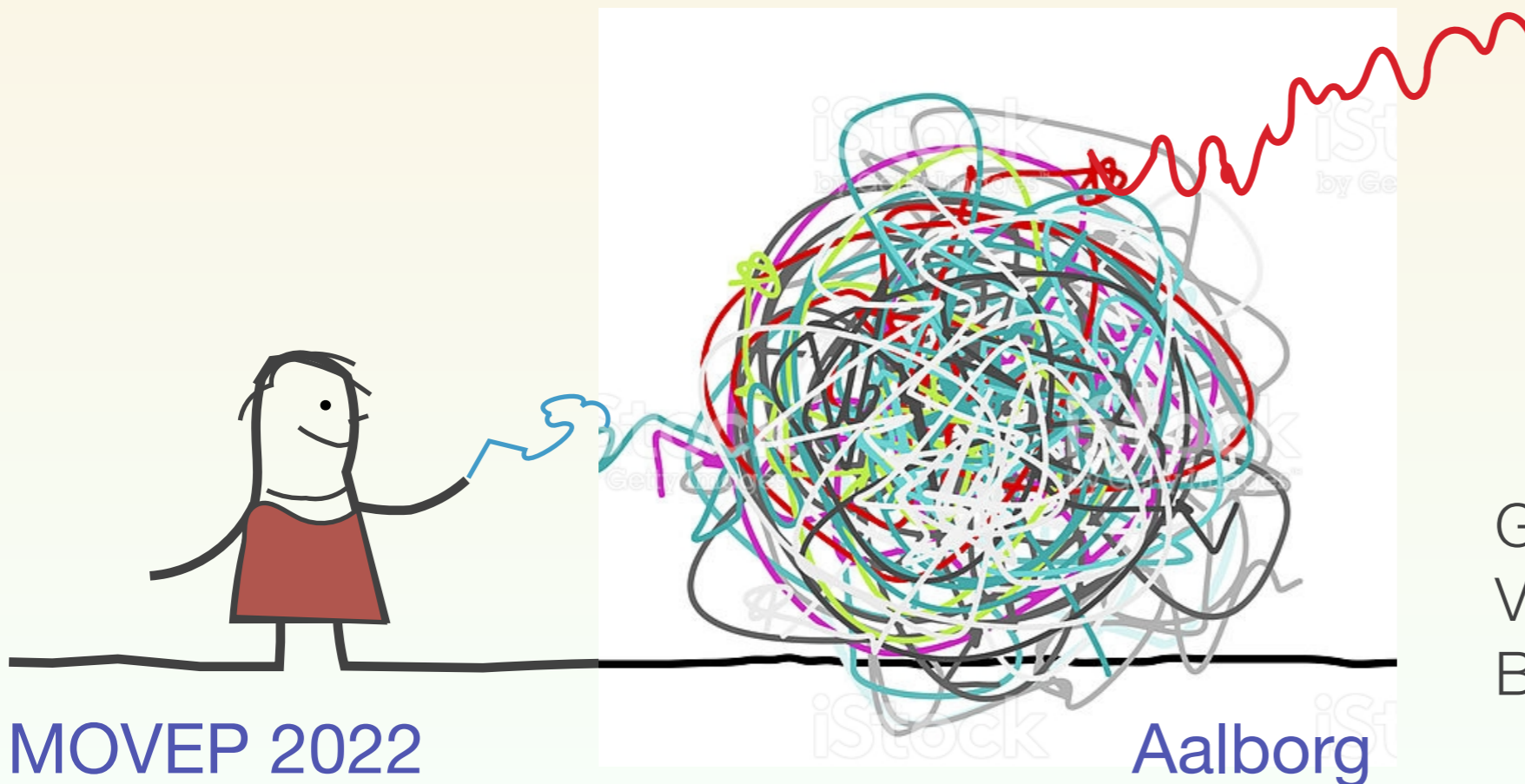


A View on String Transducers

Anca Muscholl



Gabriele Puppis, Olivier Gauwin,
Vincent Penelle, Felix
Baschenis, Sougata Bose

MOVEP 2022

Aalborg

Transductions: history

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

E. F. Moore 1956 “Gedankenexperiments on sequential machines”

D. Scott 1967: “[...] the functions computed by the various machines are more important - or at least more basic - than the sets accepted by these devices“

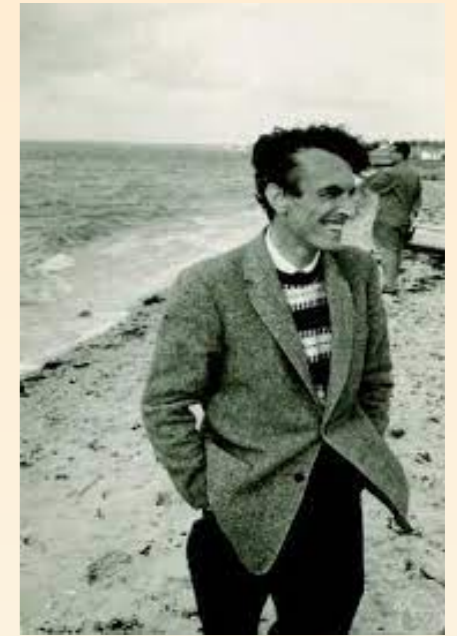
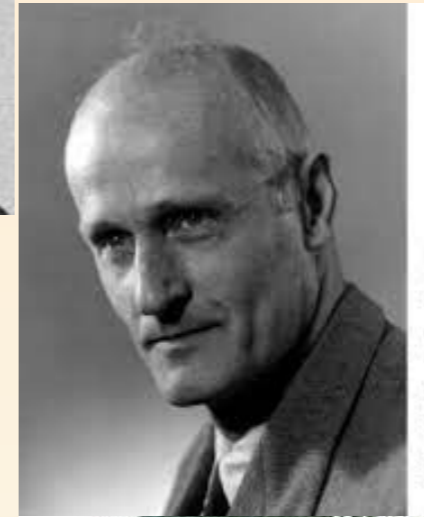
Schützenberger 1961, Elgot-Mezei 1965, Ginsburg-Rose 1966, Nivat 1968, Aho-Hopcroft-Ullman 1969, Engelfriet 1972, Eilenberg 1976, Choffrut 1977,...

Overview

Word transductions



Büchi



Kleene

Schützenberger



Equivalence problem



Culik-Karhumäki

Ehrenfeucht & Hilbert



Bojańczyk

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely-valued transductions

origin equivalence

Transducers

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

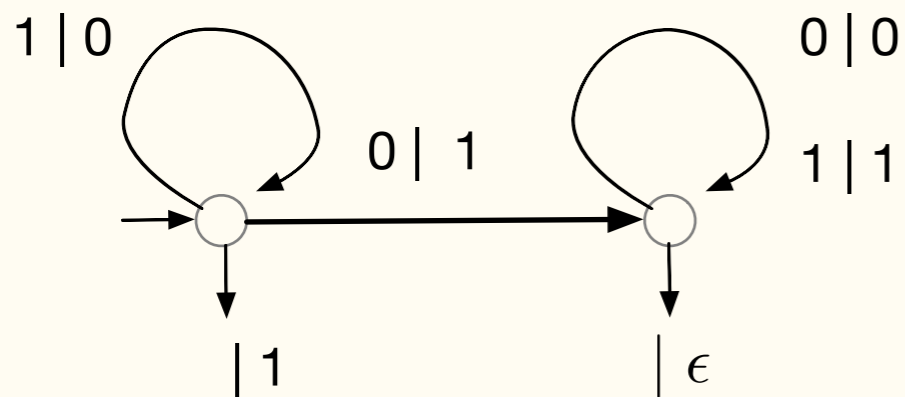
Equivalence problem

finitely-valued transductions

origin equivalence

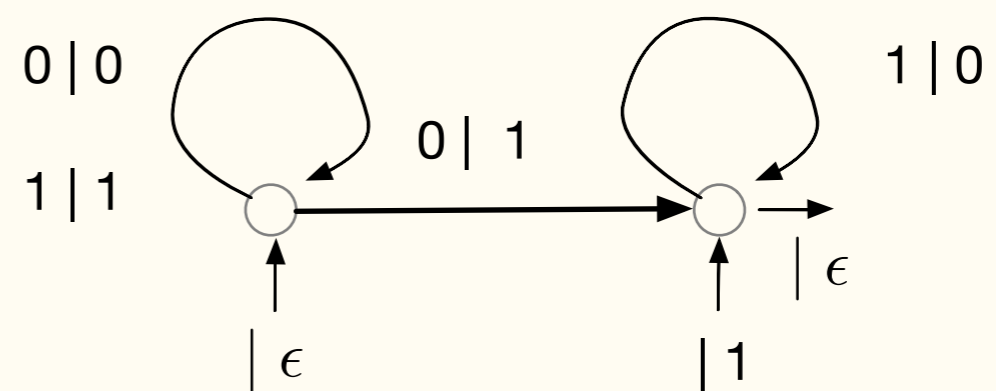
transform objects - here: [words into words](#)

Example: binary increment (least significant bit left/right)



deterministic

lsb left



non-deterministic

lsb right

Transducers

one-way finite-state transducers

metamorphosis \longrightarrow

mtmrphss

erase vowels

1 1 0 0 1 \longrightarrow

0 0 1 0 1

increment

two-way finite-state transducers

metamorphosis \longrightarrow

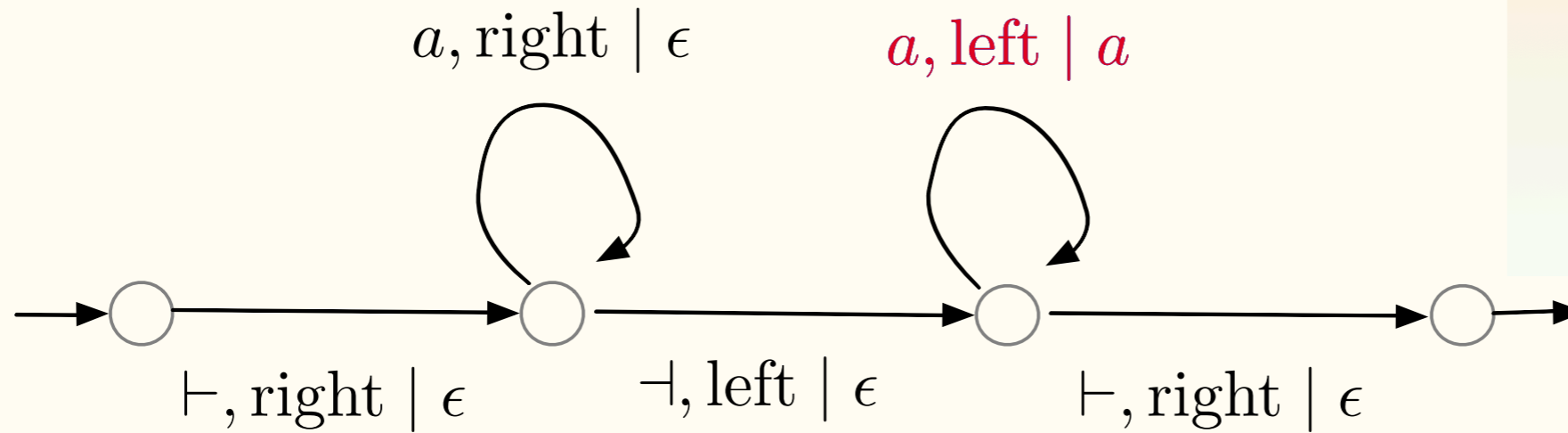
sisohpromatem

output mirror

metamorphosis \longrightarrow mmooss

output letters occurring more than once

Example



Overview

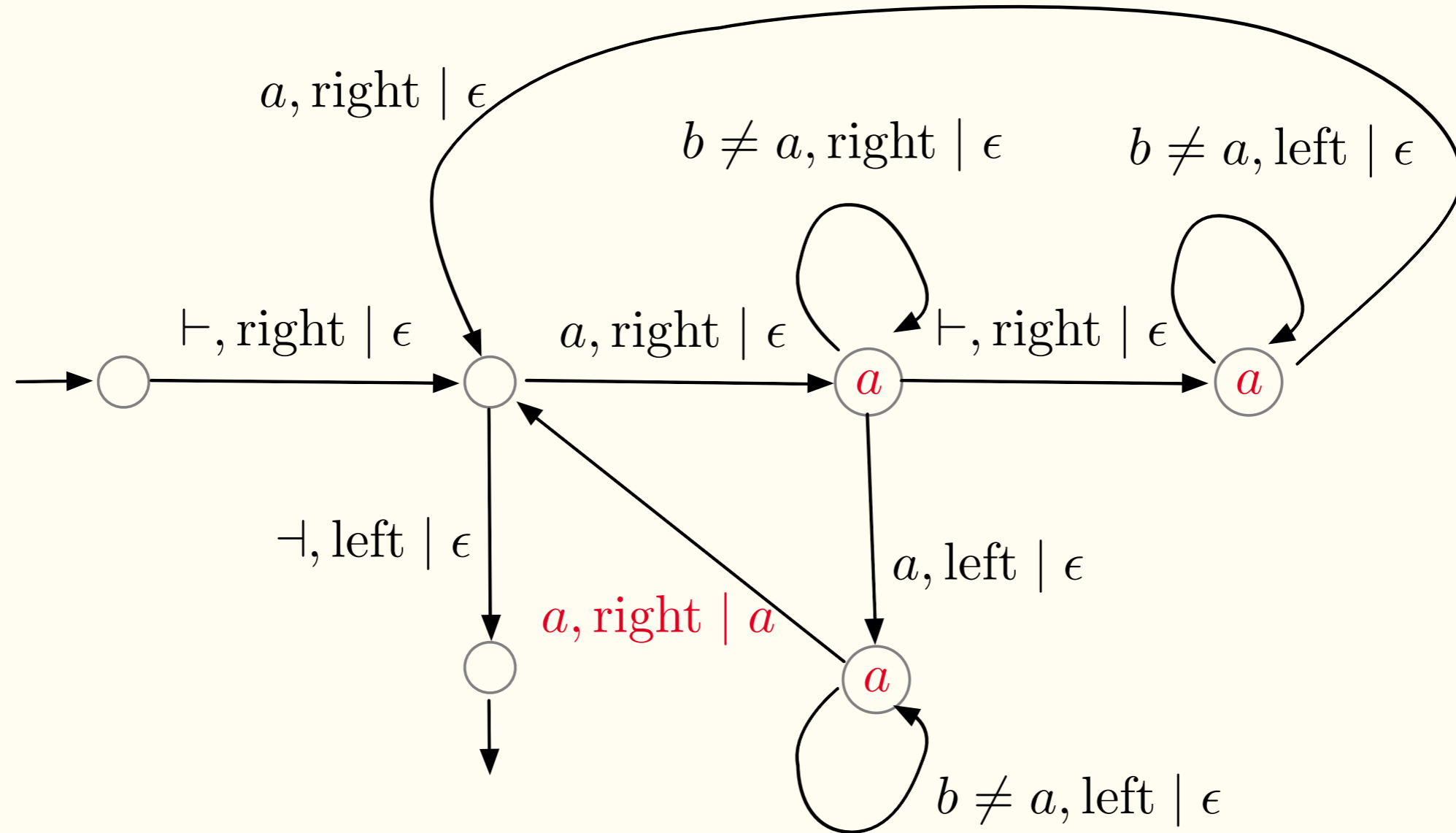
- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely-valued transductions
 - origin equivalence

deterministic 2-way transducer computing the mirror transduction



Example: **deterministic** 2-way transducer computing the transduction:

“subsequence of letters occurring more than once”



metamorphosis \longrightarrow mmooss

Finite state transducers are finite automata with transitions:

one-way

$$s \xrightarrow{a|u} s'$$

read a from input and output word u

two-way

$$s \xrightarrow{a, D|u} s'$$

$D \in \{\text{left, right}\}$

read a from input, move D and output word u

other features:

- ❖ deterministic/non-deterministic
- ❖ regular look-ahead, look-around
- ❖ pebbles

Example: “subsequence of letters occurring more than once”

metamorphosis \longrightarrow mmooss

Can be computed by

- ❖ deterministic one-way transducer with regular look-ahead
- ❖ non-deterministic one-way transducer

but not by any deterministic one-way transducer

Deterministic one-way transducers with regular look-ahead and **single-valued** non-deterministic one-way transducers compute the same word functions.

Logic

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely-valued transductions

origin equivalence

MSOT: monadic second-order transductions

[Courcelle, Engelfriet]

❖ output consists of fixed number of **copies** of input positions

❖ **domain** formula: unary MSO formula $\text{dom}_{a,i}(x)$

“***i***-th copy of input position ***x*** occurs in the output and is labeled by symbol ***a***”

❖ **order** formula: binary MSO formula $\text{ord}_{i,j}(x, y)$

“***i***-th copy of input position ***x*** precedes ***j***-th copy of input position ***y*** in the output”

Logic

MSOT: monadic second-order transductions

Example: $w \mapsto w w$

❖ 2 copies

❖ domain formula: $\text{dom}_{a,1}(x) = \text{dom}_{a,2}(x) \equiv a(x)$

❖ order formula: $\text{ord}_{1,1}(x, y) = \text{ord}_{2,2}(x, y) = (x < y)$

$\text{ord}_{1,2}(x, y) = \text{true}$

MSOT = 2DFT

[Engelfriet-Hoogeboom 2001]

Logic

NMSOT: non-deterministic monadic second-order transductions

maps a structure into a (finite) set of structures

Example: $u v \longrightarrow v u$ relation (not function)

- ❖ one copy
- ❖ color input as $0^* 1^*$ $\exists X_0 \exists X_1 (\text{Partition}(X_0, X_1) \wedge \forall x \in X_0, y \in X_1 : x < y)$
- ❖ order formula: if both positions in same set, same order; else positions from X_1 before positions of X_0

$$\text{ord}_{1,1}(x, y) = (x \in X_1 \wedge y \in X_0) \vee \bigvee_i (x \in X_i \wedge y \in X_i \wedge x < y)$$

Transducers with registers

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

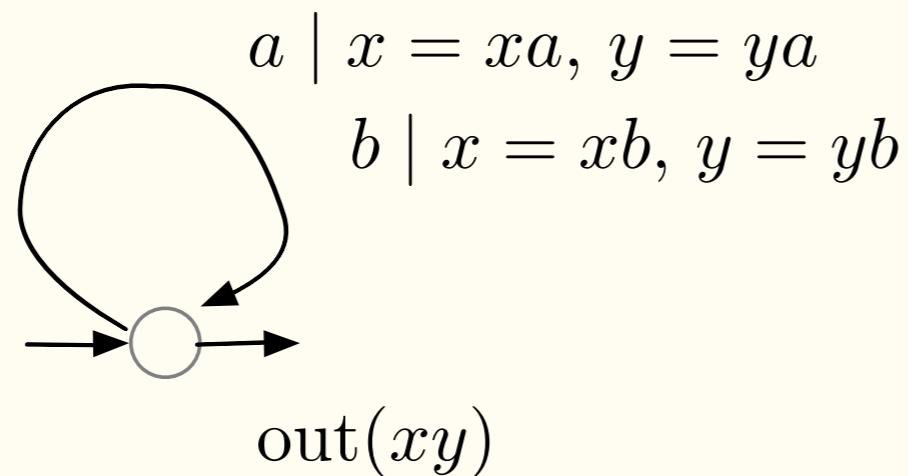
finitely-valued transductions

origin equivalence

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

Copyless = no register occurs twice in RHS

SST means **copyless**

Copyful SST = HDTOL

Word functions

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely-valued transductions

origin equivalence

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

single-valued: **functions** $f : A^* \longrightarrow B^*$

2DFT = DSST = NSST = MSOT

regular word functions

[Engelfriet-Hoogeboom 2001]

[Alur-Cerny 2010]

Landscape of transducers

1DFT

$$a W \mapsto W a$$

subsequential functions

2DFT = DSST = MSOT

$$W \mapsto WW$$

regular functions

1NFT

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

rational functions

$$UV \mapsto VU$$

decidable equivalence
undecidable equivalence

2NFT

$$W \mapsto W^*$$

NSST = NMSOT

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely-valued transductions

origin equivalence

Properties

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem

- ❖ [Single-valuedness](#) can be decided in NLOG for one-way transducers, and in PSPACE for streaming and two-way transducers.
- ❖ Single-valued two-way and streaming transducers can be [determinised](#).
- ❖ Non-deterministic one-way transducers, and deterministic two-way/streaming transducers are closed under [composition](#).

Properties

Single-valuedness can be decided in NLOG for one-way transducers, and in PSPACE for streaming and two-way transducers.

- ❖ **1NFT**: Guess on-the fly 2 runs on same input, together with the output position where the two runs differ (resp. length difference). The output position is identified by a **counter** that maintains the difference of output lengths. Emptiness of counter automata is in NLOG.
- ❖ **2NFT**: Same, but counter automaton is exponential, so complexity is PSPACE. Lower bound from intersection of finite automata.
- ❖ **NSST**: More or less the same (PSPACE upper bound, exponential in the number of registers). **No lower bound**.

Properties

Single-valued two-way and streaming transducers can be **determinised**.

- ❖ **NSST**: use subset construction and maintain a copy for each register/state pair. This leads to **bounded-copy** DSST, which are equivalent to DSST.

[Alur-Filiot-Trivedi 2012]

- ❖ **2NFT**: with regular look-around we can follow the minimal accepting run and do the outputs accordingly. Regular look-around can be implemented by using **reversible** automata. See next slides.

A 2DFA is **reversible** if it is co-deterministic: the current state + previous letter determine the previous state.

Properties

Non-deterministic one-way transducers, and deterministic two-way/streaming transducers are closed under **composition**.

- ❖ 1NFT: direct product
- ❖ For 2DFT/DSST one can use closure under composition of MSO interpretations.
- ❖ Direct proof for 2DFT uses **reversible** 2DFT (see next slides).

A 2DFT is **reversible** if its automaton is reversible.

From DSST to 2DFT

Overview

Word transductions

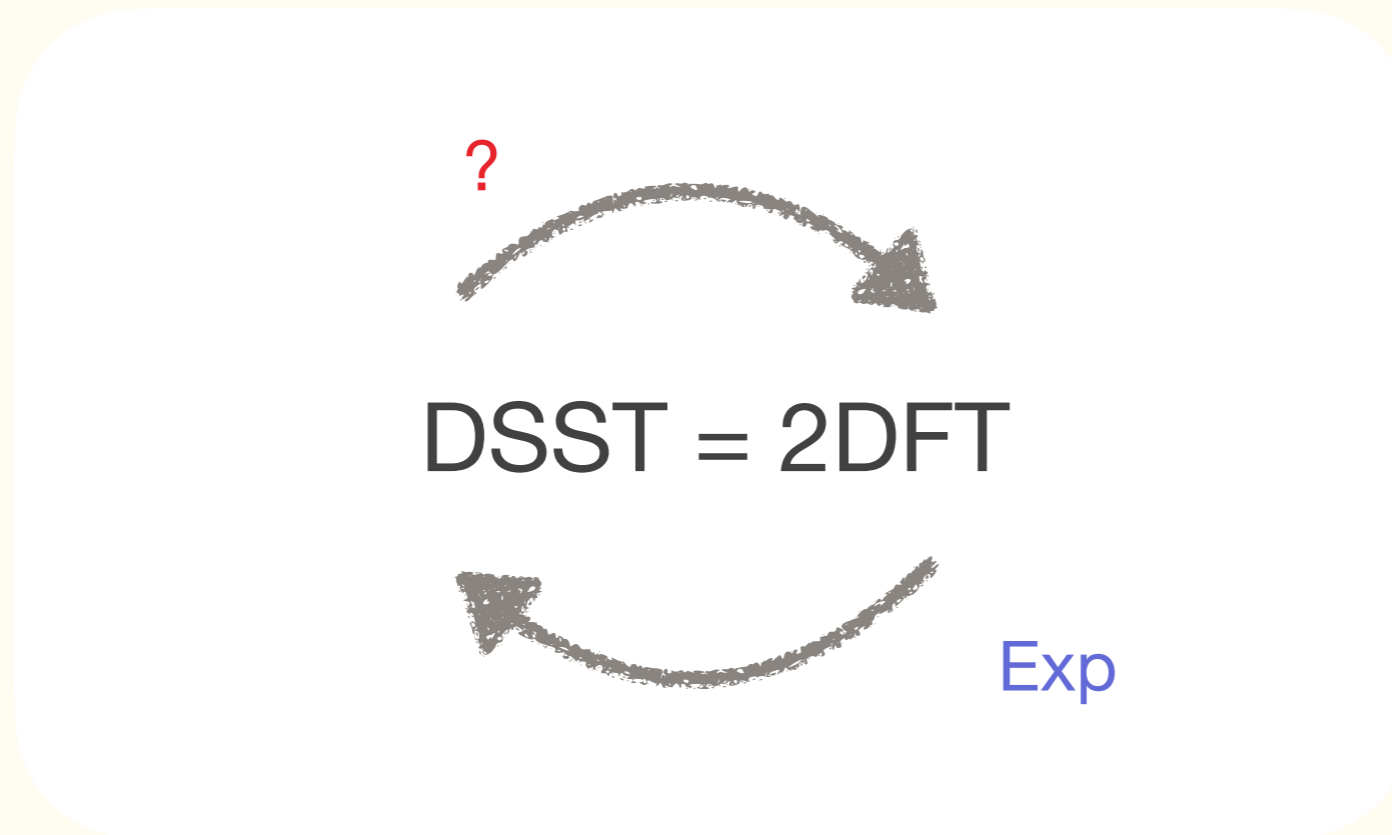
automata = logic

properties

expressions

class membership problems

Equivalence problem

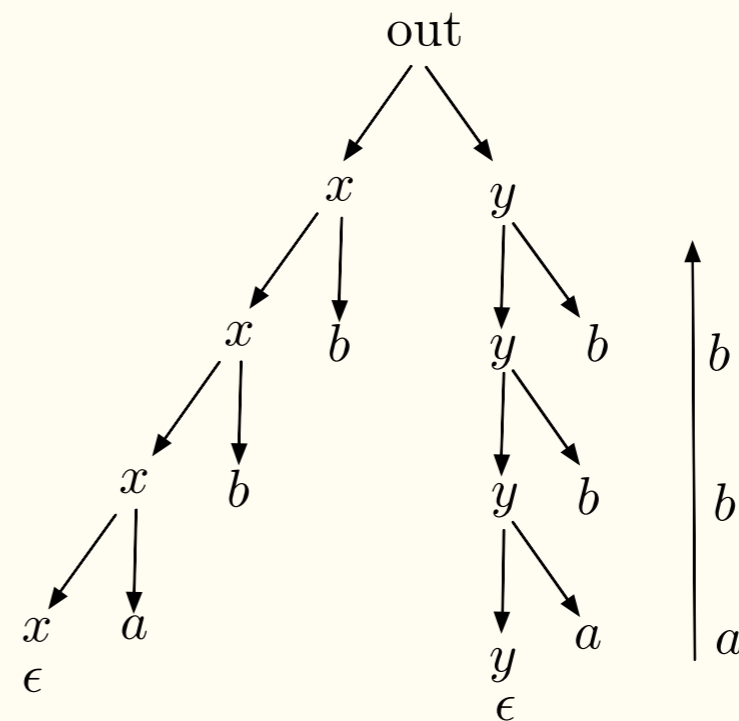
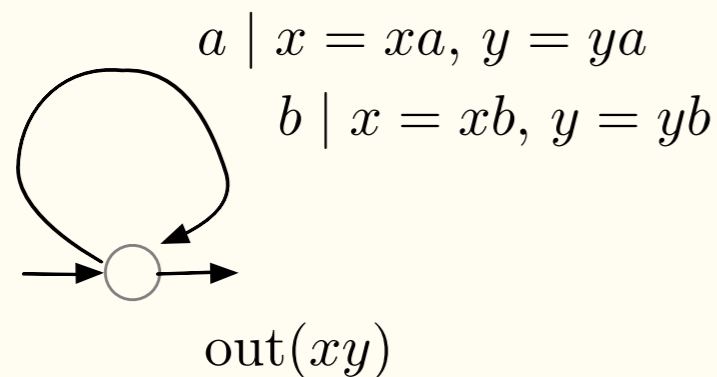


- ❖ a one-way transducer **T** annotates the input by the accepting run of the DSST
- ❖ a two-way transducer **OUT** builds the output from the **annotated** input
- ❖ if we can replace **T** by **reversible T'**: just compose **T'** with **OUT**

From DSST to 2DFT


 DSST = 2DFT

- ❖ a one-way transducer **T** annotates the input by the accepting run of the DSST
- ❖ 2DFT **OUT** can build the output from the annotated input (DFS on tree of updates)



- ❖ if we can replace **T** by reversible **T'**: just compose **T'** with **OUT**

Word transductions

automata = logic

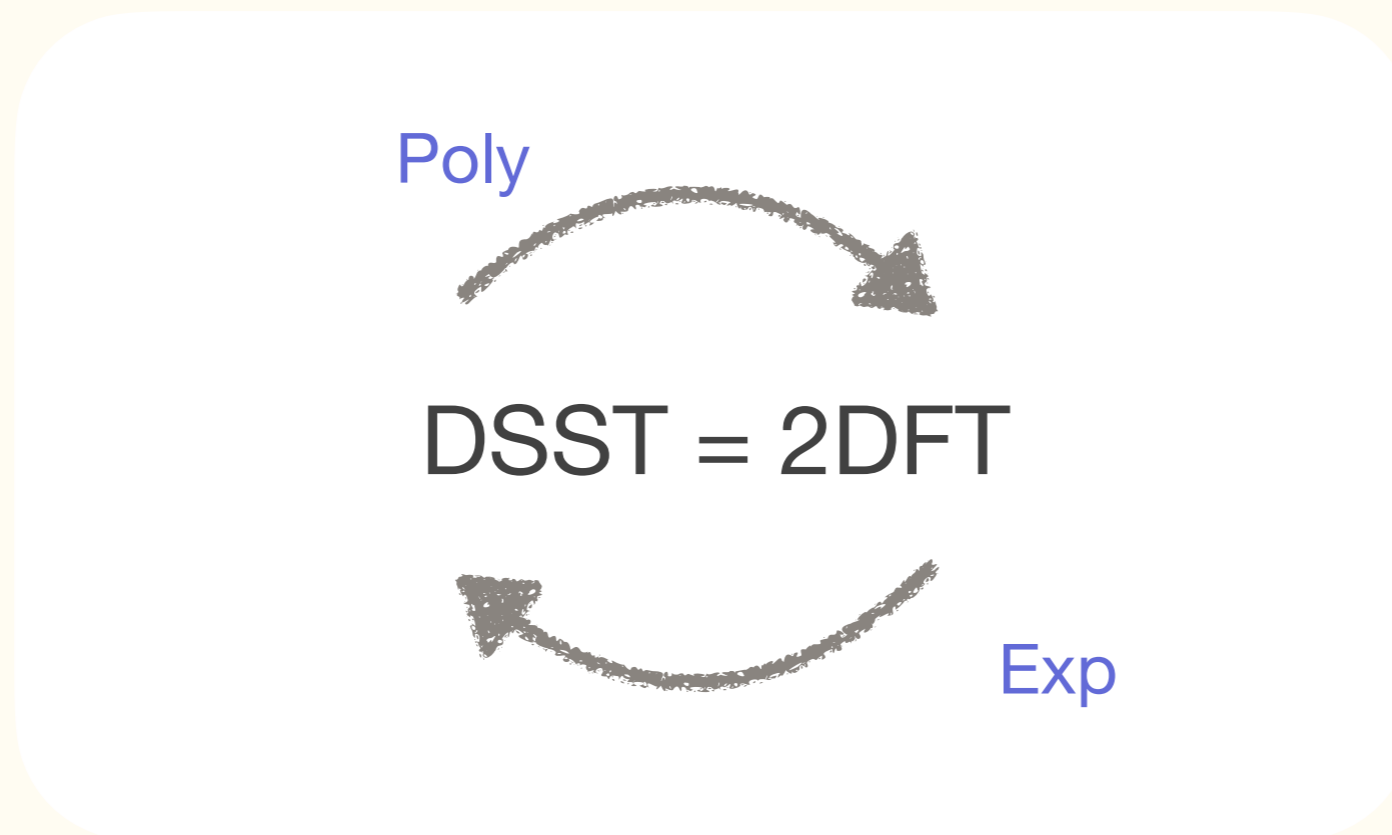
properties

expressions

class membership problems

Equivalence problem

From DSST to 2DFT



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

[Dartois, Fournier, Jecker, Lhote 2017]

Cor: DSST can be simulated by 2DFT with polynomial blow-up.

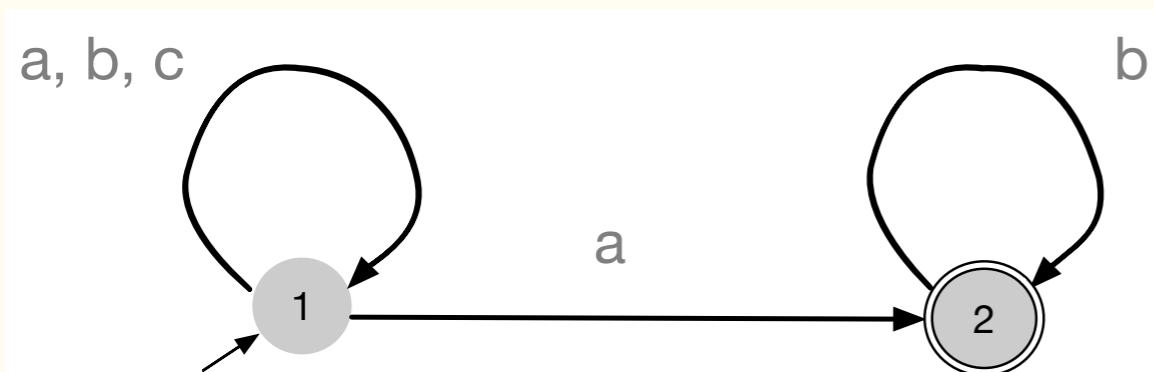
reversible = deterministic and co-deterministic

Reversible computations

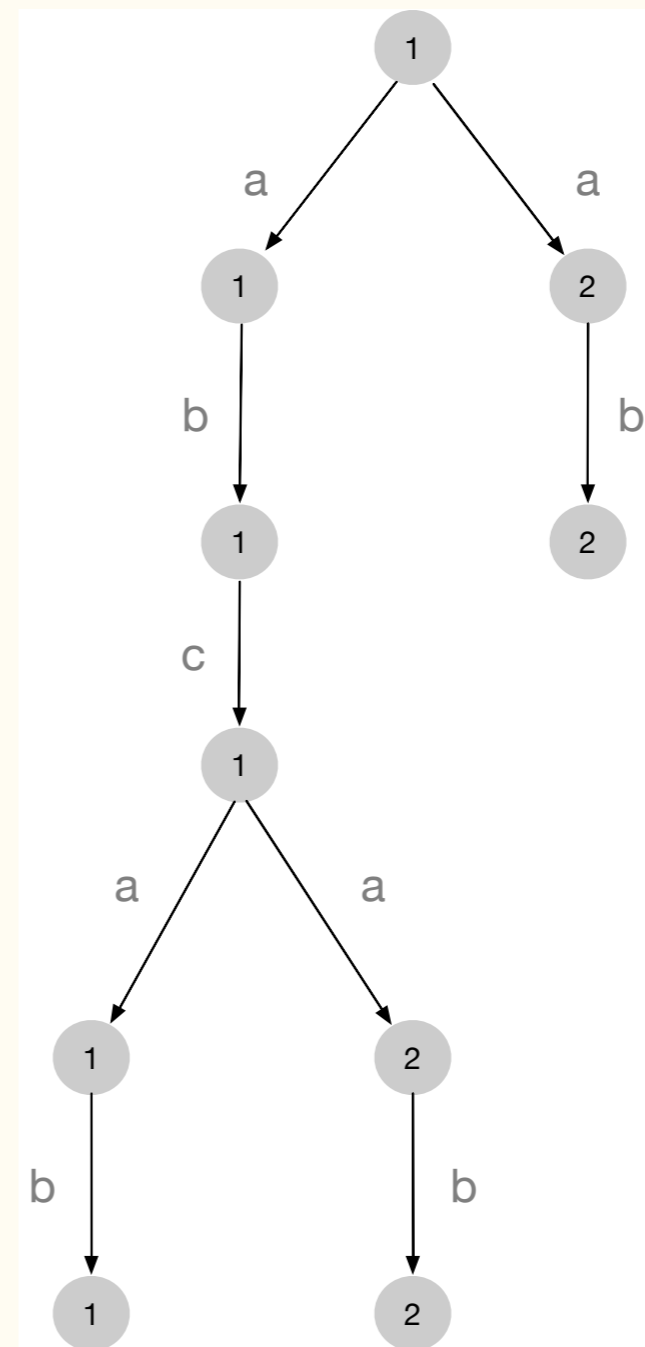
reversible: deterministic and co-deterministic

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

[Hopcroft-Ullman'67, Sipser'78]



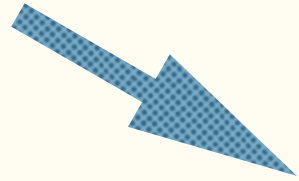
a b c a b



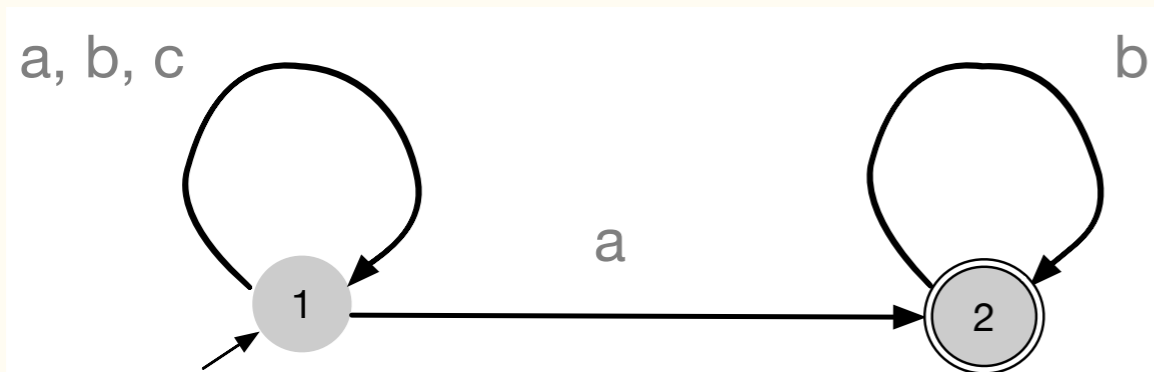
Reversible computations

reversible: deterministic and co-deterministic

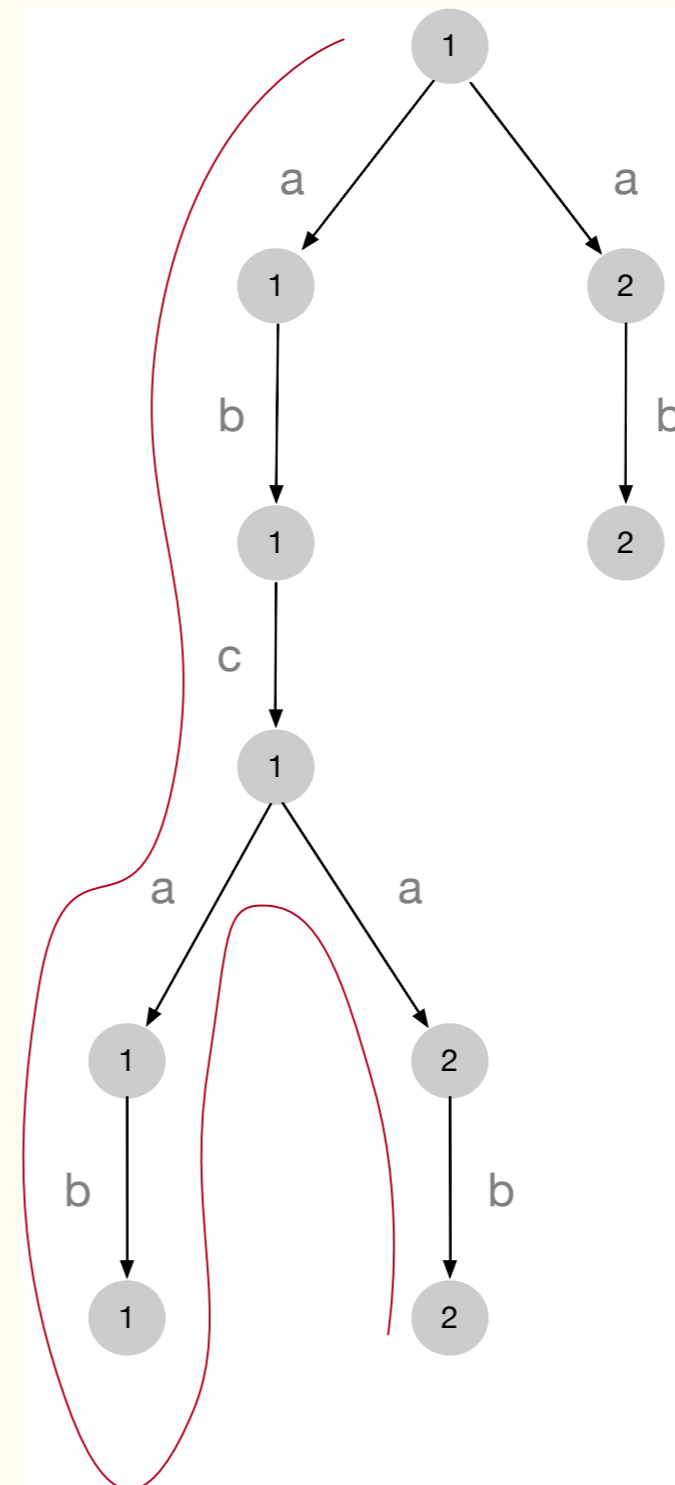
DFS of computation tree of **co**-deterministic one-way automata



reversible



a b c a b



Reversible transducers

reversible: deterministic and co-deterministic

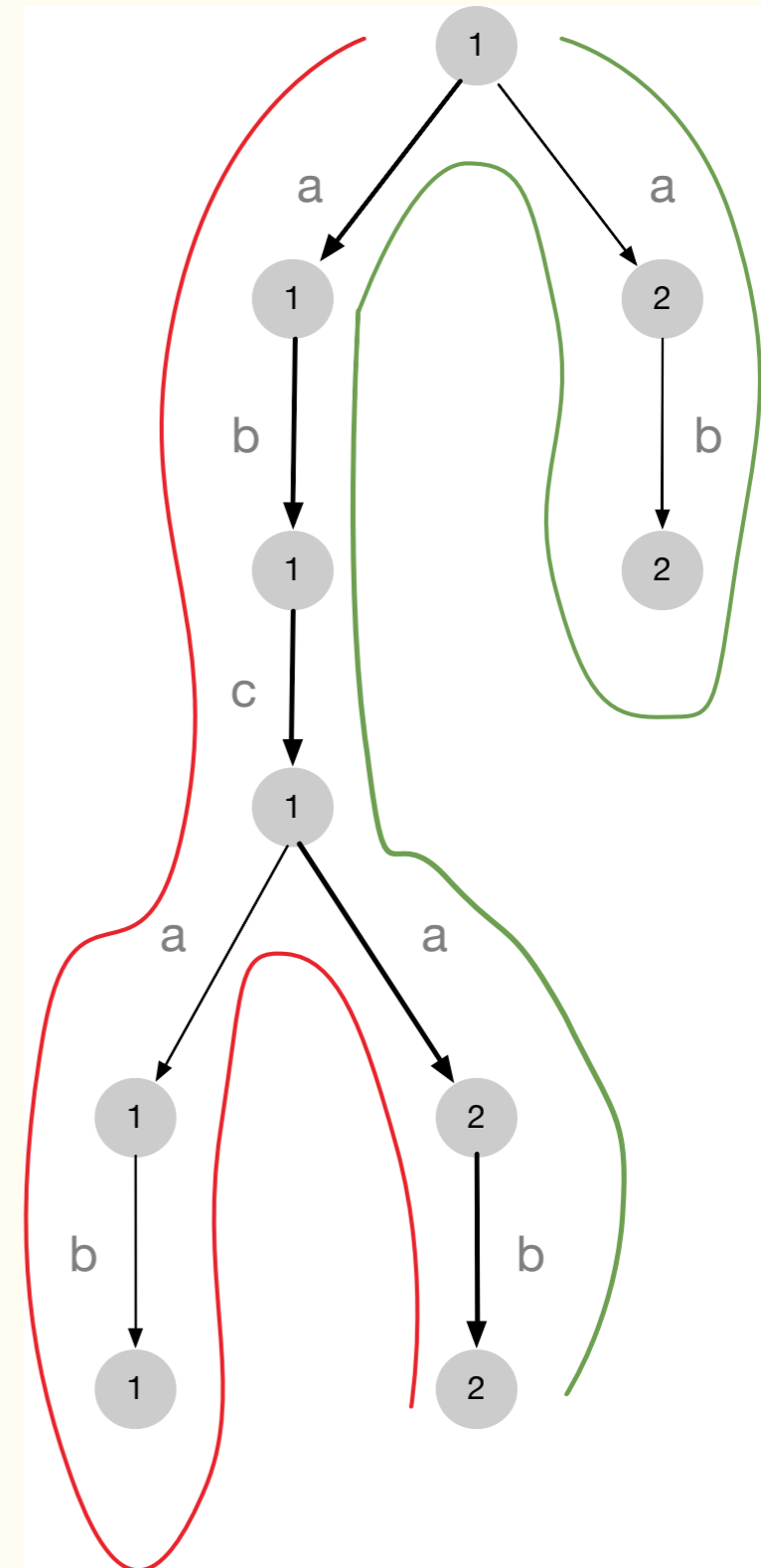
Computation tree of co-deterministic transducers

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

2DFT can be made reversible with exponential blow-up

[Dartois, Fournier, Jecker, Lhote'17]

From 2DFT T:

- ❖ build exp-size, co-deterministic “look-ahead” 1NFT LA
- ❖ build exp-size 1DFT R that outputs acc. run of T, using LA
- ❖ make LA, R reversible, compose and project on output:

input $\xrightarrow{\text{LA}}$ input + look-ahead $\xrightarrow{\text{R}}$ input + look-ahead + acc. run of T

Reversible transducers

reversible: deterministic and co-deterministic

- ❖ 2DFT can be made reversible
- ❖ reversible 2DFT can be composed easily

Open question: what about DSST?

Composing DSST through (reversible) 2DFT is doubly-exponential.
Any better construction? Lower bound?

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely-valued transductions

origin equivalence

Word transductions

automata = logic

properties

expressions

class membership problems

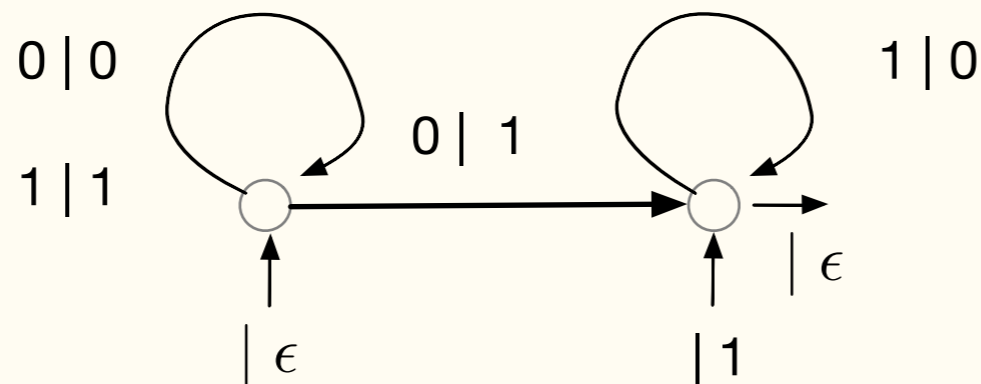
Equivalence problem

Rational expressions

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

(all rational operations are unambiguous)

Example (increment):



$$\text{copy} \cdot (0, 1) \cdot (1, 0)^* + (\epsilon, 1)(1, 0)^*$$

Regular expressions

$f \circ g$ composition

$f \odot g(w) = f(w)g(w)$ Hadamard product

Regular word functions are equivalent to expressions

$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid \text{reverse} \mid f \circ g \mid f \odot g$

$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid \text{reverse} \mid \text{duplicate} \mid f \circ g$

(all rational operations are unambiguous)

Regular expressions

Regular word functions are equivalent to expressions

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid \text{reverse} \mid f \circ g \mid f \odot g$$

Example

$$u_1 \# u_2 \# \dots u_k \# \mapsto u_1 u_3 \dots \# u_2 u_4 \dots$$

$$f_{\text{odd}} = ((\text{copy}(\#, \epsilon) \text{erase}(\#, \epsilon)))^*$$

$$f_{\text{even}} = \text{erase}((\text{copy}(\#, \epsilon) \text{erase}(\#, \epsilon))^* \text{copy})$$

$$f_{\text{odd}} \odot (\#, \#) \odot f_{\text{even}}$$

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

2DFT = regular expressions

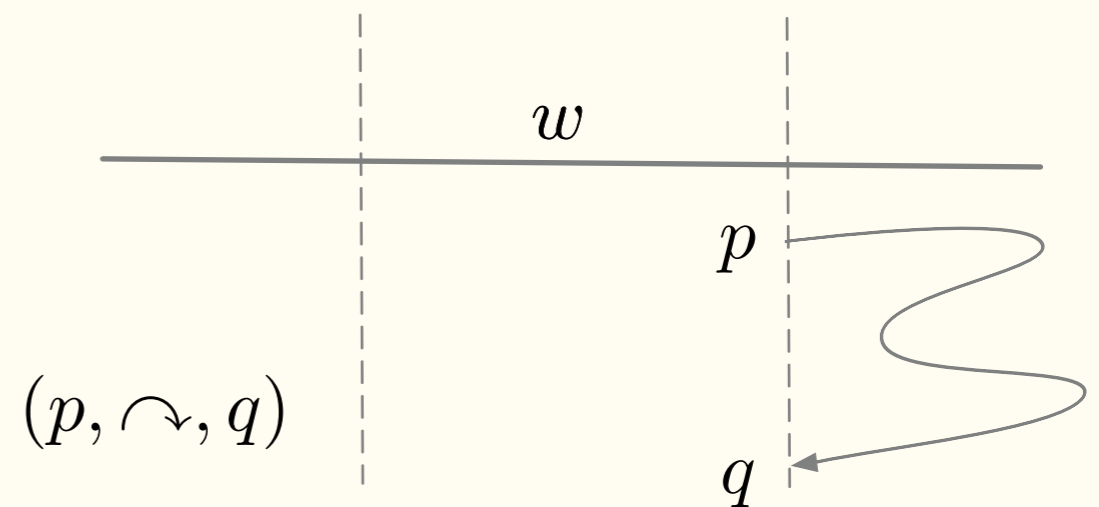
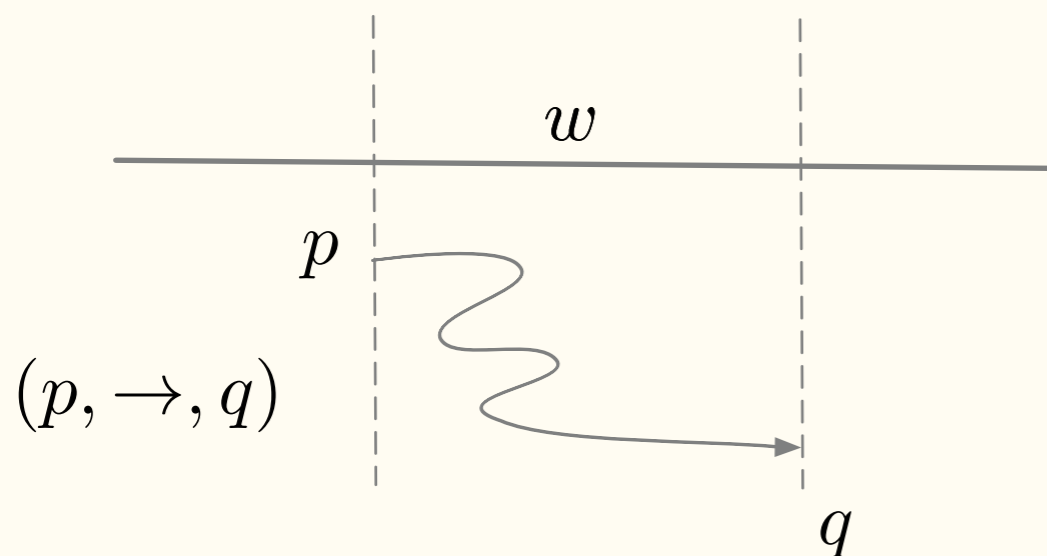
$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid \text{reverse} \mid f \circ g \mid f \odot g$$

From 2DFT to expressions: use algebra

[Dave, Gastin, Krishna'18]

Transition monoid of 2DFT associates a word (input factor) with set of triples

$$(p, D, q) \quad D \in \{\rightarrow, \leftarrow, \curvearrowright, \curvearrowleft\}$$



Word transductions

automata = logic

properties

expressions

class membership problems

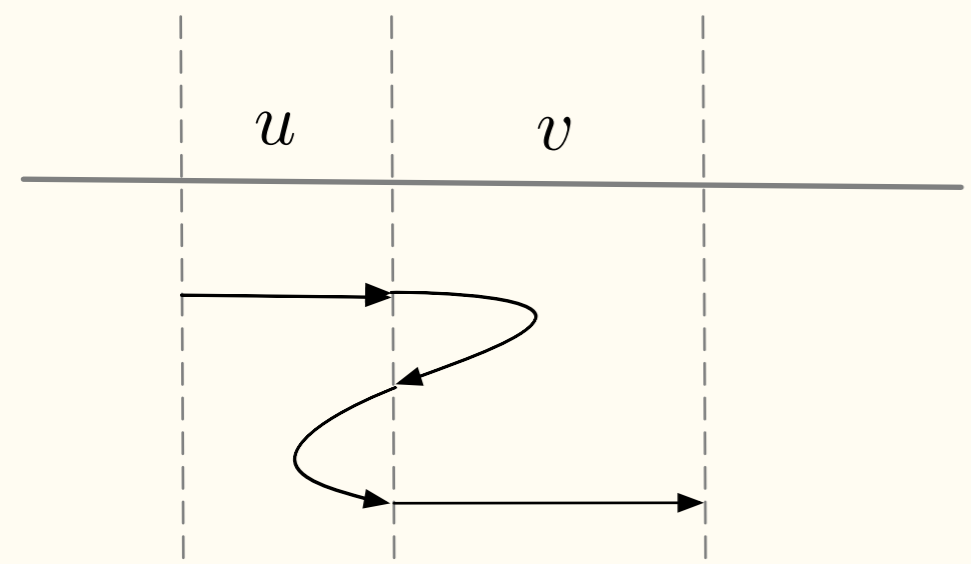
Equivalence problem

2DFT = regular expressions

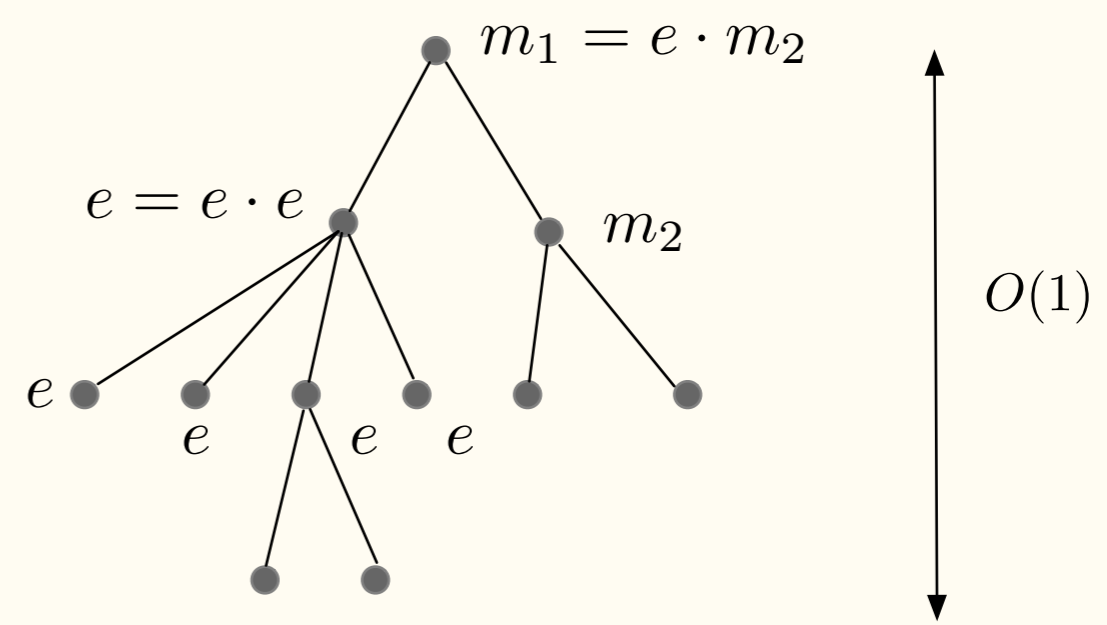
$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid \text{reverse} \mid f \circ g \mid f \odot g$$

Transition **monoid** of 2DFT associates a word (input factor) with set of triples

$$(p, D, q) \quad D \in \{\rightarrow, \leftarrow, \curvearrowright, \curvearrowleft\}$$



Simon's forest factorisation theorem (unambiguous version)



Word transductions

automata = logic

properties

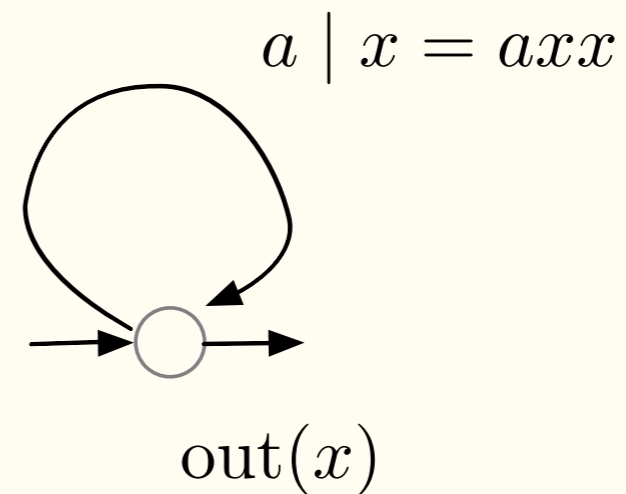
expressions

class membership problems

Equivalence problem

Beyond linear growth

Copyful SST: exponential output growth



Copyful DSST and HDT0L are equivalent.

HDT0L (Lindenmayer systems, '70)

- ❖ input alphabet A , output alphabet B , working alphabet C
- ❖ initial word u_0 from C^*
- ❖ family of morphisms $(\phi_a)_{a \in A}$ and final morphism $\psi : C^* \rightarrow B^*$

$w = a_1 \dots a_n$ maps to $\psi \circ \phi_{a_n} \circ \dots \circ \phi_{a_1}(u_0)$

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

Beyond linear growth

Polyregular word functions

[Bojanczyk'18]

Smallest class of word functions that

- ❖ is closed under composition
- ❖ contains the regular functions
- ❖ contains the squaring function

squaring $abcd \longrightarrow \underline{a}bcd \ a\underline{b}cd \ ab\underline{c}d \ abc\underline{d}$

Polyregular: **polynomial** output growth

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

Beyond linear growth

Polyregular word functions:

- ❖ smallest class of functions containing regular word functions, closed under composition, and containing squaring
- ❖ two-way deterministic transducers with **pebbles** (nested)
- ❖ for-transducers, polynomial list functions
- ❖ **MSO interpretations**

[Bojanczyk, Kiefer, Lhote'19]

[Bojanczyk, Daviaud, Krishna'18]

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

Polynomial growth

Square $abcd \rightarrow \underline{a}bcd \underline{a}bcd \underline{a}bcd \underline{a}bcd$

Example Prefixes $abcd \rightarrow a \ ab \ abc \ abcd$

$abcd \rightarrow abcd \# \xrightarrow{\text{square}} \underline{a}bcd \# \underline{a}bcd \# \underline{a}bcd \# \underline{a}bcd \# \underline{a}bcd \#$

$\xrightarrow{\text{copy}_\#}$	# a	# ab	# abc	# abcd
$\xrightarrow{\text{erase}}$	a	ab	abc	abcd

2DFT with 2 nested pebbles:

Two nested loops: first pebble moves left-to-right over the input;
second pebble copies content between left border and first pebble

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

Polynomial growth

Squaring $abcd \rightarrow \underline{a}bcd \underline{a}bcd \underline{a}bcd \underline{a}bcd$

Example Prefixes $abcd \rightarrow a \ ab \ abc \ abcd$

MSO interpretation: every output position is encoded by two input positions:

$$\text{dom}_a(x, y) = (y \leq x) \wedge a(y)$$

$$\text{ord}(x, y, x', y') = (x < x') \vee (x = x' \wedge y < y')$$

Polyregular functions are strictly weaker than copyful DSST:

A copyful DSST computes a polyregular word function if and only if it has **polynomial** output growth. [Douéneau-Tabot et al.'19]

k-layered DSST

- ❖ registers partitioned in k layers
- ❖ updates of registers from R_j use only registers from R_1, \dots, R_j
- ❖ updates of registers from R_j are copyless on R_j

k-layered DSST compute exactly the polyregular functions with output growth $O(n^{k+1})$

[Douéneau-Tabot et al.'19]

Polynomial growth

Composition-by-substitution

[Nguyen, Pradic'21]

$$f : A^* \longrightarrow C^* \quad (g_c)_{c \in C}, \quad g_c : A^* \longrightarrow B^*$$

$$\text{CbS}(f, (g_c)_c) : w \mapsto g_{c_1}(w) \dots g_{c_m}(w) \quad f(w) = c_1 \dots c_m$$

Comparison-free polyregular functions: smallest class containing regular word functions and closed under composition-by-substitution.

Equivalently: functions computed by **comparison-free pebble transducers**. They also form the smallest class containing regular word functions and cf-squaring:

$$abcd \longrightarrow \underline{a} abcd \underline{b} abcd \underline{c} abcd \underline{d} abcd$$

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely-valued transductions

origin equivalence

Word functions

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

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

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

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

[Choffrut 1977]

[Filiot, Gauwin, Reynier, Servais 2013]

Sequential functions

Overview

Word transductions

automata = logic

properties

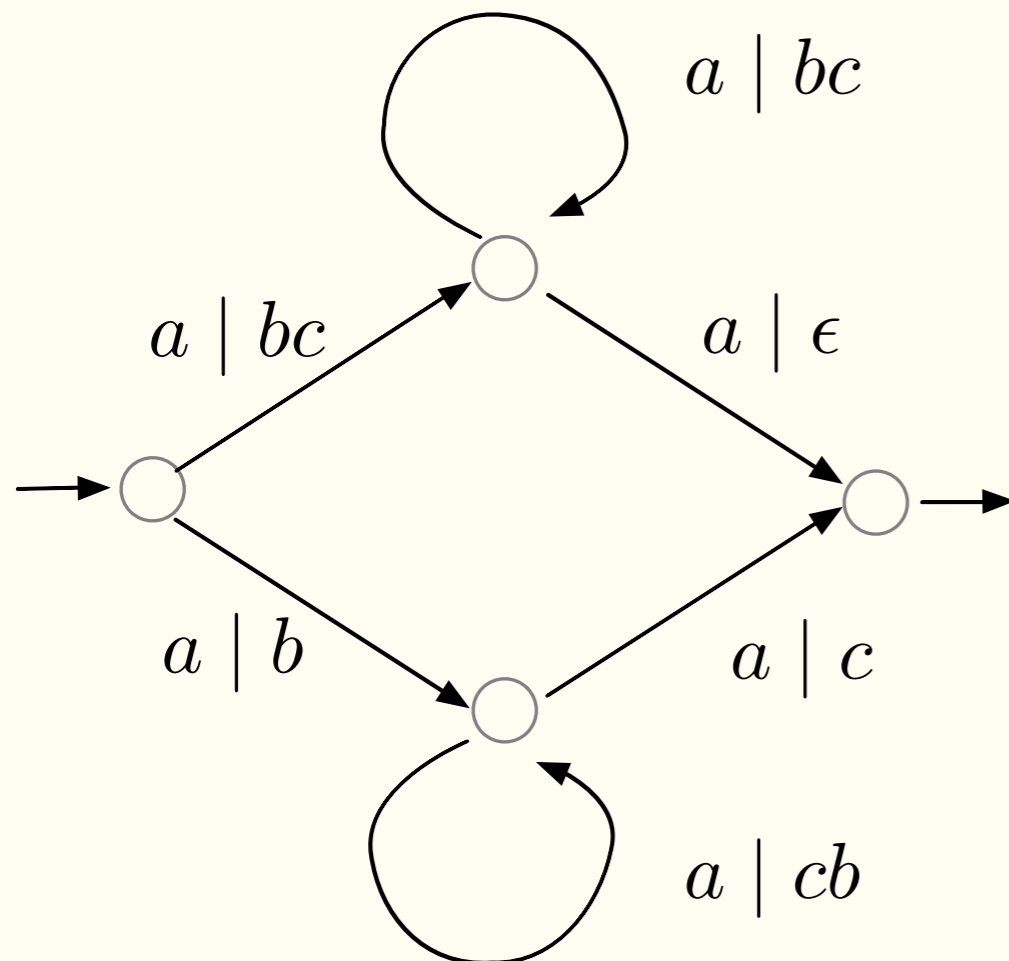
expressions

class membership problems

Equivalence problem

NL \curvearrowright **sequential** word functions: 1DFT

rational word functions: 1NFT



A single-valued 1NFT can be determined if and only if it satisfies the twinning property.

$$bc(bc)^\omega = b(cb)^\omega$$

Sequential functions

A rational word function is **sequential** iff

- ❖ it has bounded variation
- ❖ any two states are twinned

Bounded variation: Lipschitz property w.r.t. prefix distance

$$\Delta(f(u), f(v)) \leq c \cdot d(u, v)$$

$$\Delta(u, v) = |u| + |v| - 2 \cdot |u \wedge v|$$

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

Rational functions

Given a single-valued **2NFT**:

- ❖ it is decidable whether an equivalent **1NFT** exists

[Filiot, Gauwin, Reynier, Servais 2013]

- ❖ **ExpSPACE** algorithm

[Baschenis, Gauwin, M., Puppis 2017]

- ❖ if “yes”: construction of 2-exp size equivalent **1NFT**

Lower bounds

- ❖ PSPACE-hard to decide whether equivalent **1NFT** exists
- ❖ the **size** of the **1NFT** is optimal

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem

Rational functions

If the given 2NFT is not single-valued: undecidable if it computes a rational relation. [Baschenis, Gauwin, M., Puppis 2015]

Reduction from PCP

$$f, g : A^* \longrightarrow B^*$$

Good encodings $(wz, w\#^{|z|})$ with $z = f(w) = g(w)$

Bad encodings

$$(wz, w\#^n) \text{ with}$$

$$n \neq |z| \quad \text{or} \quad z \neq f(w) \quad \text{or} \quad z \neq g(w)$$

Overview
Word transductions
automata = logic
properties
expressions
class membership problems
Equivalence problem

Rational functions

If the given 2NFT is not single-valued: undecidable if it computes a rational relation.

Bad encodings

$(wz, w\#^n)$ with

$$n \neq |z| \quad \text{or} \quad z \neq f(w) \quad \text{or} \quad z \neq g(w)$$

A 2NFT can generate all bad encodings:

$$\begin{array}{ccc}
 w = w_1 a w_2, & z = z_1 t z_2 & \text{if } n = |z| \\
 \downarrow & \downarrow & \\
 \# |f(w_1)| & \# |z_2| & \text{then error found } t \neq f(a)
 \end{array}$$

Rational functions

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem

If the given 2NFT is not single-valued: undecidable if it computes a rational relation.

Bad encodings

$(wz, w\#^n)$ with

$n \neq |z|$ or $z \neq f(w)$ or $z \neq g(w)$

If PCP has no solution: all pairs are bad encodings, so rational.

If PCP has solution $z = f(w) = g(w)$ consider $(w^n z^m, w^n \#^m |z|)$

Pumping turns good encodings into bad ones, so the relation cannot be rational.

Rational functions

Given a single-valued **2NFT** the existence of an equivalent **1NFT** is decidable in **ExpSPACE**.

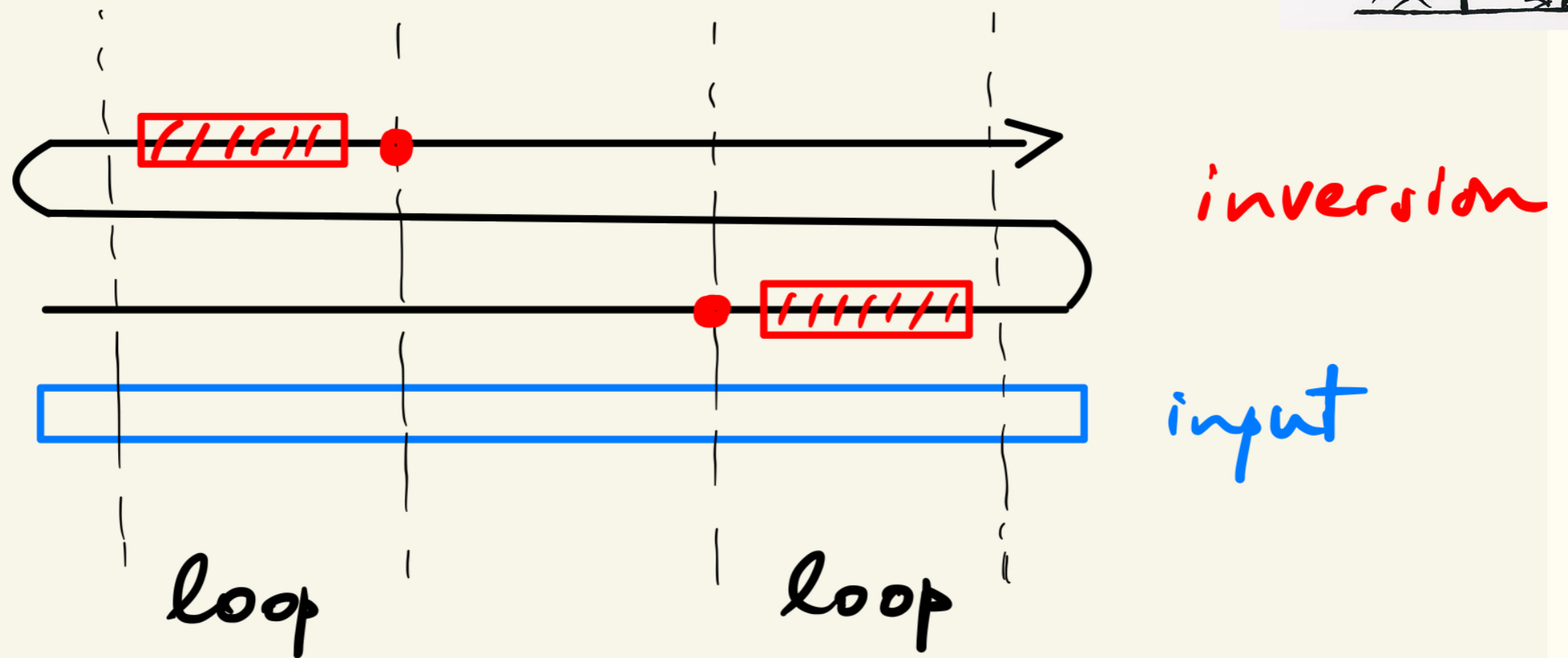
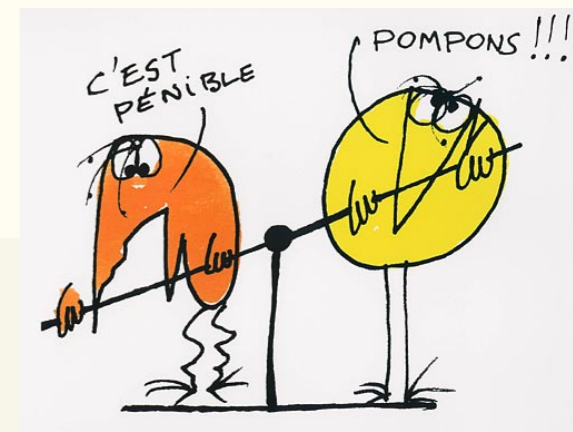
[Baschenis, Gauwin, M., Puppis 2017]

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

- ❖ if $R = (a + b)^*$ no equivalent 1NFT
- ❖ if $R = (ab)^*$ equivalent 1NFT exists

↑
period 2

Key tool: **inversions** + word combinatorics



The output between the **red dots** has exponentially-bounded period

$$v_0 v_1^m v_2 v_3^n v_4 = w_0 w_1^n w_2 w_3^m w_4$$

Pumping the two loops yields a bounded period for the middle parts.

Some open problems

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

- ❖ PSPACE lower bound for deciding if a rational function is sequential
 - better lower bound?
- ❖ Better complexity for “2NFT to 1DFT”?
- ❖ Extension from single-valued to finitely-valued transductions?
- ❖ Canonical DSST? Perhaps easier: minimising the number of registers?

We can compute the minimal number of registers for concatenation-free (cf) DSST. Moreover, k -cf-DSST are equivalent to $2k$ -sweeping 2DFT.

Overview

Word transductions

automata = logic

translations between models

expressions

algebra

Equivalence problem

finitely valued transducers

origin equivalence

Sequential functions

sequential: 1DFT, rational: 1NFT

[Choffrut'79]

Myhill-Nerode theorem for 1DFT, so canonical (minimal) transducer

$$f : A^* \longrightarrow B^* \qquad \hat{f}(u) = \bigwedge_w f(uw) \quad (\text{largest common prefix})$$

Define $u \equiv_f v$ if

$$u \equiv_{\text{dom}(f)} v \quad \text{and} \quad \forall w, \quad \hat{f}(u)^{-1} f(uw) = \hat{f}(v)^{-1} f(vw)$$

- ❖ A word function is **sequential** iff the associated congruence \equiv_f has finite index.
- ❖ The associated congruence defines the canonical (minimal) transducer computing the function.

Rational functions

1NFT are equivalent to 1DFT with regular look-ahead

[Elgot-Mezei'65]

Example: “subsequence of letters occurring more than once”

$$w \mapsto \text{proj}_D(w) \quad D = \{a : |w|_a > 1\}$$

\equiv_f has infinite index:

$$\hat{f}(ab^n) = \epsilon$$

$$ab^m \equiv_f ab^n \text{ iff } m = n$$

Canonical look-ahead $\text{LA}(f)$

[Reutenauer-Schützenberger'91]

$$u \equiv_{l_a} v \text{ if } \Delta(f(wu), f(wv)) \leq c, \quad \forall w$$

Example: $u \equiv_{l_a} v \text{ iff } \text{alph}(u) = \text{alph}(v)$

Rational functions

A word function f is **rational** if and only if its canonical look-ahead $\text{LA}(f)$ has finite index and $f + \text{LA}(f)$ is sequential.

[Reutenauer-Schützenberger'91]

Example: “subsequence of letters occurring more than once”

$$w \mapsto \text{proj}_D(w) \quad D = \{a : |w|_a > 1\}$$

$f + \text{LA}(f)$:

$$(a_1, \text{alph}(a_2 \dots a_n))(a_2, \text{alph}(a_3 \dots a_n)) \dots (a_n, \emptyset) \mapsto \text{proj}_D(a_1 \dots a_n)$$

is sequential

Canonical 1DFT for $f + \text{LA}(f)$ = canonical **bimachine** for f

First-order transductions

The **domain** and **order** formula are FO (instead of MSO).

Example: $w \mapsto w w$ FO

❖ domain

$$\text{dom}_{a,1}(x) = \text{dom}_{a,2}(x) \equiv a(x)$$

❖ order

$$\text{ord}_{1,1}(x, y) = \text{ord}_{2,2}(x, y) = (x < y)$$

$$\text{ord}_{1,2}(x, y) = \text{true}$$

Example: $w \mapsto a^{|w|/2}$ not FO

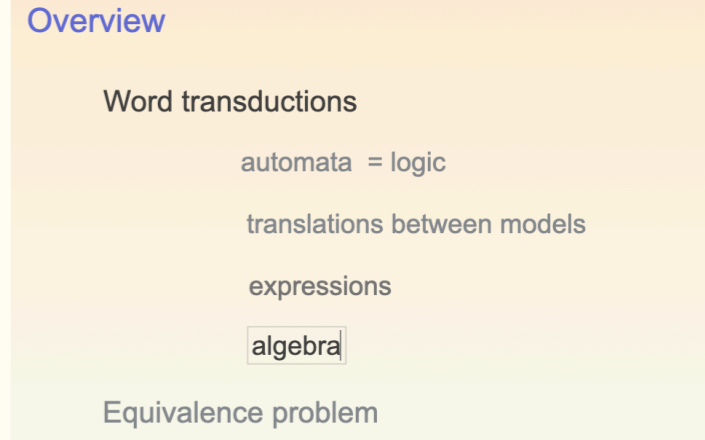
❖ domain

$$\text{dom}_{a,1}(x) = a(x) \wedge \text{even}(x)$$

❖ order

$$\text{ord}_{1,1}(x, y) = x < y$$

Algebra



Long line of research on algebra for word 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 = FO logic

[McNaughton, Papert'71]

star-free = aperiodic

[Schützenberger'65]

Can we decide if a regular transduction is a FO transduction?

First-order transductions (FOT)

A word function f is a [first-order transduction](#) if and only if the canonical look-ahead $LA(f)$ is aperiodic and the canonical 1DFT for $f + LA(f)$ has an aperiodic transition monoid.

It is decidable if a rational word function f is a [first-order transduction](#).

[Filiot, Gauwin, Lhote'16]

No decision procedure for regular word functions so far, but:

FOT = aperiodic 2DFT = aperiodic DSST

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

Open problem

Can we decide whether a regular word function is FOT ?

Issue here: come up with a canonical 2DFT/DSST. Not even clear for sweeping transducers.

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

Equivalence problem

Given two transducers, do they compute the same relation?

The equivalence problem for **non-deterministic one-way** transducers (**1NFT**) is undecidable.

[Fischer-Rosenberg, Griffiths'68]

Reduction from PCP

$$f, g : A^* \longrightarrow B^*$$

[Ibarra'77]

$$(u z, c^m) \in R_f \text{ if } m \neq |f(u)| \text{ or } z \neq f(u)$$

R_f is **rational**: if $m = |f(u)|$ guess $u_1 a u_2 z_1 t z_2$ with

$$t \neq f(a) \qquad m = |f(u_1)| + |t z_2|$$

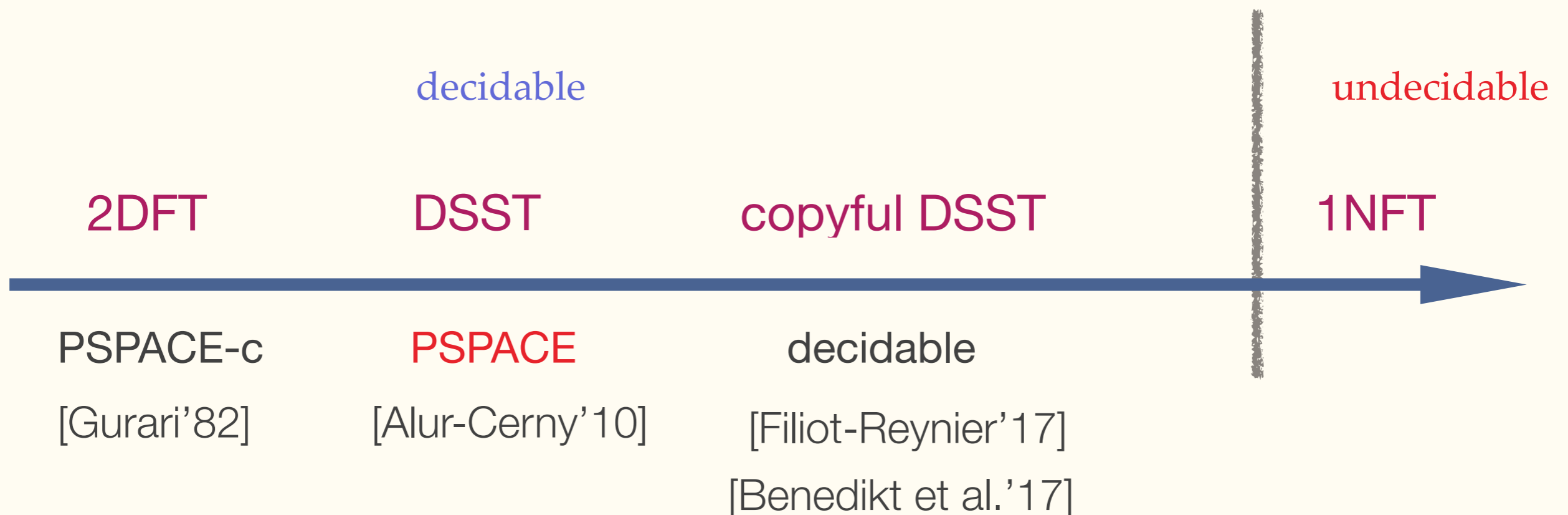
$$R_f \cup R_g = (A \cup B)^* \times c^* \text{ iff PCP has no solution}$$

Equivalence problem

Given two transducers, do they compute the same relation?

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

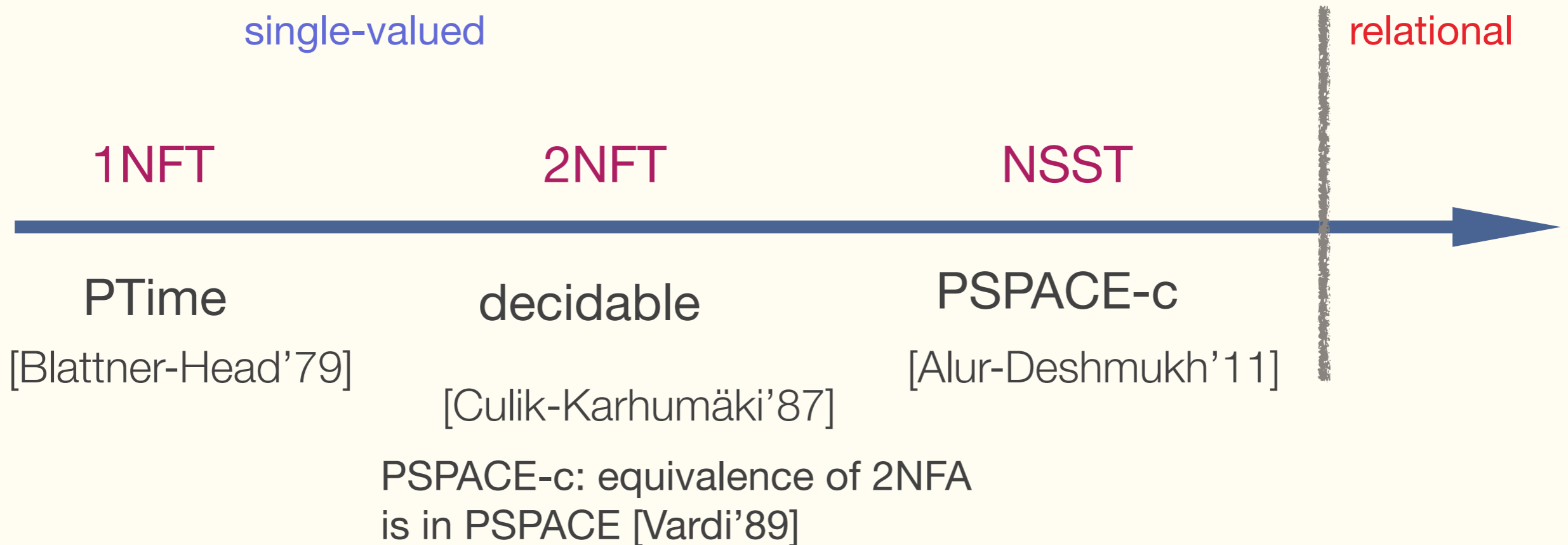
[Fischer-Rosenberg, Griffiths'68]



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

Single-valuedness 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

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

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 $x = b c \quad y = z = b \quad t = c b$

[proved 1986 by Albert & Lawrence, and Guba]

Word transductions

automata = logic

properties

expressions

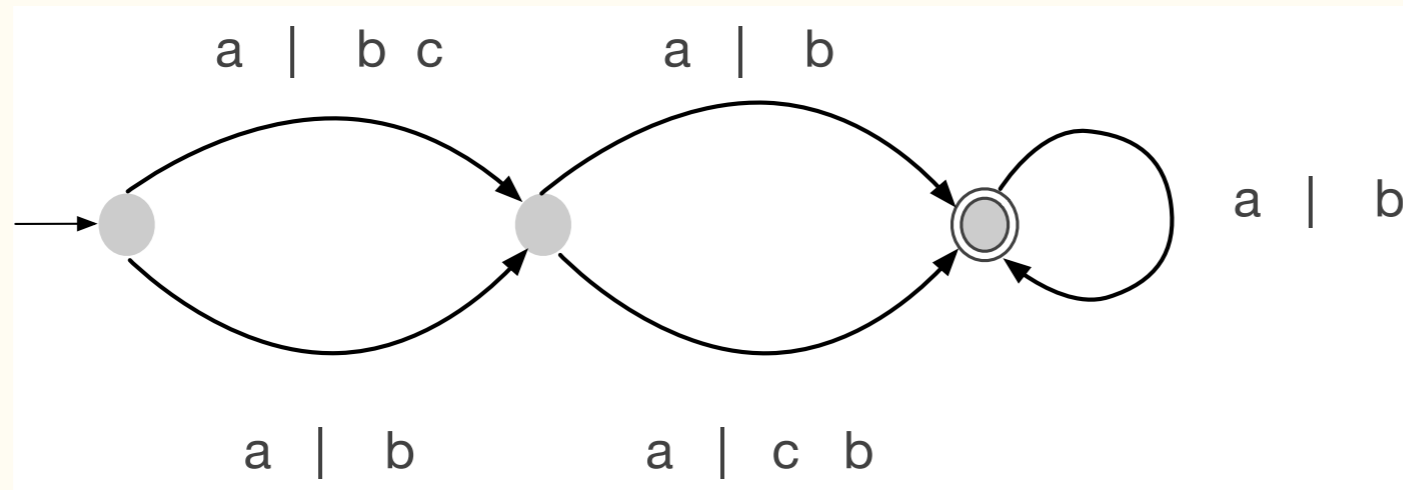
class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

Equivalence of k -valued 1NFT 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

Word transductions

automata = logic

properties

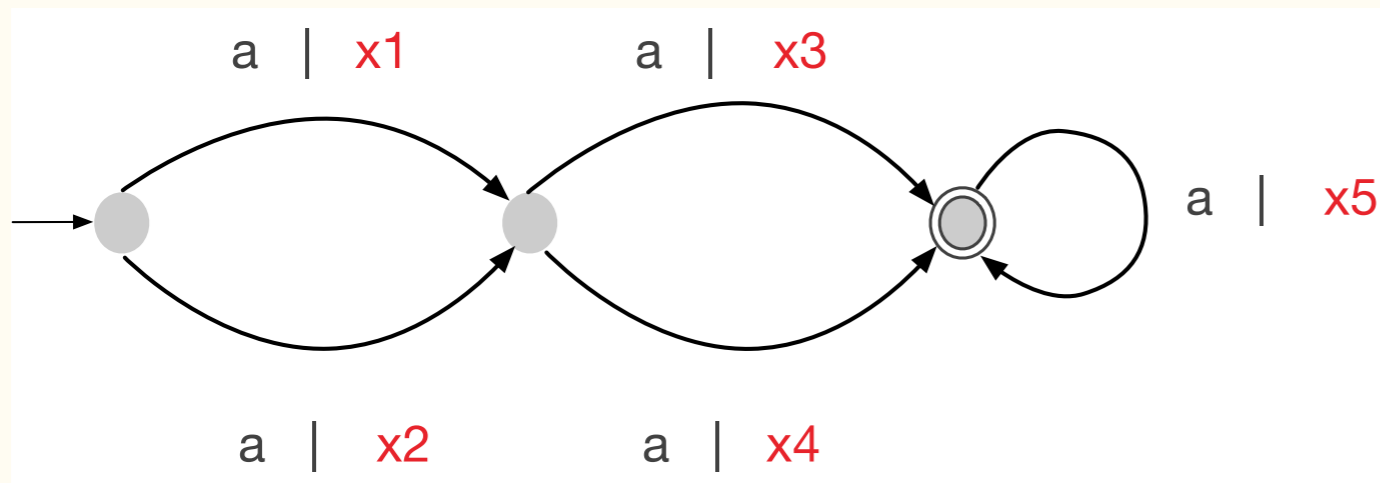
expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence



k-valued, one-way implies
bounded outdegree

Given two transducers, replace output words by variables

$$x_1 \ x_3 \ \underbrace{x_5 \ \dots \ x_5}_m = x_2 \ x_4 \ \underbrace{x_5 \ \dots \ x_5}_m \quad \text{on input } a^{m+2}$$

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

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

equivalent to
(Ehrenfeucht)

$$\bigwedge_{w \in \Sigma^{\leq m}} \bigvee \text{ groups } S$$

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

Equivalence of **k-valued** 1NFT
is decidable.

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

Find m effectively:

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

“Left quotient”

$$T_a(w) := T(aw) \quad a \in \Sigma$$

(same number of states and outdegree, so same m)

$T_1 \equiv_m T_2$ if T_1, T_2 equivalent over $\Sigma^{\leq m}$

Show inductively $T_1 \equiv_N T_2$ iff $T_1 \equiv_m T_2$ for all $N \geq m$ using T_a

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

Equivalence of k -valued NSST is decidable.

[M., Puppis 2019]



First **issue**: bounded outdegree no longer obvious: e.g. $x = (bc)^j \ x \ (bc)^{n-j}$

Normalization: invariant about **periods** of **register** and **gaps**

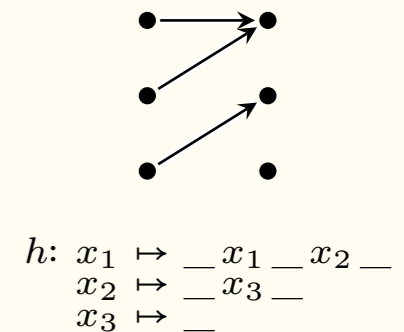
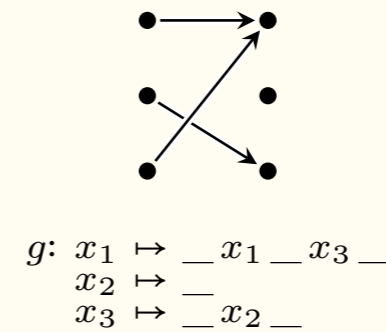
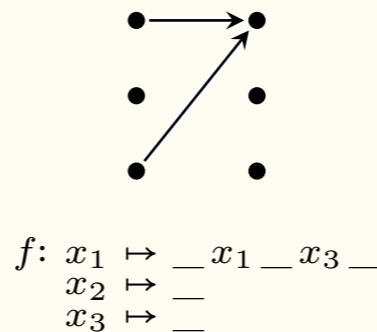
Example (left): $c \ \underbrace{bc \ bc}_{x} \ bc \ bc \ bc \ bc = \text{final output}$

Equivalence of **k-valued** NSST is decidable.

Overview	
Word transductions	
automata	= logic
properties	
expressions	
class membership problems	
Equivalence problem	
finitely valued transducers	
origin equivalence	

Several normalization steps:

- ❖ make registers non-erasing, non-permuting



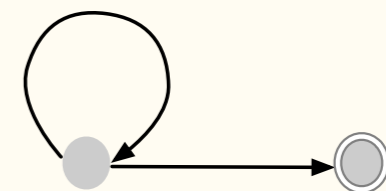
- ❖ associate invariants with states: information about boundedness or periodicity of register content, resp. (future) gap content

left gap bounded (c)

right gap period bc

x: period bc

a | x = x b c



a | x = c x

c b c b c b c b c b c b c

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

Equivalence of k -valued NSST is decidable.

Let T be k -valued NSST.

- ❖ Based on = invariants, two transitions (between the same pair of states) are either always output-equivalent, or never.
- ❖ We can show that the outdegree of each state is bounded by a constant depending on k and the number of states and registers.

Consequence: for fixed alphabets/number of registers/states, the set of k -valued NSST is **finite**.

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

Equivalence of **k-valued** NSST is decidable.

Consequence: for **fixed** alphabets/number of registers/states and outdegree, the set \mathcal{C} of k-valued NSST is finite.

Next step: show that for any T as above and input word u , the “ u -quotient” belongs to \mathcal{C}

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

(Naive construction preserves the number of states/registers, but not the outdegree.)

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

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 k -valued NSST in \mathcal{C} .

$$T_1 \equiv_N T_2$$

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 \mathcal{C}

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

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

origin equivalence

Equivalence of **k-valued** NSST is decidable.

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

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

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

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

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

$$\text{iff} \quad T_1 \equiv_r T_2$$

Open questions

Decomposition theorem for finitely-valued NSST?

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 then:

$\text{NSST} = 2\text{NFT}$ holds in finite-valued case (conjectured).

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

copyful transducers

origin equivalence

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:

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

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

Concatenation $u = (u_1, u_2), v = (v_1, v_2)$

$$(u_1, u_2) \circ (v_1, v_2) = (u_1v_2 + v_1, u_2v_2)$$

Equivalence of **copyful** DSST is decidable.

Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

copyful transducers

origin equivalence

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

polynomial automata

[Benedikt et al.'17]

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

Build the product of (encodings of) DSST T_1 , T_2 ;

output = difference of output registers

$T_1 = T_2$ iff the output is constantly 0

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely valued transducers
 - copyful transducers
 - origin equivalence

Zeroneess of polynomial automata is decidable.


[Benedikt et al.'17], [Seidl,-Maneth-Kemper'15]

[Bojanczyk, SIGLOG'19]

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

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

polynomial invariants



Overview

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

finitely valued transducers

copyful transducers

origin equivalence

Origin equivalence

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

[Bojańczyk'14]

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

[Bojańczyk'14]

Regular word functions with origin information enjoy a Myhill-Nerode congruence: machine-independent characterisation

derivatives: left-right, left, right

$$f(w) = w^R w$$

$$f_{\ell,r}(v) = r v^R \ell v r$$

$$f_{\ell}(v) = v^R \ell v$$

$$f_r(v) = v^R v$$

A word function with origin semantics is regular iff it has finitely many left and finitely many right derivatives.

First-order definable regular word functions have an effective characterisation

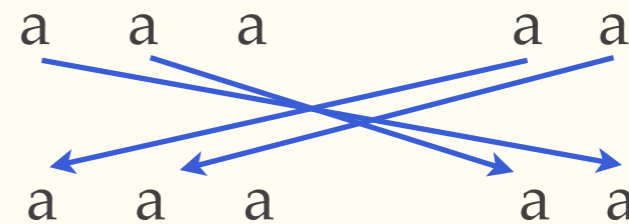
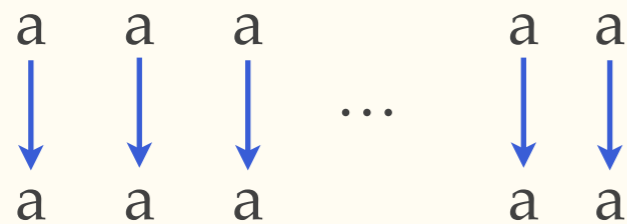
Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely valued transducers
 - copyful transducers
 - origin equivalence

Origin equivalence

“Tag” each output symbol with its origin

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



not origin-equivalent

Origin-equivalence of 2NFT is decidable: PSPACE-complete.

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

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

Word transductions

automata = logic

properties

expressions

class membership problems

Equivalence problem

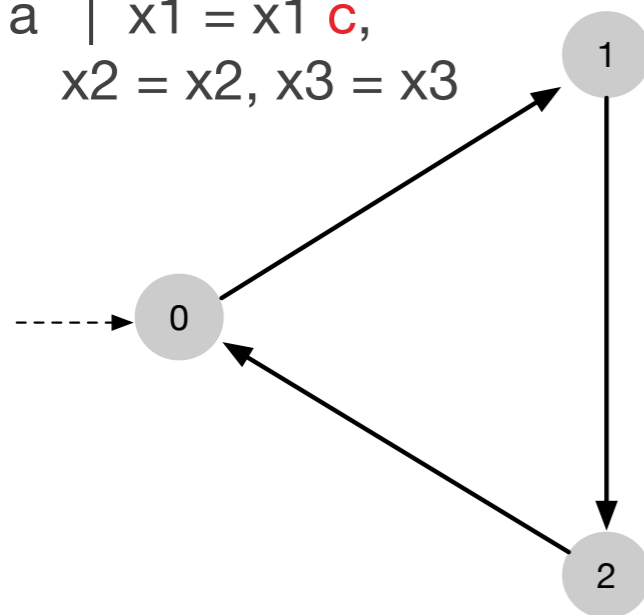
finitely valued transducers

copyful transducers

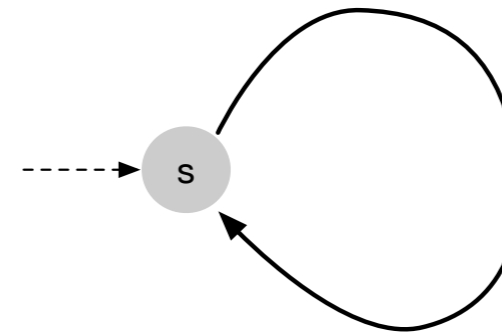
origin equivalence

Origin-equivalence of DSST in PSPACE.

$a \mid x_1 = x_1 \mathbf{c},$
 $x_2 = x_2, x_3 = x_3$



$a \mid x_2 = x_2 \mathbf{c},$
 $x_3 = x_3, x_1 = x_1$



$a \mid y_1 = y_2 \mathbf{c},$
 $y_2 = y_3, y_3 = y_1$

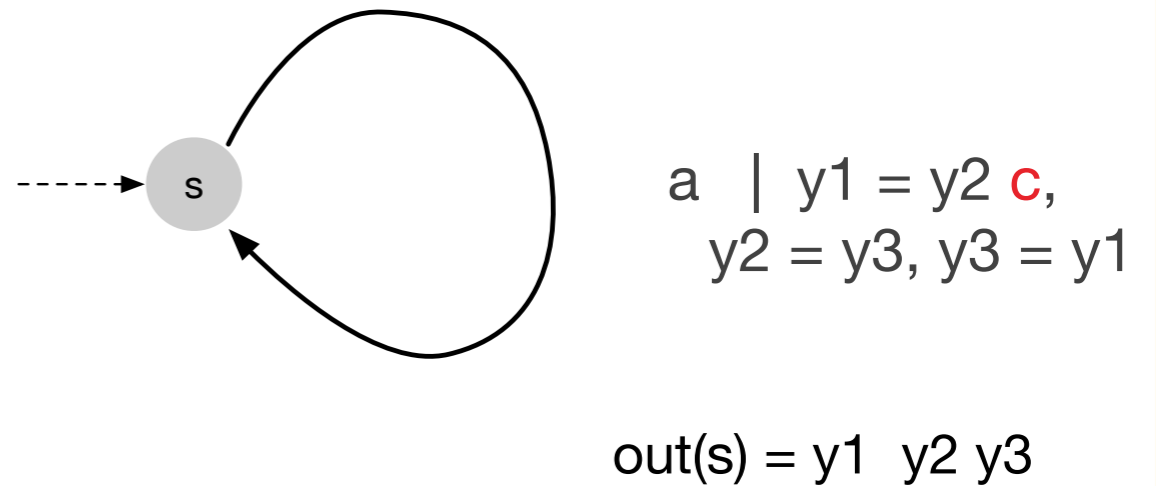
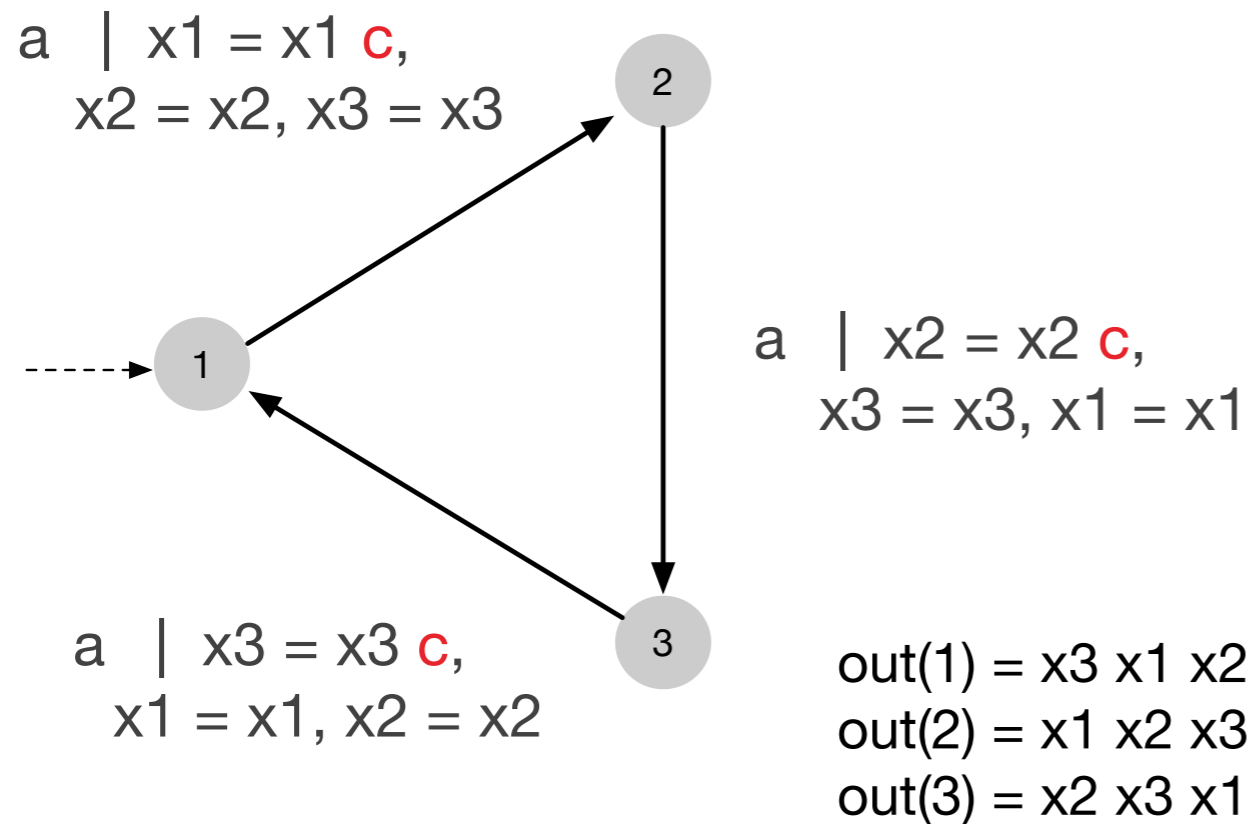
$a \mid x_3 = x_3 \mathbf{c},$
 $x_1 = x_1, x_2 = x_2$

$out(0) = x_3 x_1 x_2$
 $out(1) = x_1 x_2 x_3$
 $out(2) = x_2 x_3 x_1$

$out(s) = y_1 y_2 y_3$

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

Origin-equivalence of DSST in PSPACE.



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

$$(1,s) \leftarrow (3,s) \quad x3 \text{ c } x1 \ x2 = y2 \text{ c } y3 \ y1$$

$$(3,s) \leftarrow (2,s) \quad x3 = y3, \ x1 \ x2 \text{ c} = y1 \ y2 \text{ c}$$

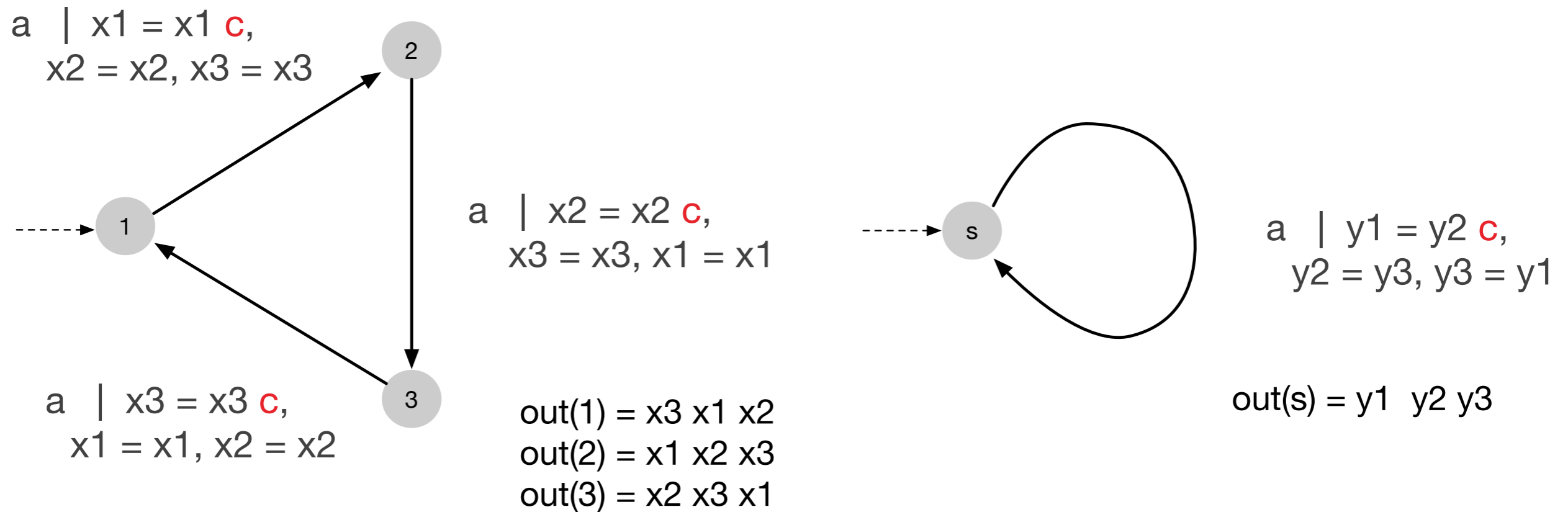
$$(2,s) \leftarrow (1,s) \quad x3 = y1, \ x1 \text{ c } x2 = y2 \text{ c } y3$$

$$x3 = y2, \ x1 \ x2 = y3 \ y1$$

$$x1 \ x2 = y1 \ y2$$

$$x1 = y2, \ x2 = y3$$

Origin-equivalence of DSST in PSPACE.



$$out(1) = out(s) \quad x_3 \ x_1 \ x_2 = y_1 \ y_2 \ y_3$$

Invariants: at state 1
 at state 2
 at state 3

$$x_1 = y_2, x_2 = y_3, x_3 = y_1$$

$$x_1 = y_1, x_2 = y_2, x_3 = y_3$$

$$x_1 = y_3, x_2 = y_1, x_3 = y_2$$

Origin-equivalence of **copyful DSST** is decidable.

Copyful: registers can occur multiple times in RHS

Algorithm:

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

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

Termination: if no inconsistency detected during propagation

Is based on Ehrenfeucht's conjecture + Makanin

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely valued transducers
 - copyful transducers**
 - origin equivalence

Origin-equivalence of **copyful DSST** is decidable.

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely valued transducers
 - copyful transducers**
 - origin equivalence

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

- ❖ additional register **m**, additional output symbol #
- ❖ update $x := a y x b$ replaced by $x : a \mathbf{m} y x b \mathbf{m}$
- ❖ $\mathbf{m} := \mathbf{m} \#$ at each transition

Unary output alphabet

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely valued transducers
 - copyful transducers
 - origin equivalence

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

For **classical** equivalence unary alphabets are presumably easier:

Equivalence of **copyful** DSST is in Ackermann (Benedikt et al.'17)

Equivalence of **copyful** DSST with **unary output** alphabet is in PTIME (Karr's algorithm, cf. MüllerOlm-Seidl'04)

Open questions

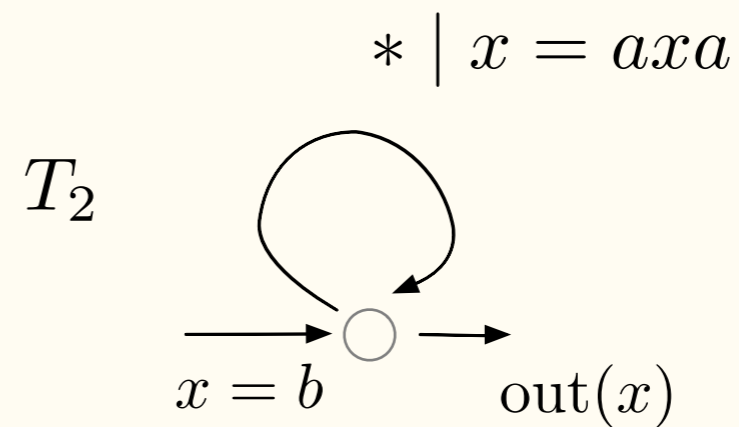
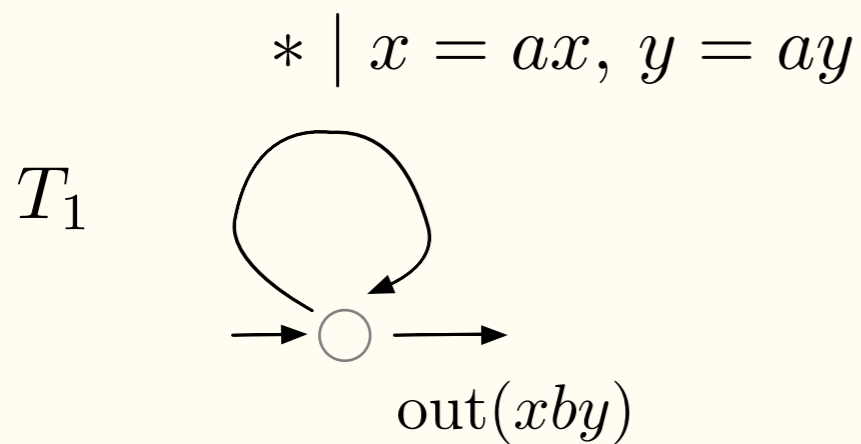
- ❖ Complexity of equivalence of deterministic SST?
- ❖ Same for origin-equivalence of DSST?
- ❖ Decidability of equivalence for pebble transducers? Or even comparison-free pebble transducers?

Resynchronizations

Overview

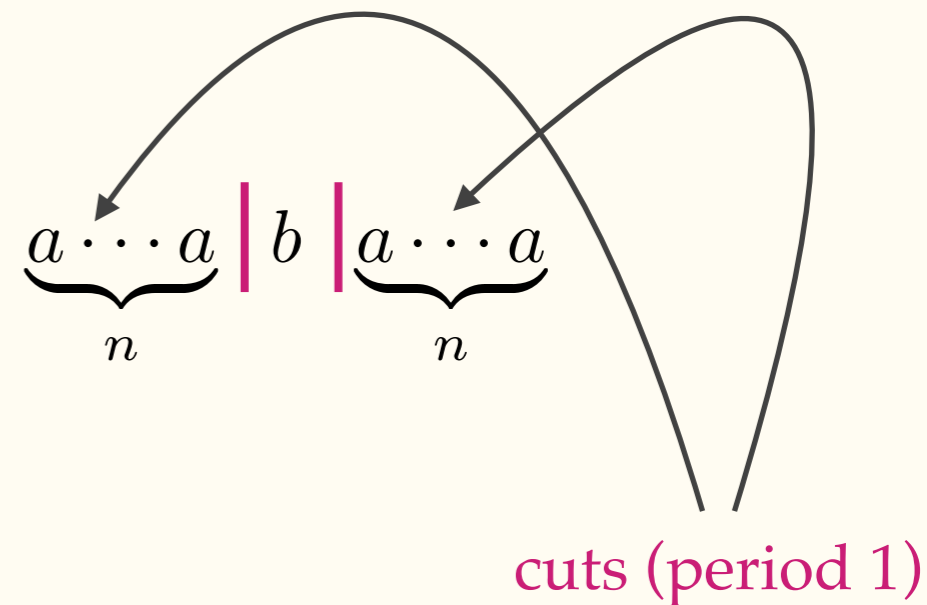
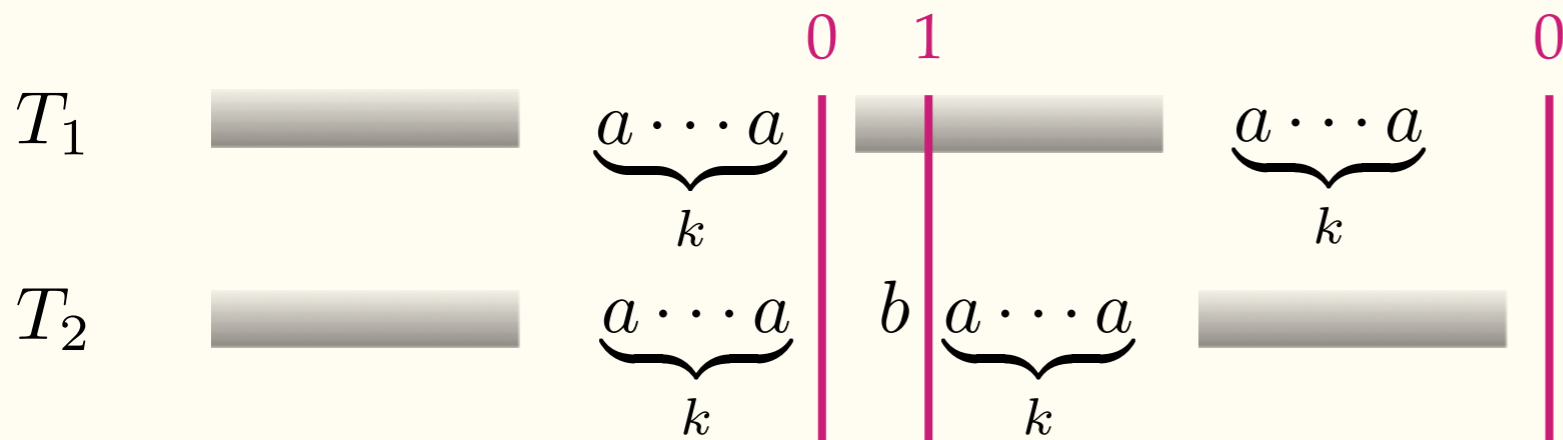
- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely valued transducers
 - copyful transducers
 - origin equivalence

Classical equivalence of 2DFT or DSST is difficult because same outputs can be generated in very different manners.



[Filiot-Jecker-Löding-Winter'22]

delay between runs with some output



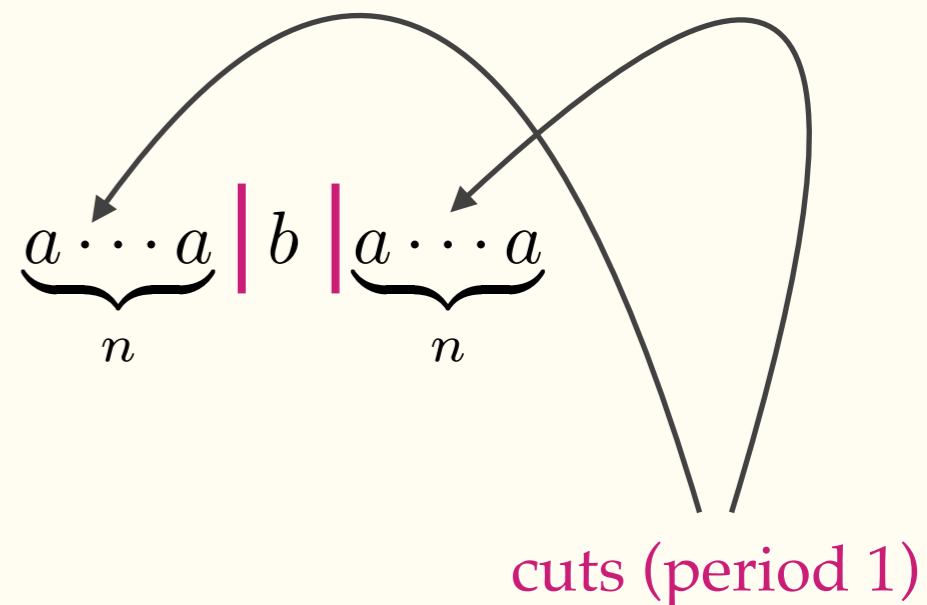
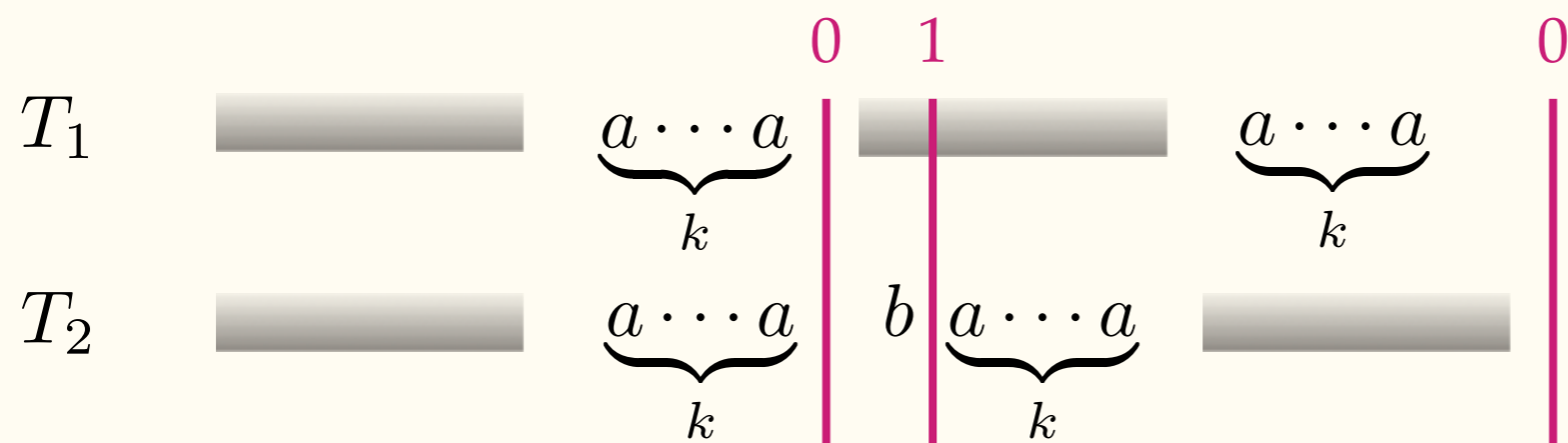
Resynchronizations

[Filiot-Jecker-Löding-Winter'22]

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely valued transducers
 - copyful transducers
 - origin equivalence

delay between runs with same output



Fix a period p and a delay d .

Two runs with same output have delay d w.r.t. period p if the delay (= length difference of output) measured at cuts is at most d .

Resynchronizations

[Filiot-Jecker-Löding-Winter'22]

Overview

- Word transductions
 - automata = logic
 - properties
 - expressions
 - class membership problems
- Equivalence problem
 - finitely valued transducers
 - copyful transducers
 - origin equivalence

Fix a period p and a delay d .

A letter-to-letter 1NFT R can be constructed such that R takes a run r_1 of SST T_1 and outputs runs r_2 of SST T_2 with $\text{delay}(r_1, r_2)$ at most d .

Given single-valued NSST T_1, T_2 one can compute p and d such that $T_1 = T_2$ iff $T_1 = R(T_2)$.

Given single-valued NSST T there exist p and d such that:

$T_1 = T_2$ for some T_2 with k registers iff $T_1 = R(T_2)$ for some T_2 with k registers

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!