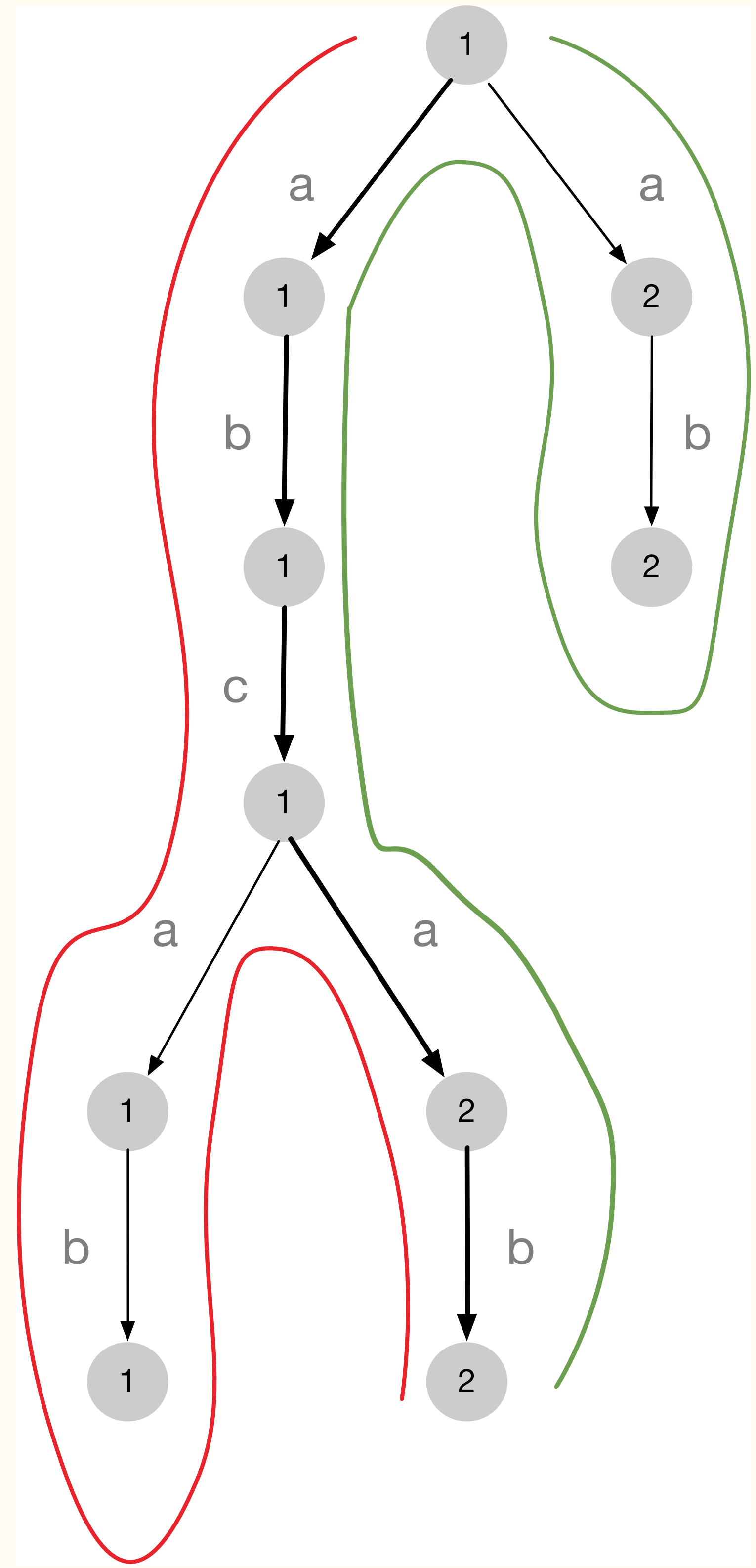


Reversible transducers

reversible: deterministic and co-deterministic

Computation tree of co-deterministic transducers

When to produce the output?



Reversible transducers

reversible: deterministic and co-deterministic

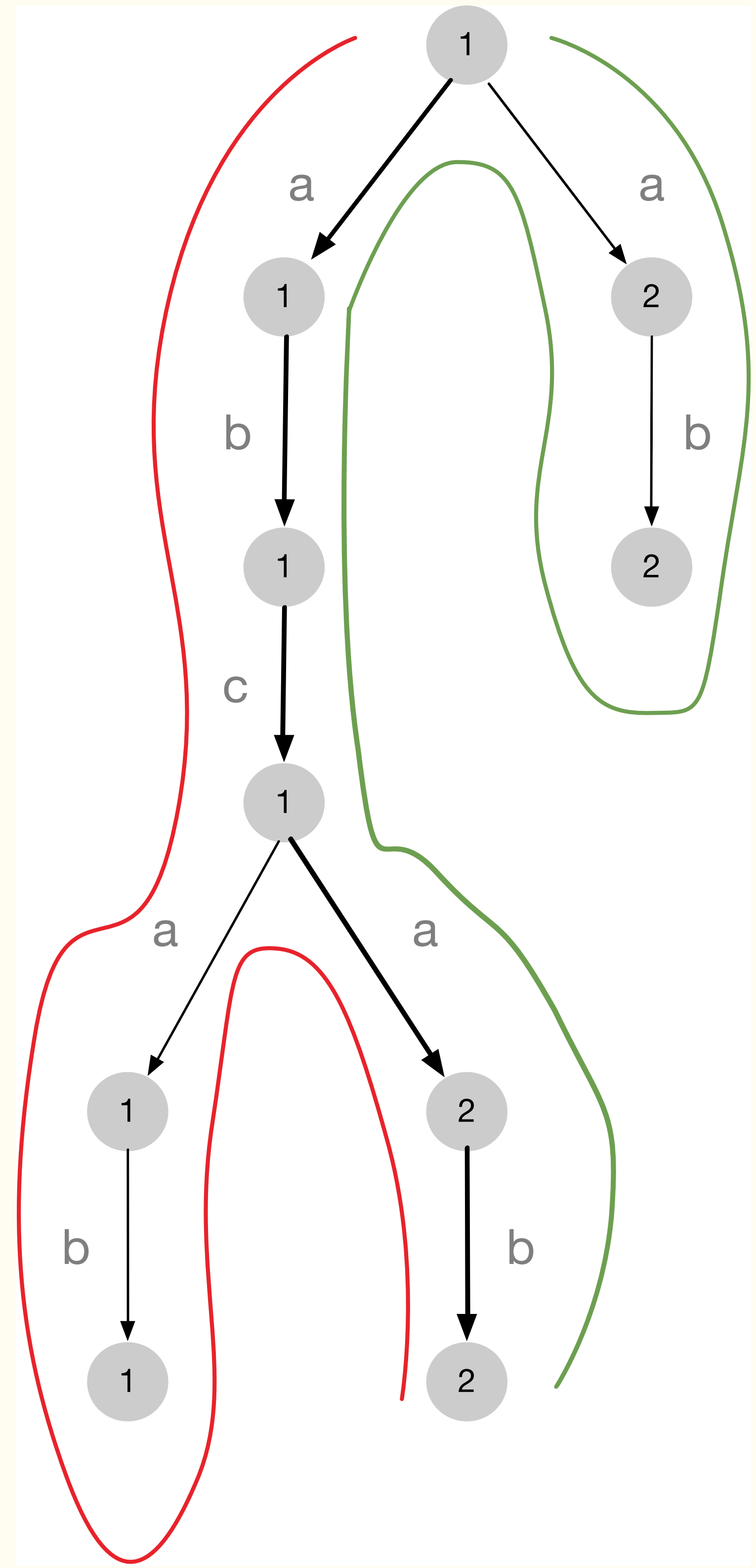
Computation tree of co-deterministic transducers

When to produce the output?

Double DFS "surrounding" accepting run

1DFT can be made reversible with quadratic blow-up

[Dartois, Fournier, Jecker, Lhote'17]



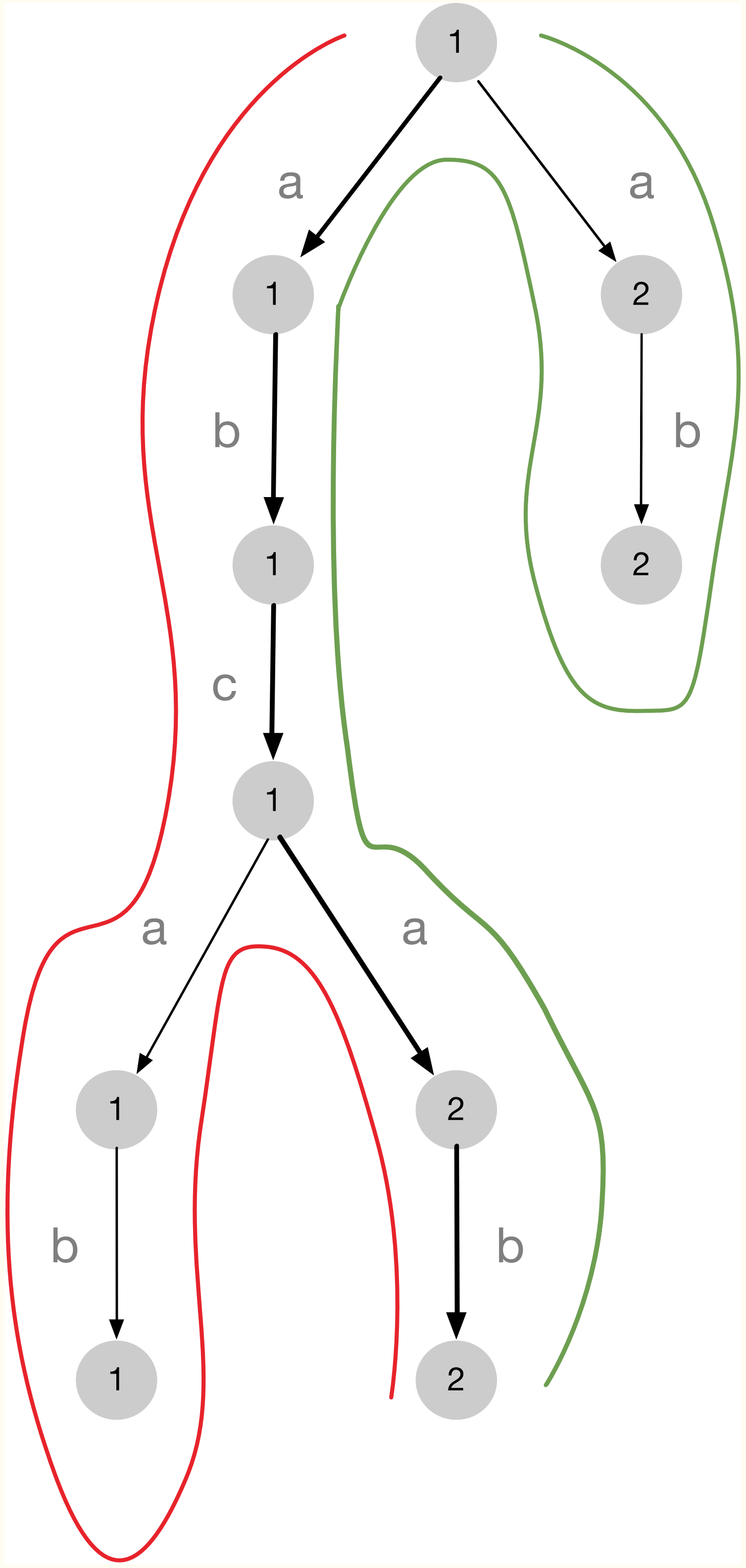
Reversible transducers

reversible: deterministic and co-deterministic

Computation tree of co-deterministic transducers

When to produce the output?

Double DFS "surrounding" accepting run



2DFT can be made reversible with exponential blow-up

[Dartois, Fournier, Jecker, Lhote'17]

Some open problems

- PSPACE lower bound for decision procedure “two-way transducers to one-way” - better lower bound?
- Better complexity for “two-way to deterministic one-way”?
- Extension from single-valued transducers to finitely-valued ones?

Overview

Word transductions

- automata = logic
- translations between models
- expressions
- algebra?

Equivalence problem

- finitely valued transducers
- Ehrenfeucht & Hilbert
- origin equivalence

Rational expressions

One-way (rational) string functions equivalent to simple expressions $f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^*$ (unambiguous)

2DFT equivalent to rational expressions plus

- composition
- mirror transduction
- doubling transduction or Hadamard product

[Gastin'19, Dave, Gastin, Krishna'18, Alur et al.'14]

Polyregular word functions

[Bojanczyk'18]

Polyregular word functions: functions computed by

- two-way deterministic transducers with pebbles
- smallest class of functions containing functions computed by 1DFT, and closed under composition, iterated reverse and squaring
- for-transducers, polynomial list functions
- MSO interpretations

[Bojanczyk, Kiefer, Lhote'19]

squaring abcd ⟶ **a**bcd a**b**cd ab**c**d abc**d**

Algebra

Long line of research on algebra for regular languages:

- algebra offers machine-independent characterizations, canonical objects (minimization), decision procedures for subclasses
- prominent example: decide whether a regular language is star-free

star-free = aperiodicity [Schützenberger'65]

star-free = first-order logic [McNaughton, Papert'71]

Algebra for transducers?

- A Myhill-Nerode theorem for deterministic one-way transducers …

  [Choffrut'79]

  … thus a canonical (minimal) transducer

Algebra for transducers?

- A Myhill-Nerode theorem for deterministic one-way transducers …

[Choffrut'79]

… thus a canonical (minimal) transducer

- Non-deterministic one-way transducers

Any one-way transducer is equivalent to the composition of a deterministic left2right and a deterministic right2left transducer

[Elgot, Mezei'65]

Bimachine: as above + output

For every one-way transducer there is a family of canonical bimachines.

[Reutenauer, Schützenberger'91]

First-order transductions

- One-way transducers: equivalent to order-preserving MSOT

Decidable whether a one-way transducer is equivalent to some order-preserving first-order transduction.

[Filiot,Gauwin,Lhote'16]

proof uses canonical bimachines

First-order transductions

- One-way transducers: equivalent to order-preserving MSOT

Decidable whether a one-way transducer is equivalent to some order-preserving first-order transduction.

[Filiot,Gauwin,Lhote'16]

proof uses canonical bimachines

- Two-way transducers: no decision procedure for first-order transductions so far, but …

First-order transductions = aperiodic two-way transducers = aperiodic streaming transducers

[Carton, Dartois'15], [Filiot, Krishna, Trivedi'15]

More open problems

Can we decide whether a regular word function is aperiodic / first-order definable?

Same for polyregular word functions.

Overview

Word transductions

- automata = logic
- translations between models
- expressions
- algebra?

Equivalence problem

- finitely valued transducers
- Ehrenfeucht & Hilbert
- origin equivalence

Equivalence problem

Given two transducers, do they compute the same relation?

Equivalence of non-deterministic one-way transducers (1NFT) is undecidable.

[Fischer-Rosenberg, Griffiths'68]

Equivalence problem

Given two transducers, do they compute the same relation?

Equivalence of non-deterministic one-way transducers (1NFT) is undecidable.

[Fischer-Rosenberg, Griffiths'68]

PCP h, g : A* maps-to B*

[Ibarra]

inputs from A* $ B*, output alphabet c

Equivalence problem

Given two transducers, do they compute the same relation?

Equivalence of non-deterministic one-way transducers (1NFT) is undecidable.

[Fischer-Rosenberg, Griffiths'68]

PCP h, g : A* maps-to B*

[Ibarra]

inputs from A* $ B*, output alphabet c

Take h. On input u $ z output

either a number of c's different from |h(u)| (and smaller than |h(u)z|)

or, for some u = u' a u" and z = z' t z": number of c = |h(u'a)z"|, t not= h(a)

Same for g. Equivalence with "universal" 1NFT iff PCP has a no solution.

Equivalence problem

Given two transducers, do they compute the same relation?

Equivalence of non-deterministic one-way transducers (1NFT) is undecidable.

[Fischer-Rosenberg, Griffiths'68]

linearly ambiguous

[Ibarra]

PCP   h, g : A* maps-to B*

inputs from A* $ B*, output alphabet c

Take h. On input u $ z output

either a number of c's different from |h(u)| (and smaller than |h(u)z|)

or, for some u = u' a u" and z = z' t z": number of c = |h(u'a)z"|, t not= h(a)

Same for g. Equivalence with "universal" 1NFT iff PCP has a no solution.

Equivalence problem

Given two transducers, do they compute the same relation?

Equivalence of non-deterministic one-way transducers (1NFT) is undecidable.

[Fischer-Rosenberg, Griffiths'68]

decidable

undecidable

2DFT

2DSST

copyful 2DSST

1NFT

PSPACE-c

[Gurari'82]

PSPACE

[Alur-Cerny'10]

decidable

[Filiot-Reynier'17]

[Benedikt et al.'17]

Equivalence problem

Single-valued transducer: at most one output per input word

Equivalence problem

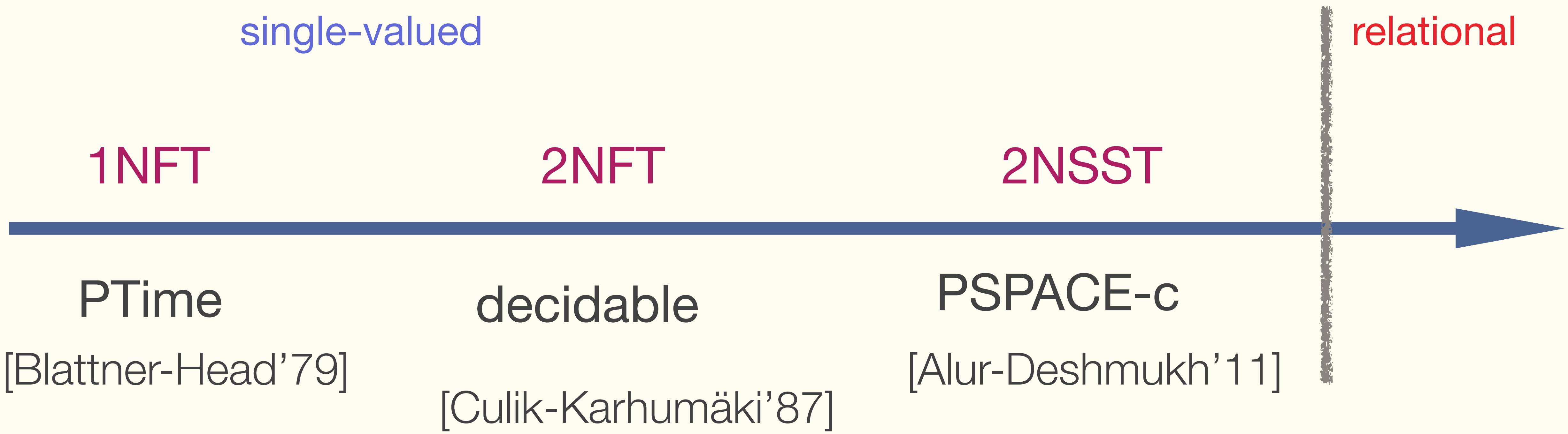
Single-valued transducer: at most one output per input word

To check equivalence, single-valuedness is as good as determinism!

Equivalence problem

Single-valued transducer: at most one output per input word

To check equivalence, single-valuedness is as good as determinism!



PSPACE-c: equivalence of 2NFA
is in PSPACE [Vardi'89]

Equivalence problem

Single-valued transducer: at most one output per input word

single-valuedness is not yet the end for equivalence problem

Equivalence problem

Single-valued transducer: at most one output per input word

single-valuedness is not yet the end for equivalence problem

k-valued transducer: for every input word at most k outputs
finitely-valued transducer: k-valued for some k

Equivalence of k-valued 1DFT (and 2DFT) is decidable.

[Culik-Karhumäki'86]

Overview

Word transductions

- automata = logic
- translations between models
- expressions
- algebra?

Equivalence problem

- finitely valued transducers
- Ehrenfeucht & Hilbert
- origin equivalence

Finitely valued

k-valued transducer: for every input at most k different outputs

Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]

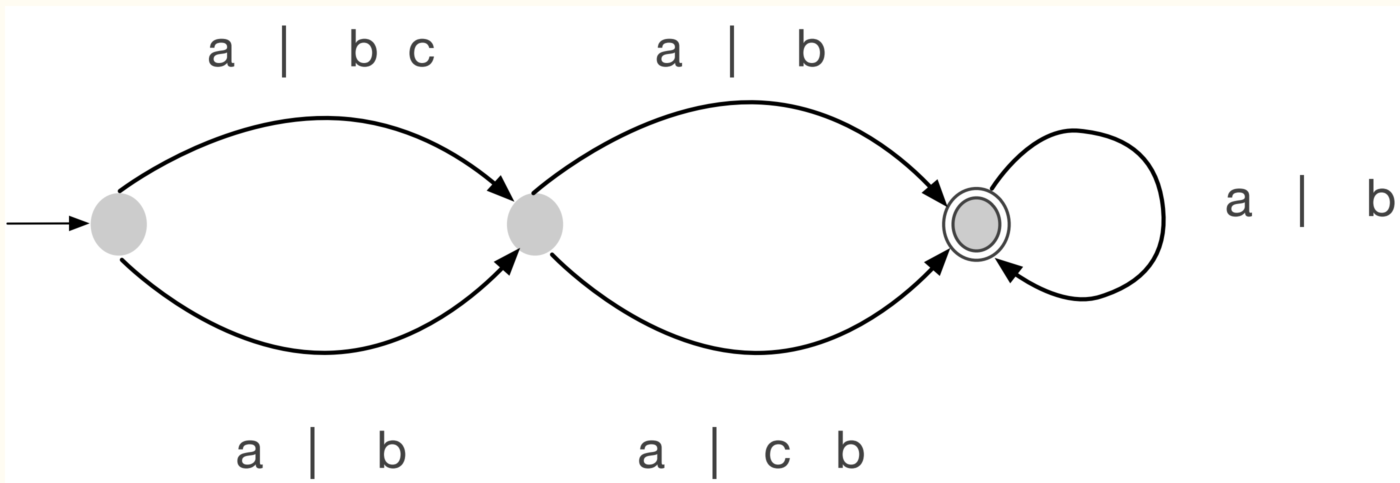
Proof based on the Ehrenfeucht's conjecture:

Every infinite system of word equations has a finite, equivalent subsystem

word equation $x\,y = z\,t$ solution $x = b\,c$ $y = z = b$ $t = c\,b$

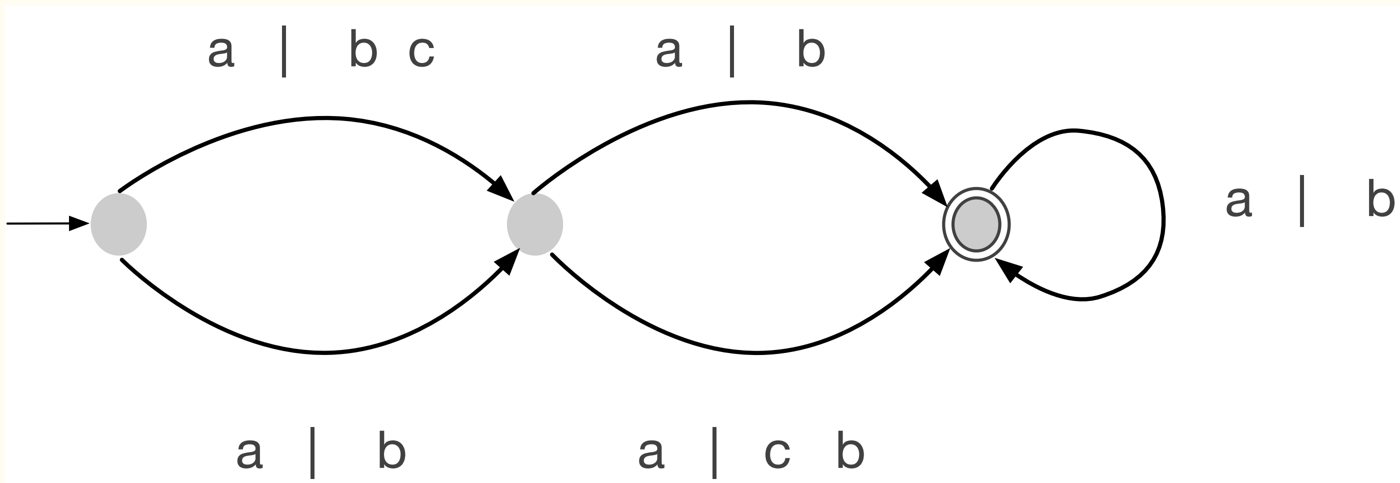
[shown in 1986 by Albert & Lawrence, and Guba]

Equivalence of k-valued one-way transducers is decidable.



k = 3

Equivalence of k-valued one-way transducers is decidable.



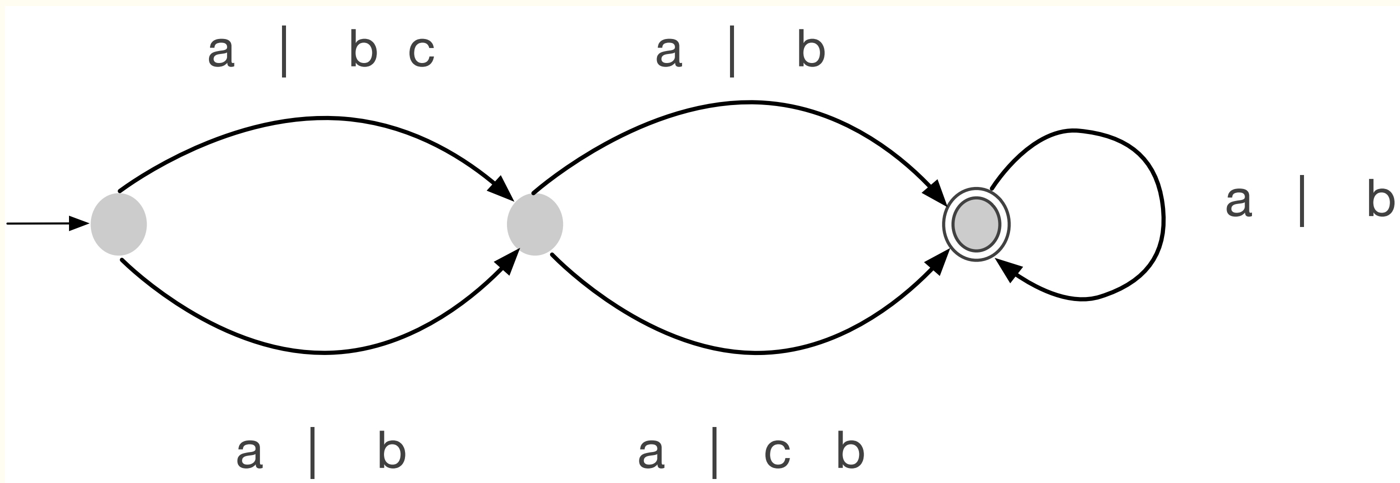
k = 3

[Culik-Karhumäki'86]

1. show that there exists some m such that for any k-valued transducers with at most n states, their equivalence needs to be tested only on words up to length m

2. show that m can be computed effectively

Equivalence of k-valued one-way transducers is decidable.



k = 3

[Culik-Karhumäki'86]

1. show that there exists some m such that for any k-valued transducers with at most n states, their equivalence needs to be tested only on words up to length m

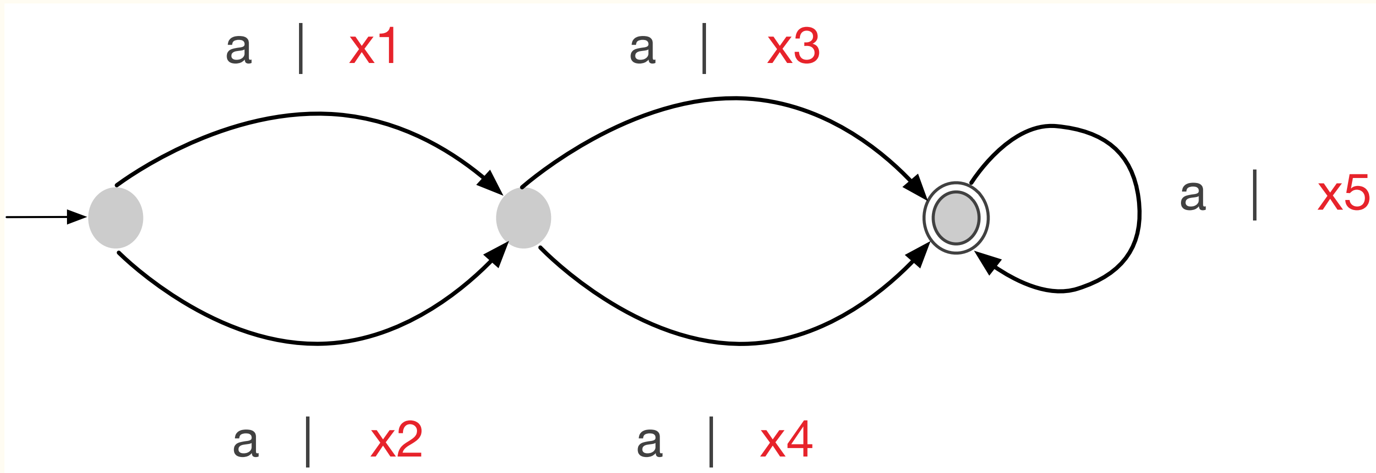
2. show that m can be computed effectively

step 1: Ehrenfeucht's conjecture

step 2: Makanin's algorithm for word equations

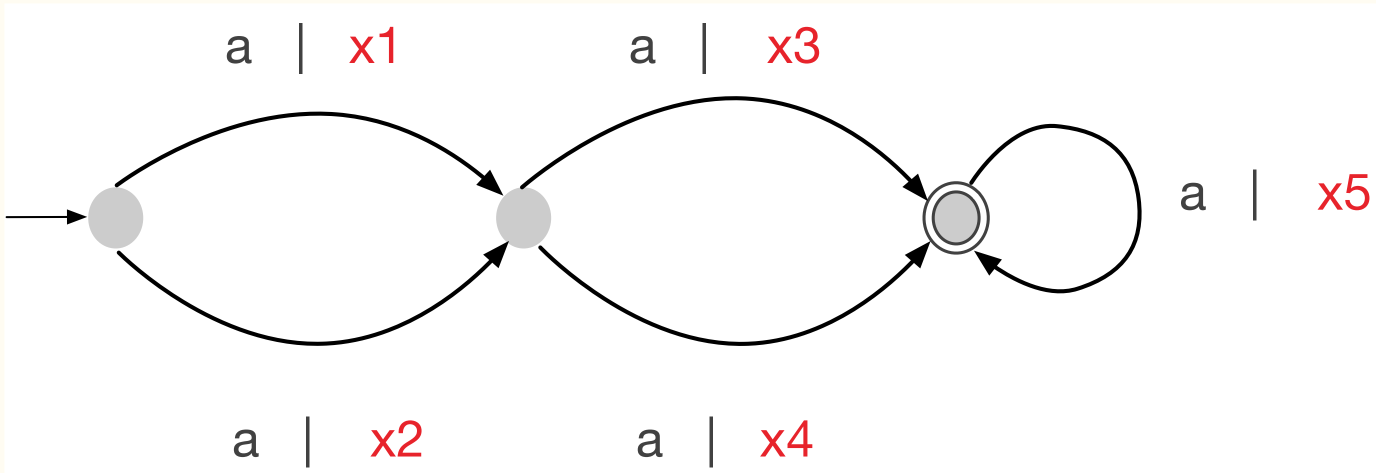
Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]



Equivalence of k-valued one-way transducers is decidable.

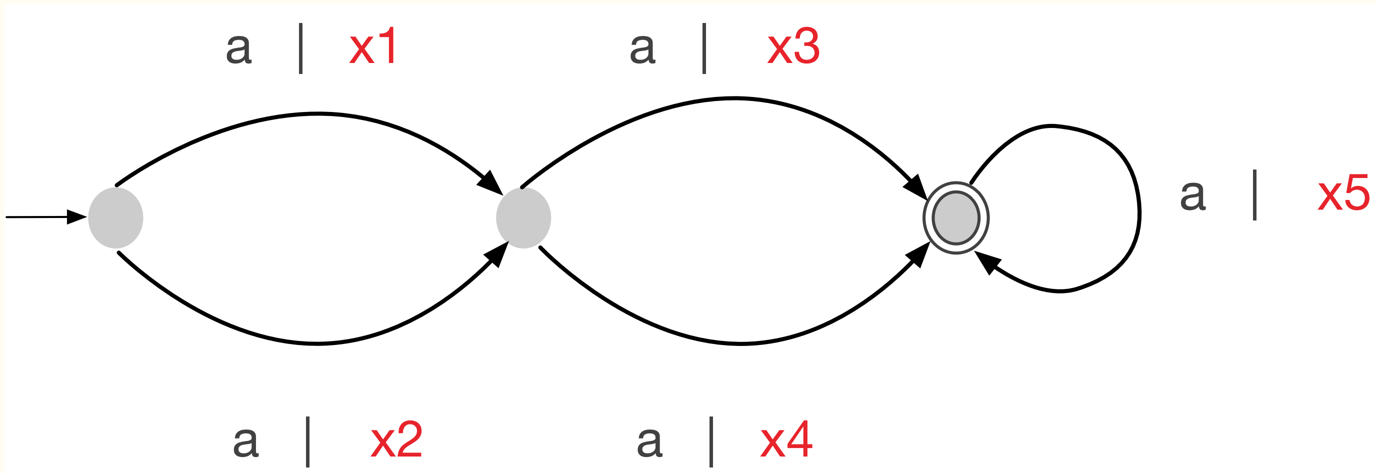
[Culik-Karhumäki'86]



k-valued, one-way implies
bounded outdegree

Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]



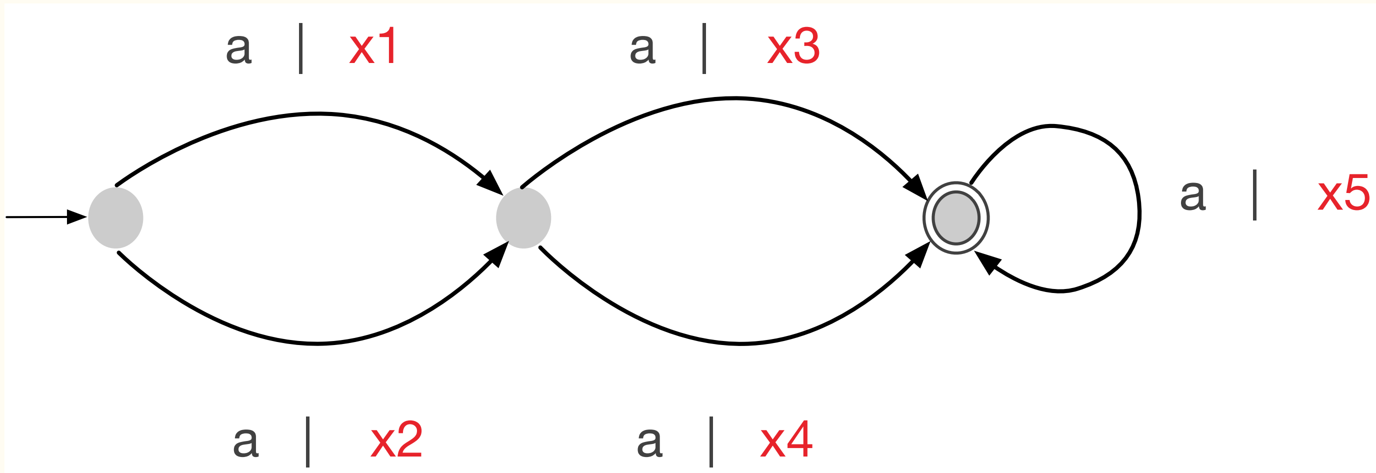
k-valued, one-way implies
bounded outdegree

Given two transducers, replace output words by variables

$$x1\ x3\ \underbrace{x5 \ldots\ x5}_{m} = x2\ x4\ \underbrace{x5\ \ldots\ x5}_{m} \qquad \text{on input} \quad a^{m+2}$$

Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]



k-valued, one-way implies bounded outdegree

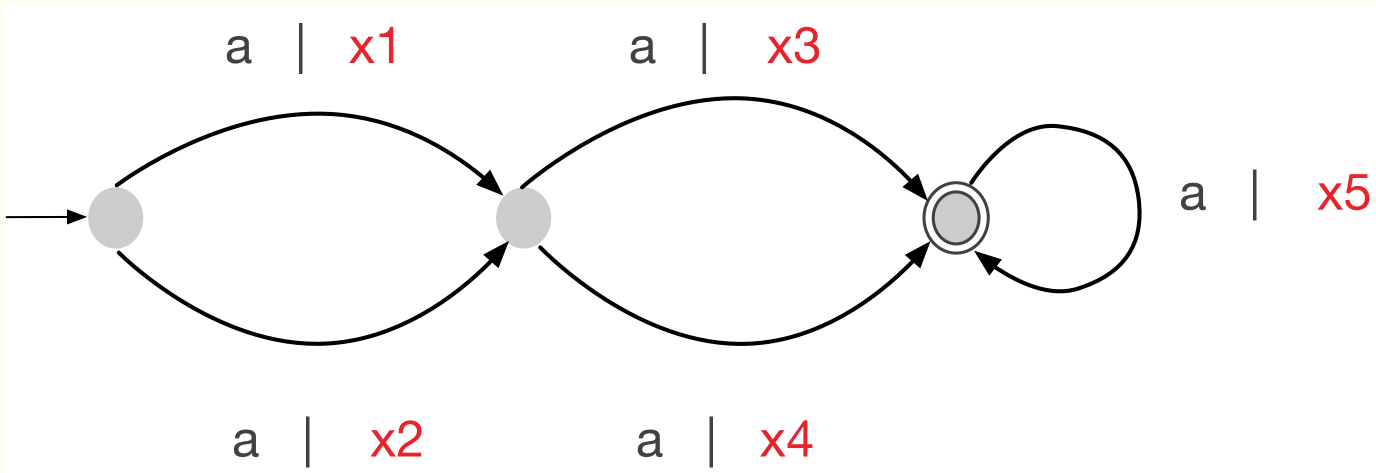
Given two transducers, replace output words by variables

$$\text{x1 x3} \underbrace{\text{x5} \ldots \text{x5}}_{m} = \text{x2 x4} \underbrace{\text{x5} \ldots \text{x5}}_{m} \quad \text{on input} \quad \mathsf{a}^{m+2}$$

- For every input word: group outputs of each transducer in at most k groups

Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]



k-valued, one-way implies bounded outdegree

Given two transducers, replace output words by variables

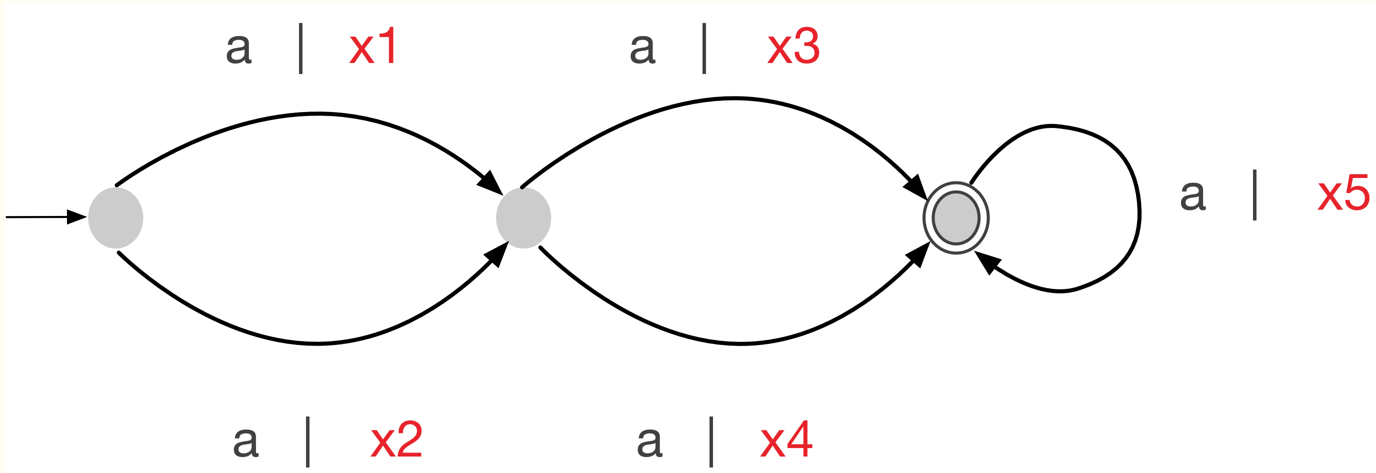
$$x1\ x3\ \underbrace{x5 \ldots\ x5}_{m} = x2\ x4\ \underbrace{x5\ \ldots\ x5}_{m} \qquad \text{on input} \quad a^{m+2}$$

- For every input word: group outputs of each transducer in at most k groups
- System of equations expresses equalities between groups of the two transducers

$$\bigwedge_{w \in \Sigma^*} \bigvee_{\text{groups}} S$$

Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]



k-valued, one-way implies bounded outdegree

Given two transducers, replace output words by variables

$$x1\ x3\ \underbrace{x5 \ldots\ x5}_{m} = x2\ x4\ \underbrace{x5\ \ldots\ x5}_{m} \quad \text{on input} \quad a^{m+2}$$

- For every input word: group outputs of each transducer in at most k groups
- System of equations expresses equalities between groups of the two transducers

$$\bigwedge_{w \in \Sigma^*} \bigvee_{\text{groups}} S \quad \text{equivalent to (Ehrenfeucht)} \quad \bigwedge_{w \in \Sigma^{\leq m}} \bigvee_{\text{groups}} S$$

Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]

$\bigwedge_{w \in \Sigma^*} \bigvee_{\text{groups}} S$ equivalent to $\bigwedge_{w \in \Sigma^{\leq m}} \bigvee_{\text{groups}} S$

Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]

$$\bigwedge_{w \in \Sigma^*} \bigvee_{\text{groups}} S \quad \text{equivalent to} \quad \bigwedge_{w \in \Sigma^{\leq m}} \bigvee_{\text{groups}} S$$

Find *m* effectively: check for

$$\bigwedge_{w \in \Sigma^{\leq m}} \bigvee_{\text{groups}} S = \bigwedge_{w \in \Sigma^{\leq m+1}} \bigvee_{\text{groups}} S$$

Equivalence of k-valued one-way transducers is decidable.

[Culik-Karhumäki'86]

$$\bigwedge_{w \in \Sigma^*} \bigvee_{\text{groups}} S \qquad \text{equivalent to} \qquad \bigwedge_{w \in \Sigma^{\leq m}} \bigvee_{\text{groups}} S$$

Find *m* effectively: check for

$$\bigwedge_{w \in \Sigma^{\leq m}} \bigvee_{\text{groups}} S = \bigwedge_{w \in \Sigma^{\leq m+1}} \bigvee_{\text{groups}} S$$

"Left quotient" of transducer *T* $\qquad T_a(w) := T(aw) \qquad a \in \Sigma$

Equivalence of k-valued NSST is decidable.

[M., Puppis 2019]

Equivalence of k-valued NSST is decidable.

[M., Puppis 2019]

First issue: bounded outdegree no longer obvious

Equivalence of k-valued NSST is decidable.

[M., Puppis 2019]

First issue: bounded outdegree no longer obvious

Reason: SST produce their output piecewise, comparison is harder

Equivalence of k-valued NSST is decidable.

Difficulty: decide if runs of streaming transducers are “close”

Equivalence of k-valued NSST is decidable.

Difficulty: decide if runs of streaming transducers are "close"

Normalization: invariant about periods of register and gap contents, up to a suitable bound

Example (left): c b c b c b c b c b c = final output

x

Equivalence of k-valued NSST is decidable.

Normalization: keep invariants about periods of register/gap contents

Let T be k-valued. We may assume that the number of different updates in transitions is bounded by a constant depending only on k and the number of registers/states of T.

Equivalence of k-valued NSST is decidable.

Consequence: for fixed alphabets/number of registers/states, the set X of k-valued SST is finite (modulo pruning unnecessary transitions).

Equivalence of k-valued NSST is decidable.

Consequence: for fixed alphabets/number of registers/states, the set X of k-valued SST is finite (modulo pruning unnecessary transitions).

Next step: show that for any T in X and input word u, the "u-quotient" also belongs to X:

$$T_u(w) := T(uw)$$

Naive construction preserves the number of states, but not the update size

Equivalence of k-valued NSST is decidable.

Ehrenfeucht: there is some N such that the set of words of length at most N is a test set for all SST in X.

$$T_1 \equiv_N T_2$$

equivalence over words of length at most N

Equivalence of k-valued NSST is decidable.

Ehrenfeucht: there is some N such that the set of words of length at most N is a test set for all SST in X.

$T_1 \equiv_N T_2$ equivalence over words of length at most N

How do we compute N? inductively

Equivalence of k-valued NSST is decidable.

Ehrenfeucht: there is some N such that the set of words of length at most N is a test set for all SST in X.

$T_1 \equiv_N T_2$ equivalence over words of length at most N

How do we compute N? inductively

Assume that we found N such that $T_1 \equiv_N T_2$ iff $T_1 \equiv_{N+1} T_2$

for all transducers from X

How? E.g. using an algorithm for solving word equations (Makanin)

Equivalence of k-valued NSST is decidable.

$T_a(w) := T(aw)$

$T \in X$ implies $T_a \in X$

$T_1 \equiv_N T_2$ iff $T_1 \equiv_{N+1} T_2$

$$T_1 \equiv_r T_2 \iff T_1 \equiv_N T_2 \quad \text{for all } r \geq N \text{ and } T_1, T_2 \in X$$

Equivalence of k-valued NSST is decidable.

$T_a(w) := T(aw)$

$$T \in X \ \text{ implies } \ T_a \in X$$

$$T_1 \equiv_N T_2 \quad \text{iff} \quad T_1 \equiv_{N+1} T_2$$

$$T_1 \equiv_r T_2 \quad \Longleftrightarrow \quad T_1 \equiv_N T_2 \quad \text{for all } r \geq N \text{ and } T_1, T_2 \in X$$

$$T_1 \equiv_{r+1} T_2 \quad \text{iff} \quad T_{1,a} \equiv_r T_{2,a} \quad \text{for all } a$$

Equivalence of k-valued NSST is decidable.

$T_a(w) := T(aw)$

$T \in X \;\text{ implies }\; T_a \in X$

$T_1 \equiv_N T_2$ iff $T_1 \equiv_{N+1} T_2$

$$T_1 \equiv_r T_2 \iff T_1 \equiv_N T_2 \quad \text{for all } r \geq N \text{ and } T_1, T_2 \in X$$

$T_1 \equiv_{r+1} T_2$ iff $T_{1,a} \equiv_r T_{2,a}$ for all a

iff $T_{1,a} \equiv_{r-1} T_{2,a}$ for all a

Equivalence of k-valued NSST is decidable.

$T_a(w) := T(aw)$

$T \in X$ implies $T_a \in X$

$T_1 \equiv_N T_2$ iff $T_1 \equiv_{N+1} T_2$

$$T_1 \equiv_r T_2 \iff T_1 \equiv_N T_2 \quad \text{for all } r \geq N \text{ and } T_1, T_2 \in X$$

$T_1 \equiv_{r+1} T_2$ iff $T_{1,a} \equiv_r T_{2,a}$ for all a

iff $T_{1,a} \equiv_{r-1} T_{2,a}$ for all a

iff $T_1 \equiv_r T_2$

Overview

Word transductions

- automata = logic
- translations between models
- expressions
- algebra?

Equivalence problem

- finitely valued transducers
- Ehrenfeucht & Hilbert
- origin equivalence

Equivalence of copyful DSST is decidable.

[Filiot-Reynier'17] [Benedikt et al.'17]

copyful: registers can occur multiple times in updates

Equivalence of copyful DSST is decidable.

[Filiot-Reynier’17] [Benedikt et al.’17]

copyful: registers can occur multiple times in updates

- A. A.Markov (~ 1948): encoding words by integers

Equivalence of copyful DSST is decidable.

[Filiot-Reynier'17] [Benedikt et al.'17]

copyful: registers can occur multiple times in updates

- A. A.Markov (~ 1948): encoding words by integers

Every 2x2 matrix with non-negative integer entries and determinant 1 can be encoded in a unique way as product of matrices:

$$M_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad M_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Word concatenation turns into matrix multiplication

Equivalence of copyful DSST is decidable.

copyful: registers can occur multiple times in updates

[Filiot-Reynier'17] [Benedikt et al.'17]

- A. A.Markov (~ 1948): encoding words by integers

Every 2x2 matrix with non-negative integer entries and determinant 1 can be encoded in a unique way as product of matrices:

$$M_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad M_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Word concatenation turns into matrix multiplication

- Encode binary string by (value, 2^length), e.g. 011 encoded by (3,8)

Equivalence of copyful DSST is decidable.

[Filiot-Reynier'17] [Benedikt et al.'17]

copyful: registers can occur multiple times in updates

(Copyful) DSST turn into word-to-integer transducers with registers and polynomial operations:

polynomial automata [Benedikt et al.'17]

Equivalence of copyful DSST is decidable.

[Filiot-Reynier'17] [Benedikt et al.'17]

copyful: registers can occur multiple times in updates

(Copyful) DSST turn into word-to-integer transducers with registers and polynomial operations:

polynomial automata [Benedikt et al.'17]

Equivalence of copyful DSST reduces to zeroness problem for polynomial automata:

Is the language of a polynomial automaton included in {0}?

Equivalence of copyful DSST is decidable.

[Filiot-Reynier'17] [Benedikt et al.'17]

copyful: registers can occur multiple times in updates

(Copyful) DSST turn into word-to-integer transducers with registers and polynomial operations:

polynomial automata

[Benedikt et al.'17]

Equivalence of copyful DSST reduces to zeroness problem for polynomial automata:

Is the language of a polynomial automaton included in {0}?

Zeroness of polynomial automata is decidable.

[Benedikt et al.'17]

Equivalence of copyful DSST is decidable.

copyful: registers can occur multiple times in updates

[Filiot-Reynier'17] [Benedikt et al.'17]

Zeroness of polynomial automata is decidable.

[Benedikt et al.'17]

Equivalence of copyful DSST is decidable.

copyful: registers can occur multiple times in updates

[Filiot-Reynier'17] [Benedikt et al.'17]

Zeroness of polynomial automata is decidable.

[Benedikt et al.'17]

Two semi-algorithms: the first one searches input with non-zero output.

The other semi-algorithm searches a proof for the polynomial automaton being constant zero using Hilbert's Basis Theorem.

[Bojanczyk, SIGLOG'19]

Overview

Word transductions

- automata = logic
- translations between models
- expressions
- algebra?

Equivalence problem

- finitely valued transducers
- Ehrenfeucht & Hilbert
- origin equivalence

Origin equivalence

"Tag" each output symbol with the input position where it was generated: output alphabet is $\Gamma \times \mathbb{N}$

[Bojańczyk 2014]

Origin information brings word transducers closer to automata:

- Regular word functions with origin information enjoy a Myhill-Nerode congruence: machine-independent characterisation
- First-order definable regular word functions have an effective characterisation
- Less combinatorics, more decidability

Origin equivalence

"Tag" each output symbol with the input position where it was generated

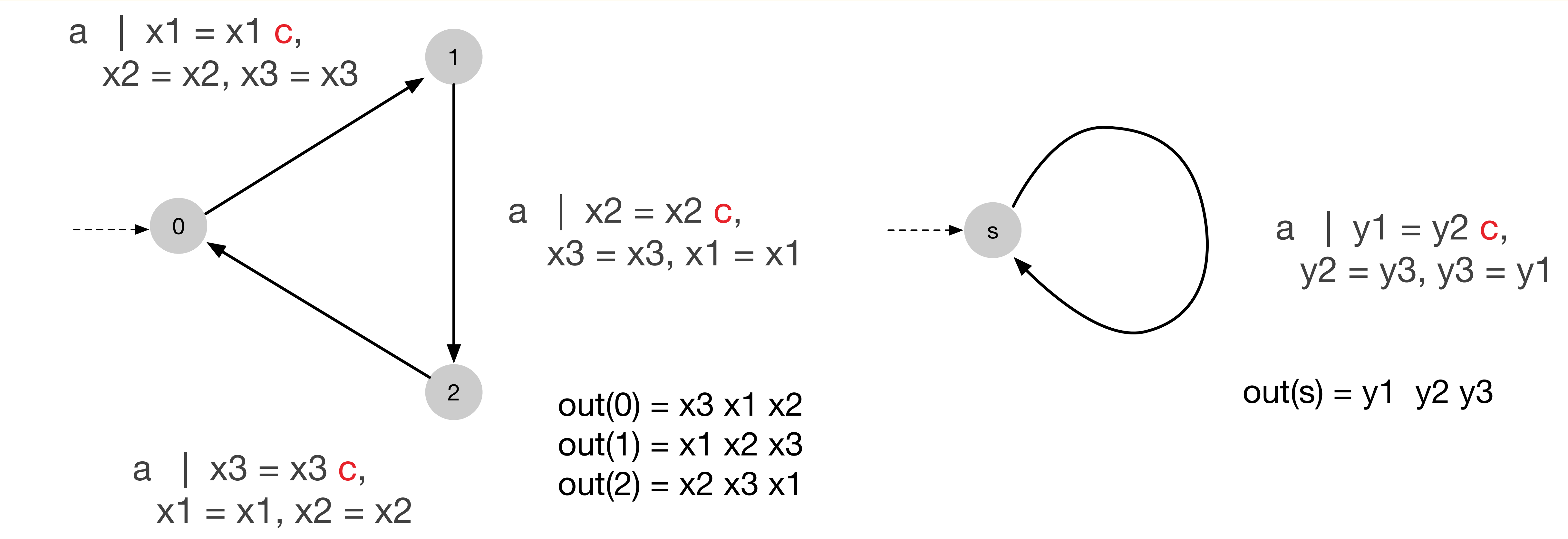
Transducers T, T' are origin-equivalent if they are equivalent in the origin semantics.

Origin-equivalence of non-deterministic two-way transducers (2NFT) is decidable: PSPACE-complete.

[Bose, M., Penelle, Puppis'18]

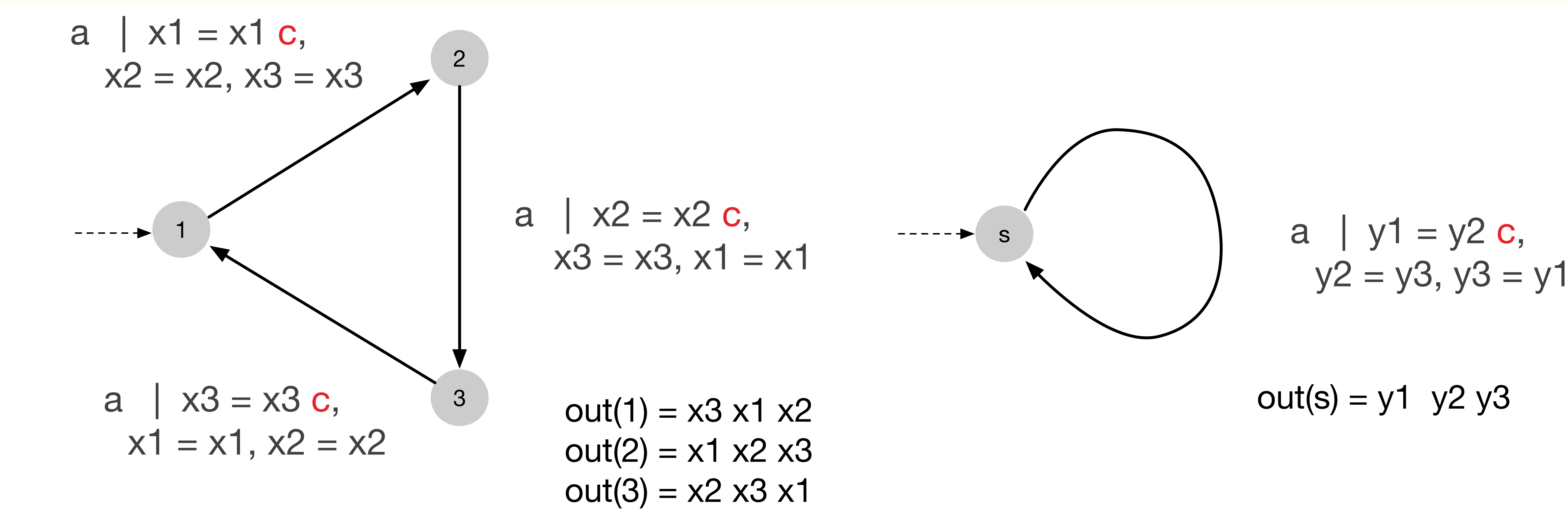
Idea: origin-equivalence for 2NFT reduces to "runs of same shape".

Origin-equivalence of deterministic SST is decidable in PSPACE.



Idea: origin-equivalence for DSST through backward propagation of constraints (= simple word equations)

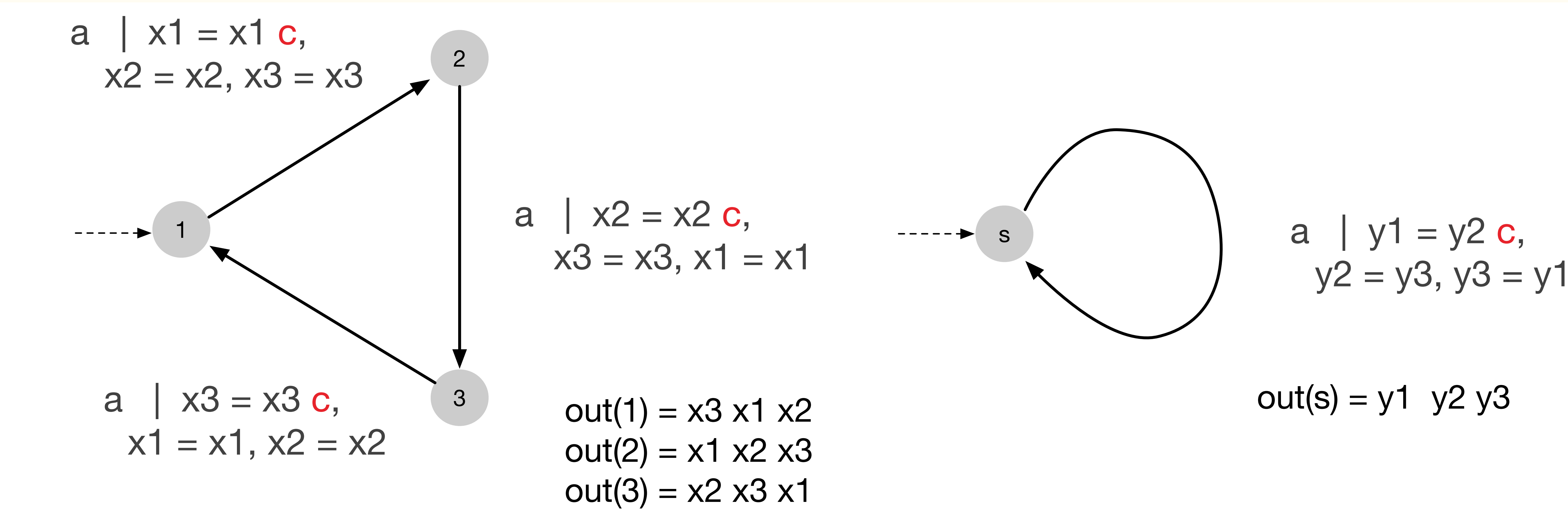
Origin-equivalence of deterministic SST is decidable in PSPACE.



out(1) = out(s)   x3 x1 x2 = y1 y2 y3

(1,s) <— (3,s)	x3 c x1 x2 = y2 c y3 y1	x3 = y2, x1 x2 = y3 y1
(3,s) <— (2,s)	x3 = y3, x1 x2 c = y1 y2 c	x1 x2 = y1 y2
(2,s) <— (1,s)	x3 = y1, x1 c x2 = y2 c y3	x1 = y2, x2 = y3

Origin-equivalence of deterministic SST is decidable in PSPACE.



out(1) = out(s)   x3 x1 x2 = y1 y2 y3

Invariants: at state 1   x1 = y2, x2 = y3, x3 = y1

at state 2   x1 = y1, x2 = y2, x3 = y3

at state 3   x1 = y3, x2 = y1, x3 = y2

Origin-equivalence of deterministic copyful SST is decidable.

Copyful: registers can occur several times in right-hand sides of updates.

Algorithm:

- build product of SST T_1, T_2
- compute backwards constraints of the form

$$\alpha = \beta, \quad \alpha \in R_1^*, \ \ \beta \in R_2^*$$

Termination: if no inconsistency detected during propagation, termination is based on Ehrenfeucht's conjecture + Makanin

Origin-equivalence of deterministic copyful SST is decidable.

Copyful: registers can occur several times in right-hand sides of updates.

Alternatively: reduction to (classical) equivalence of copyful DSST [Filiot]

- additional register **m**, additional output symbol #
- update x := a y x b replaced by x : a **m** y x b **m**
- **m** : = **m** # at each transition

Unary output alphabet

- For origin-equivalence the output alphabet can be assumed to be unary:

Origin-equivalence over arbitrary output alphabet is polynomially reducible to origin-equivalence over unary alphabet.

Unary output alphabet

- For origin-equivalence the output alphabet can be assumed to be unary:

Origin-equivalence over arbitrary output alphabet is polynomially reducible to origin-equivalence over unary alphabet.

- For usual equivalence this is conjectured to be false

Equivalence of deterministic, copyful SST is in Ackermann (Benedikt et al. LICS'17)

Equivalence of deterministic, copyful SST with unary output alphabet is in PTIME (Karr's algorithm, cf. Müller-Olm and Seidl 2004)

Open questions

- Complexity of equivalence of deterministic SST?
- Same for origin-equivalence?
- Decomposition theorem for finitely-valued SST?

Every k-valued one-way transducer can be decomposed into k single-valued one-way transducers.

[Weber'96, Sakarovitch, de Souza'08]

If similar statement holds for NSST:

NSST = 2NFT in finite valued case (conjectured).

Bibliography

E. Filiot, P.-A. Reynier. Transducers, logic and algebra for functions of finite words. ACM SIGLOG News 3(3), 4019, 2016

A. Muscholl, G. Puppis. The many facets of string transducers. LIPICs 126:1-21, STACS 2019

M. Bojanczyk. The Hilbert method for transducer equivalence. ACM SIGLOG News 6(1), 5-17, 2019

Thank you for listening!