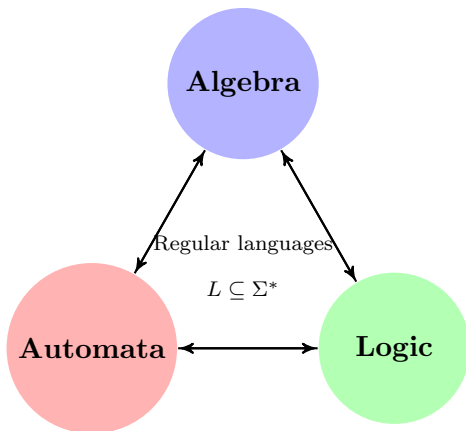


Automata, Logic and Algebra for (Finite) Word Transductions

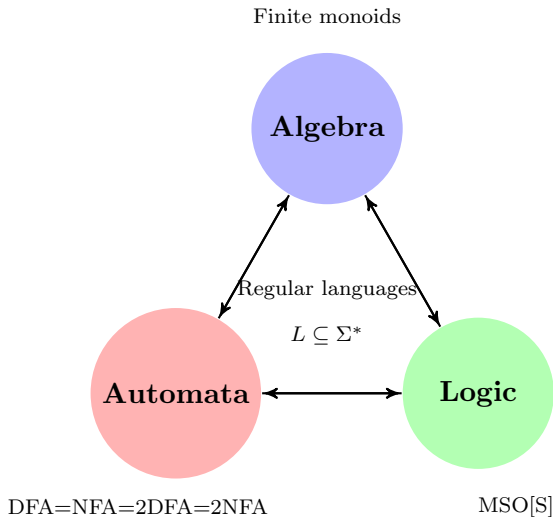
Emmanuel Filiot

Université libre de Bruxelles & FNRS

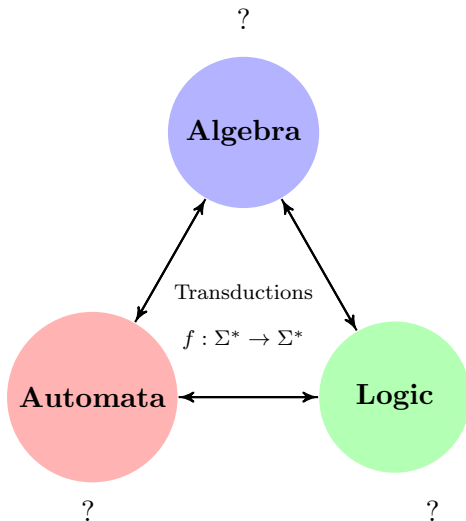
Trinity for Regular Languages



Trinity for Regular Languages

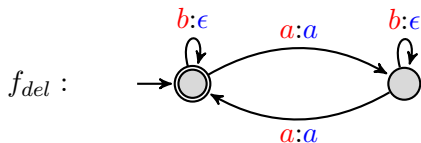


Objective of the talk

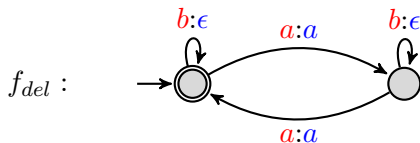


Automata models for transductions

Automata for transductions: transducers

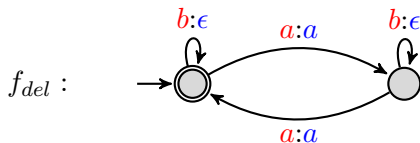


Automata for transductions: transducers



$aabaa \mapsto aaaa$

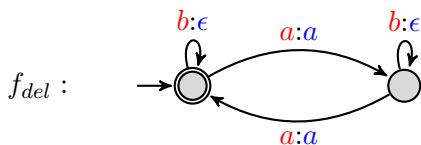
Automata for transductions: transducers



$aabaa \mapsto aaaa$

$aaba \mapsto \text{undefined}$

Automata for transductions: transducers



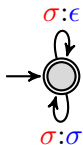
$aabaa \mapsto aaaa$

$aaba \mapsto \text{undefined}$

$\text{dom}(f_{del}) = \text{'even number of } a\text{'}$

Non-determinism

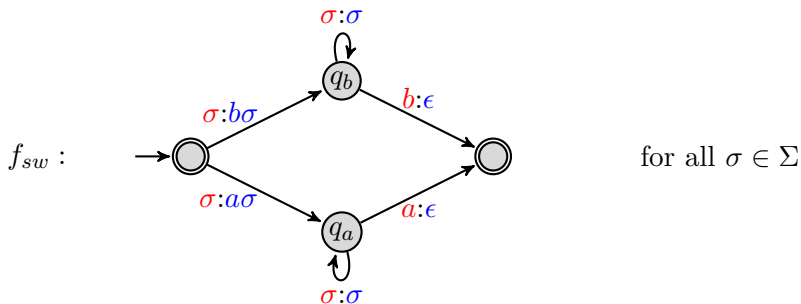
In general, transducers define binary *relations* in $\Sigma^* \times \Sigma^*$



realizes $\{(\textcolor{red}{u}, \textcolor{blue}{v}) \mid \textcolor{blue}{v} \text{ is a subword of } \textcolor{red}{u}\}$

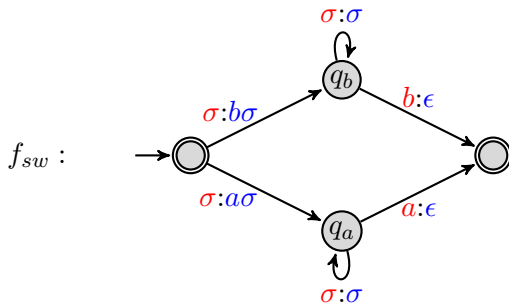
Sequential vs Non-deterministic functional

Non-deterministic transducers may define functions:



Sequential vs Non-deterministic functional

Non-deterministic transducers may define functions:

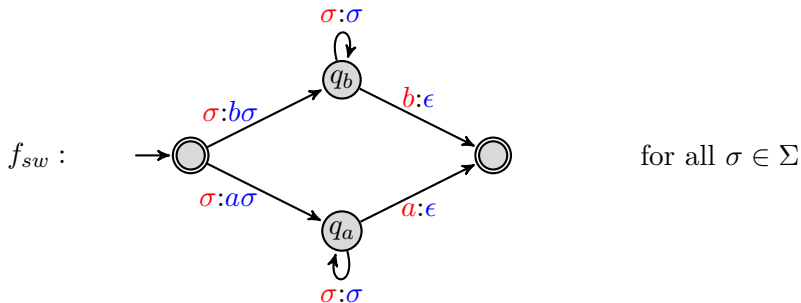


for all $\sigma \in \Sigma$

$babaa \mapsto ababa$

Sequential vs Non-deterministic functional

Non-deterministic transducers may define functions:



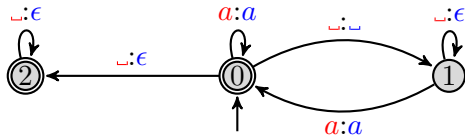
$$babaa \mapsto ababa$$

$$u\sigma \mapsto \sigma u \quad |u| \geq 1$$

input-determinism (aka sequential) < non-determinism \cap functions

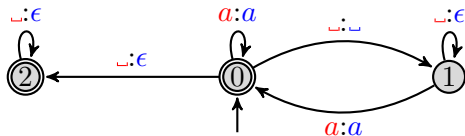
Determinizability

\sqcup = white space



Determinizability

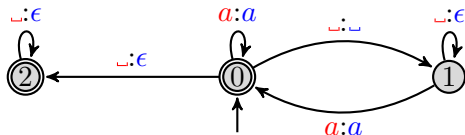
\sqcup = white space



$\sqcup a a \sqcup \sqcup a \sqcup \sqcup \mapsto \sqcup a a \sqcup a$

Determinizability

\sqcup = white space

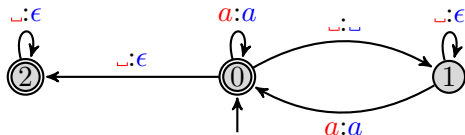


$\sqcup a a \sqcup \sqcup a \sqcup \sqcup \mapsto \sqcup a a \sqcup a$

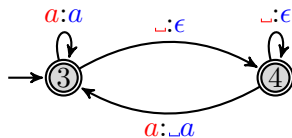
Is non-determinism needed ?

Determinizability

\sqcup = white space



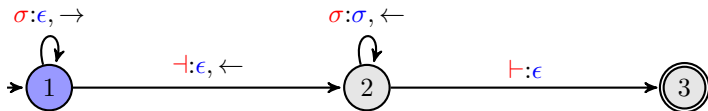
$\sqcup a a \sqcup \sqcup a \sqcup \sqcup \mapsto \sqcup a a \sqcup a$



Is non-determinism needed ? No.

Two-way transducers

input \vdash *s t r e s s e d* \dashv



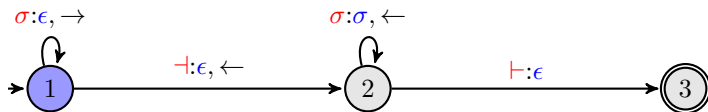
output



Two-way transducers

input $\vdash \textcolor{red}{s} \textcolor{red}{t} \textcolor{red}{r} \textcolor{red}{e} \textcolor{red}{s} \textcolor{red}{s} \textcolor{red}{e} \textcolor{red}{d} \vdash$

▲

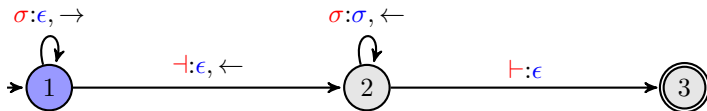


output

▲

Two-way transducers

input \vdash *s* *t* *r* *e* *s* *s* *e* *d* \vdash

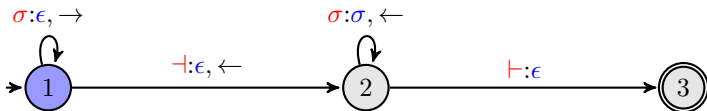


output



Two-way transducers

input $\vdash \textit{ s t r e s s e d } \vdash$

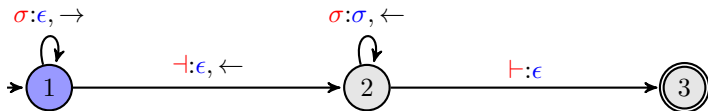


output



Two-way transducers

input $\vdash \textit{ s t r e s s e d } \vdash$

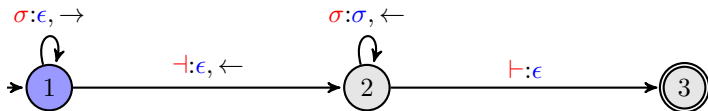


output



Two-way transducers

input $\vdash s t r e s s e d \vdash$

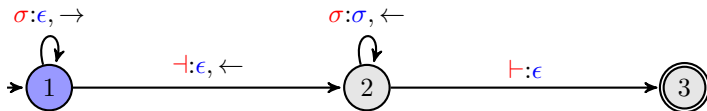


output



Two-way transducers

input $\vdash \textit{ s t r e s s e d } \vdash$



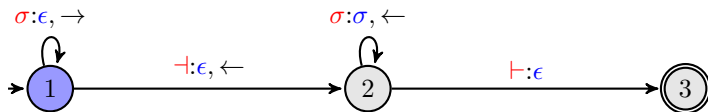
output



Two-way transducers

input \vdash s t r e s s e d \vdash

▲

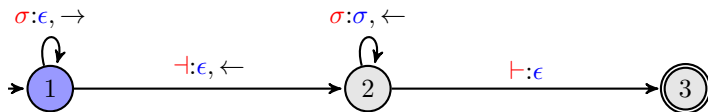


output

▲

Two-way transducers

input \vdash *s* *t* *r* *e* *s* *s* *e* *d* \vdash
 \blacktriangle



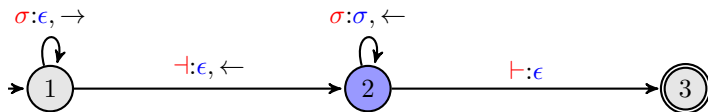
output

\blacktriangle

Two-way transducers

input \vdash s t r e s s e d \vdash

▲



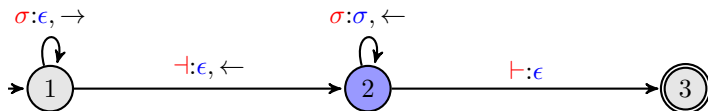
output

▲

Two-way transducers

input \vdash s t r e s s e d \vdash

▲



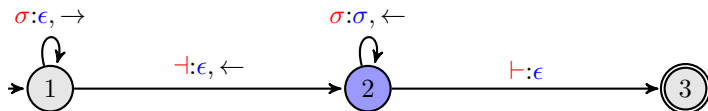
output d

▲

Two-way transducers

input $\vdash \textcolor{red}{s} \textcolor{red}{t} \textcolor{red}{r} \textcolor{red}{e} \textcolor{red}{s} \textcolor{red}{s} \textcolor{red}{e} \textcolor{red}{d} \vdash$

▲



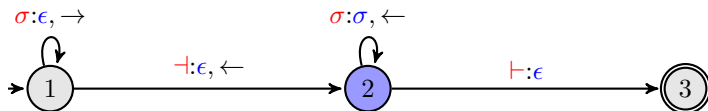
output $\textcolor{blue}{d} \textcolor{blue}{e}$

▲

Two-way transducers

input $\vdash s t r e s s e d \vdash$

▲

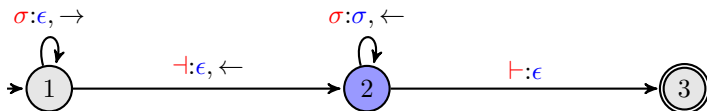


output $d e s$

▲

Two-way transducers

input $\vdash \textcolor{red}{s} \textcolor{red}{t} \textcolor{red}{r} \textcolor{red}{e} \textcolor{red}{s} \textcolor{red}{s} \textcolor{red}{e} \textcolor{red}{d} \vdash$



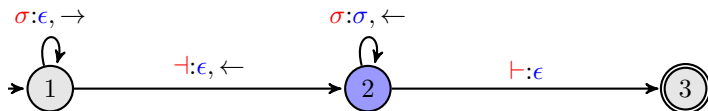
output $\textcolor{blue}{d} \textcolor{blue}{e} \textcolor{blue}{s} \textcolor{blue}{s}$



Two-way transducers

input $\vdash \textcolor{red}{s} \textcolor{red}{t} \textcolor{red}{r} \textcolor{red}{e} \textcolor{red}{s} \textcolor{red}{s} \textcolor{red}{e} \textcolor{red}{d} \vdash$

▲

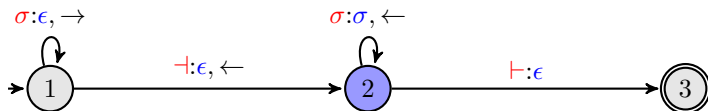


output $\textcolor{blue}{d} \textcolor{blue}{e} \textcolor{blue}{s} \textcolor{blue}{s} \textcolor{blue}{e}$

▲

Two-way transducers

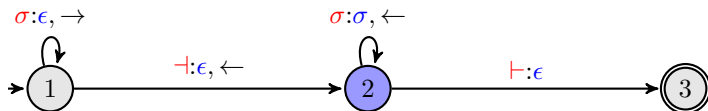
input $\vdash \textcolor{red}{s} \textcolor{red}{t} \textcolor{red}{r} \textcolor{red}{e} \textcolor{red}{s} \textcolor{red}{s} \textcolor{red}{e} \textcolor{red}{d} \vdash$
 \blacktriangle



output $\textcolor{blue}{d} \textcolor{blue}{e} \textcolor{blue}{s} \textcolor{blue}{s} \textcolor{blue}{e} \textcolor{blue}{r}$
 \blacktriangle

Two-way transducers

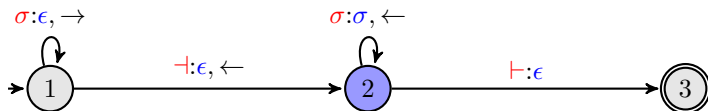
input \vdash s t r e s s e d \vdash
 \blacktriangle



output d e s s e r t
 \blacktriangle

Two-way transducers

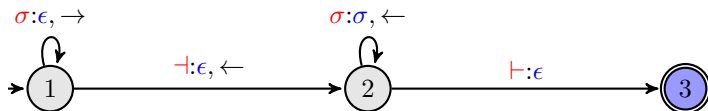
input $\vdash \quad s \quad t \quad r \quad e \quad s \quad s \quad e \quad d \quad \vdash$
 \blacktriangle



output $d \quad e \quad s \quad s \quad e \quad r \quad t \quad s$
 \blacktriangle

Two-way transducers

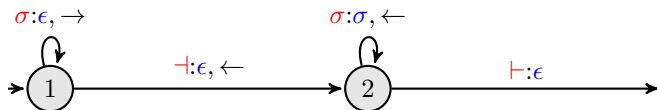
input $\vdash \quad s \quad t \quad r \quad e \quad s \quad s \quad e \quad d \quad \vdash$
 \blacktriangle



output $d \quad e \quad s \quad s \quad e \quad r \quad t \quad s$
 \blacktriangle

Two-way transducers

input s t r e s s e d ⊥

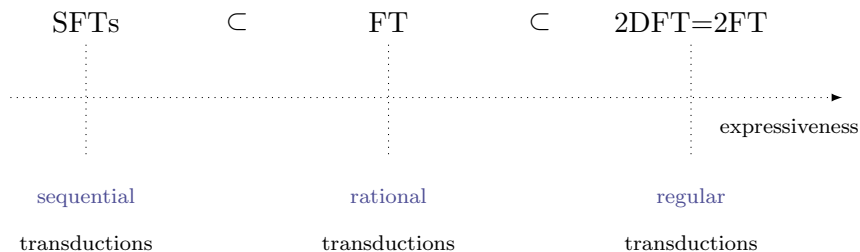


output d e s s e r t s
▲

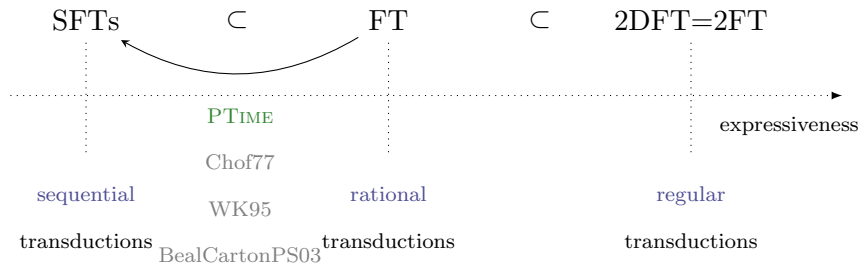
one-way < two-way

- ☺ decidable equivalence problem (Culik, Karhumaki, 87).
- ☺ closed under composition \circ (Chytil, Jakl, 77)

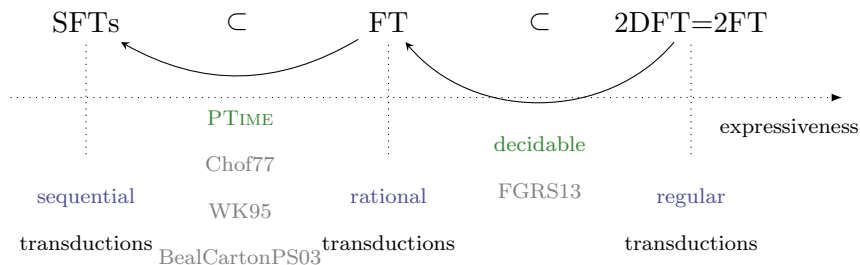
Landscape of Transducer Classes



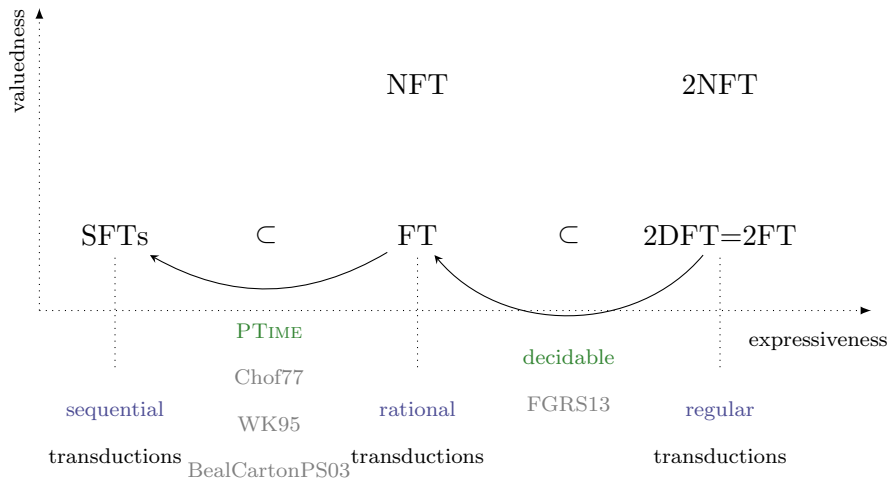
Landscape of Transducer Classes



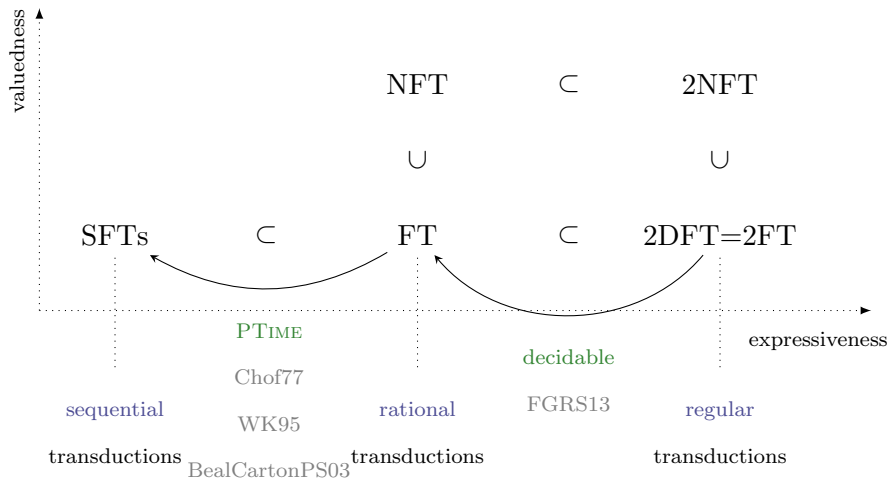
Landscape of Transducer Classes



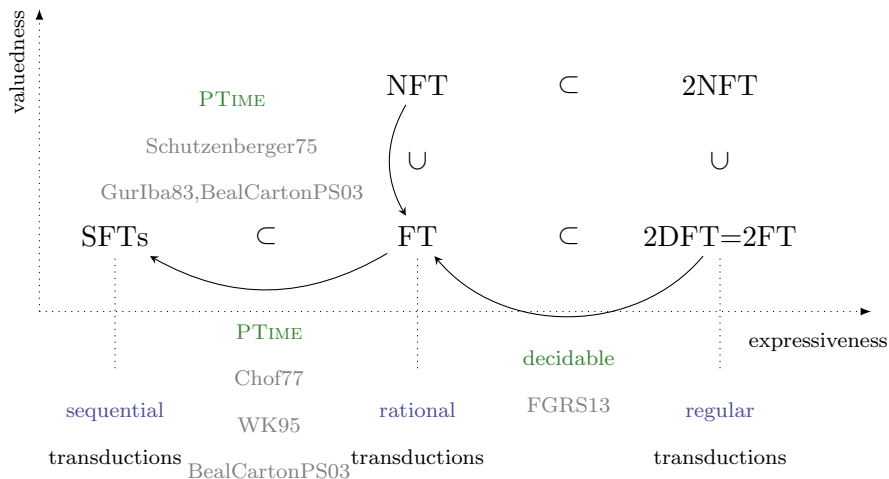
Landscape of Transducer Classes



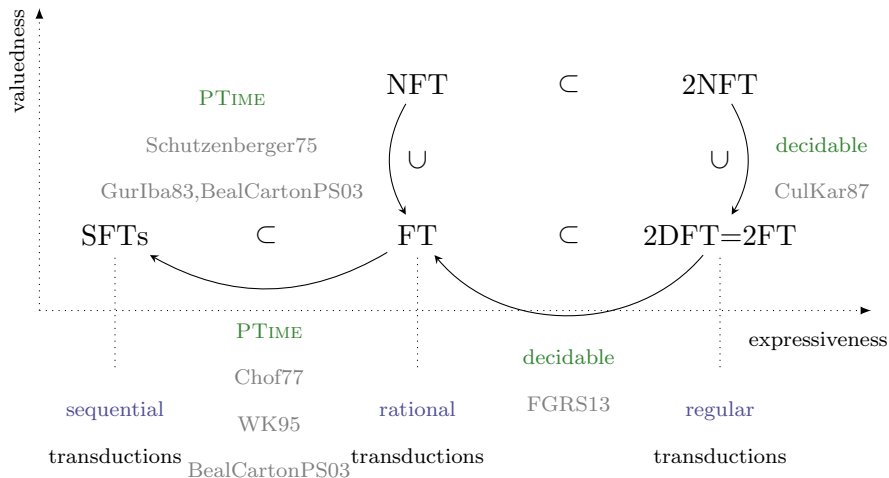
Landscape of Transducer Classes



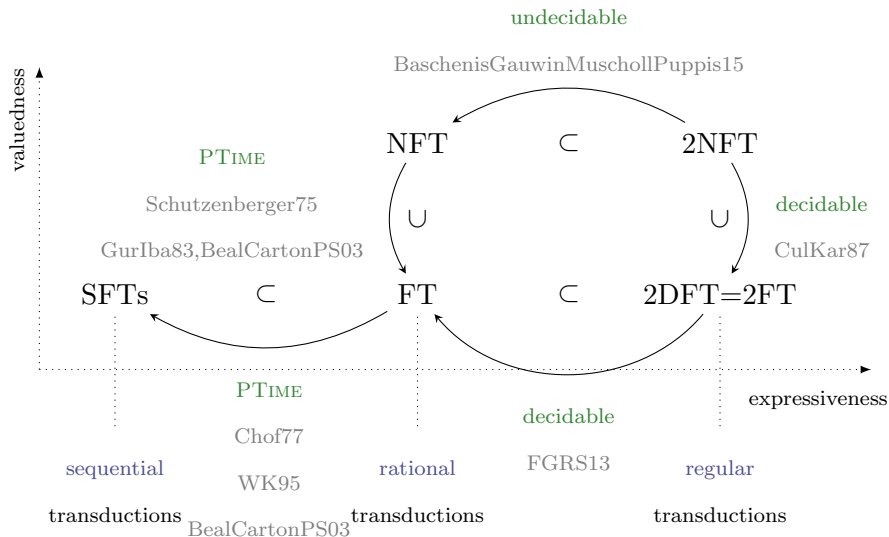
Landscape of Transducer Classes



Landscape of Transducer Classes



Landscape of Transducer Classes



Other recent results

Transducers with registers

- ▶ deterministic one-way
- ▶ equivalent to 2DFT if copyless updates (Alur, Cerny, 10)
- ▶ decidable equivalence problem (F., Reynier) \sim HDT0L

 $\sigma \mid X := \sigma X$


mirror

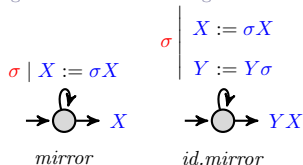
 σ
 $X := \sigma X$
 $Y := Y\sigma$


id.mirror

Other recent results

Transducers with registers

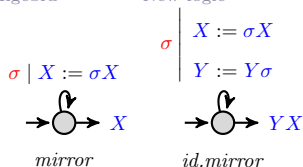
- ▶ deterministic one-way
- ▶ equivalent to 2DFT if copyless updates (Alur, Cerny, 10)
- ▶ decidable equivalence problem (F., Reynier) \sim HDT0L
- ▶ regular expressions to register transducer, implemented in DReX (Alur, D'Antoni, Raghothaman, 2015)
- ▶ register minimization for a subclass (Baschenis, Gauwin, Muscholl, Puppis, 16)



Other recent results

Transducers with registers

- ▶ deterministic one-way
- ▶ equivalent to 2DFT if copyless updates (Alur, Cerny, 10)
- ▶ decidable equivalence problem (F., Reynier) \sim HDT0L
- ▶ regular expressions to register transducer, implemented in DReX (Alur, D'Antoni, Raghothaman, 2015)
- ▶ register minimization for a subclass (Baschenis, Gauwin, Muscholl, Puppis, 16)



Two-way to one-way transducers

- ▶ decidable, but non-elementary complexity in (FGRS13)
- ▶ elementary complexity first obtained for subclasses (sweeping) by (BGMP15)
- ▶ recently for the full class (two talks at Delta)

Logic for transductions

(Courcelle) MSO Transformations

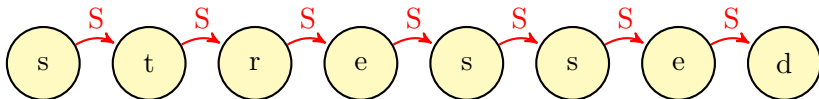
“interpreting the output structure in the input structure”

- ▶ output predicates defined by $\text{MSO}[S]$ formulas interpreted over the input structure

(Courcelle) MSO Transformations

“interpreting the output structure in the input structure”

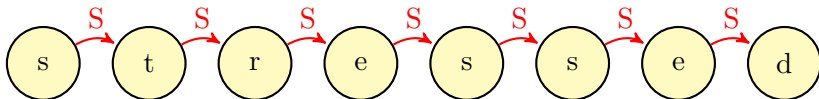
- ▶ output predicates defined by $\text{MSO}[S]$ formulas interpreted over the input structure



(Courcelle) MSO Transformations

“interpreting the output structure in the input structure”

- ▶ output predicates defined by MSO[S] formulas interpreted over the input structure



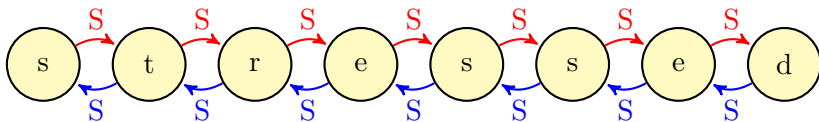
$$\phi_{\textcolor{blue}{S}}(x, y) \equiv \textcolor{red}{S}(y, x)$$

$$\phi_{\textcolor{blue}{\sigma}}(x) \equiv \textcolor{red}{\sigma}(x)$$

(Courcelle) MSO Transformations

“interpreting the output structure in the input structure”

- ▶ output predicates defined by MSO[S] formulas interpreted over the input structure



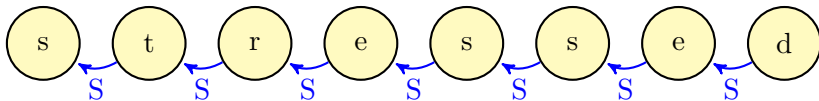
$$\phi_S(x, y) \equiv S(y, x)$$

$$\phi_\sigma(x) \equiv \sigma(x)$$

(Courcelle) MSO Transformations

“interpreting the output structure in the input structure”

- ▶ output predicates defined by MSO[S] formulas interpreted over the input structure



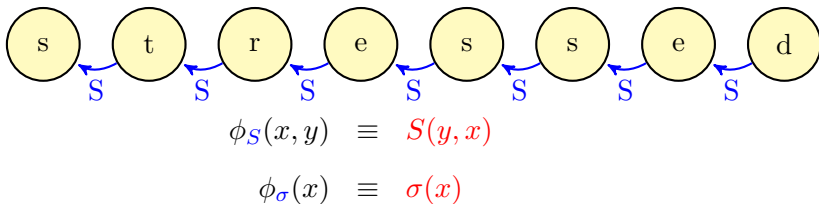
$$\phi_S(x, y) \equiv S(y, x)$$

$$\phi_\sigma(x) \equiv \sigma(x)$$

(Courcelle) MSO Transformations

“interpreting the output structure in the input structure”

- ▶ output predicates defined by MSO[S] formulas interpreted over the input structure



- ▶ input structure can be copied a fixed number of times:
 $u \mapsto uu$, or $u \mapsto u.\text{mirror}(u)$.

Büchi Theorems for Word Transductions

Let $f : \Sigma^* \rightarrow \Sigma^*$.

Theorem (Engelfriet, Hoogeboom, 01)

f is 2FT-definable iff f is MSO-definable.

Büchi Theorems for Word Transductions

Let $f : \Sigma^* \rightarrow \Sigma^*$.

Theorem (Engelfriet, Hoogeboom, 01)

f is 2FT-definable iff f is MSO-definable.

Consequence Equivalence is decidable for MSO-transducers.

Büchi Theorems for Word Transductions

Let $f : \Sigma^* \rightarrow \Sigma^*$.

Theorem (Engelfriet, Hoogeboom, 01)

f is 2FT-definable iff f is MSO-definable.

Consequence Equivalence is decidable for MSO-transducers.

Theorem (Bojanczyk 14, F. 15)

f is (1)FT-definable iff f is order-preserving MSO-definable.

Order-preserving MSO: $\phi_S^{i,j}(x, y) \models x \preceq y$.

First-order transductions

Replace MSO by FO formulas.

Results

- ▶ equivalent to aperiodic transducers with registers (F., Trivedi, Krishna S., 14)
- ▶ and to aperiodic 2DFT (Carton, Dartois, 15) (Dartois, Jecker, Reynier, 16)

Algebraic characterizations of transductions

Myhill-Nerode congruence for

- ▶ $u \sim_L v$ if: for all $w \in \Sigma^*$, $uw \in L$ iff $vw \in L$
- ▶ u and v have the same “effect” on continuations w
- ▶ **Myhill-Nerode’s Thm:** L is regular iff Σ^*/\sim_L is finite
- ▶ canonical (and minimal) deterministic automaton for L , Σ^*/\sim_L as set of states

Myhill-Nerode congruence for

- ▶ $u \sim_L v$ if: for all $w \in \Sigma^*$, $uw \in L$ iff $vw \in L$
- ▶ u and v have the same “effect” on continuations w
- ▶ **Myhill-Nerode’s Thm:** L is regular iff Σ^*/\sim_L is finite
- ▶ canonical (and minimal) deterministic automaton for L , Σ^*/\sim_L as set of states

Goal

Extend Myhill-Nerode’s theorem to classes of transductions

Sequential transductions (Choffrut)

Refinement of the MN congruence.

Two ideas

1. produce asap: $F(\textcolor{red}{u}) = LCP\{\textcolor{blue}{f(uw)} \mid uw \in \text{dom}(f)\}$

Sequential transductions (Choffrut)

Refinement of the MN congruence.

Two ideas

1. produce asap: $F(u) = LCP\{f(uw) \mid uw \in \text{dom}(f)\}$

2. $u \sim_f v$ if

2.1 $u \sim_{\text{dom}(f)} v$

2.2 $F(u)^{-1}f(uw) = F(v)^{-1}f(vw) \quad \forall w \in u^{-1}\text{dom}(f)$

“ u and v have the same effect on continuations w w.r.t. domain membership and produced outputs”

Sequential transductions (Choffrut)

Refinement of the MN congruence.

Two ideas

1. produce asap: $F(\textcolor{red}{u}) = LCP\{f(\textcolor{blue}{uw}) \mid uw \in \text{dom}(f)\}$

2. $\textcolor{red}{u} \sim_f \textcolor{red}{v}$ if

2.1 $\textcolor{red}{u} \sim_{\text{dom}(f)} \textcolor{red}{v}$

2.2 $F(\textcolor{red}{u})^{-1}f(\textcolor{blue}{uw}) = F(\textcolor{red}{v})^{-1}f(\textcolor{blue}{vw}) \quad \forall w \in \textcolor{red}{u}^{-1}\text{dom}(f)$

“ $\textcolor{red}{u}$ and $\textcolor{red}{v}$ have the same effect on continuations $\textcolor{red}{w}$ w.r.t. domain membership and produced outputs”

Theorem (Choffrut)

f is sequential iff \sim_f has finite index

\sim_f is a right congruence \rightsquigarrow canonical and minimal transducer !

Transitions: $[u] \xrightarrow{\textcolor{red}{\sigma} | F(\textcolor{red}{u})^{-1}F(\textcolor{red}{u}\sigma)} [\textcolor{red}{u}\sigma]$

Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead information* $\mathcal{I} = \{a, b, \epsilon\}$.

abb a a a b b b b a b

Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead information* $\mathcal{I} = \{a, b, \epsilon\}$.

$a b b a a a b b b b a b^{\epsilon}$

Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead information* $\mathcal{I} = \{a, b, \epsilon\}$.

$a b b a a a b b b b \overset{b}{a} \overset{\epsilon}{b}$

Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead information* $\mathcal{I} = \{a, b, \epsilon\}$.

$a b b a a a b b b b a b$
 $\quad \quad \quad b \ b \ \epsilon$

Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead information* $\mathcal{I} = \{a, b, \epsilon\}$.

$a b b a a a a b b b b a b$
 $ \overset{b \ b \ b \ \epsilon}{b \ b \ b \ \epsilon}$

Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead information* $\mathcal{I} = \{a, b, \epsilon\}$.

$a b b a a a \overset{b b b b \epsilon}{b b b b a b}$

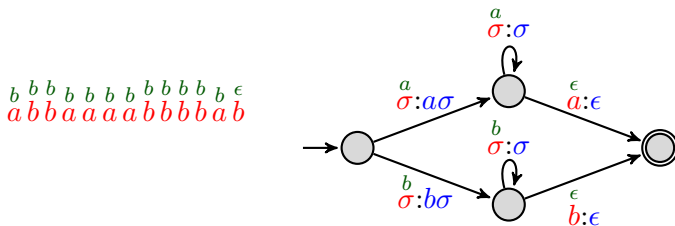
Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead information* $\mathcal{I} = \{a, b, \epsilon\}$.

$\begin{array}{cccccccccccc}
b & b & b & b & b & b & b & b & b & b & b & \epsilon \\
a & b & b & a & a & a & a & b & b & b & a & b
\end{array}$

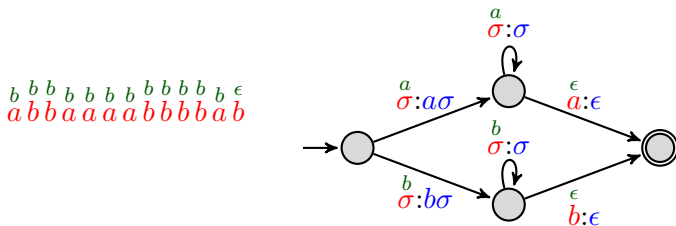
Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead* information $\mathcal{I} = \{a, b, \epsilon\}$.



Rational transductions are almost sequential

- ▶ $f_{sw} : u\sigma \mapsto \sigma u$ is not sequential
- ▶ but sequential modulo *look-ahead information* $\mathcal{I} = \{a, b, \epsilon\}$.



- ▶ look-ahead information: $\mathcal{L} : \Sigma^* \rightarrow \mathcal{I}$
- ▶ $f[\mathcal{L}]$: f with input words extended with look-ahead information

Results

Theorem (Elgot, Mezei, 65)

f is rational iff $f[\mathcal{L}]$ is sequential, for some finite look-ahead information \mathcal{L} computable by a right sequential transducer.

Original statement: $RAT = SEQ \circ RightSEQ$.

Results

Theorem (Elgot, Mezei, 65)

f is rational iff $f[\mathcal{L}]$ is sequential, for some finite look-ahead information \mathcal{L} computable by a right sequential transducer.

Original statement: $RAT = SEQ \circ RightSEQ$.

Reutenauer, Schützenberger, 91

- ▶ *canonical look-ahead* given by a congruence \equiv_f
- ▶ identify suffixes with a 'bounded' effect on the transduction of prefixes
- ▶ characterization of rational transductions
 - ▶ f is rational
 - ▶ \equiv_f has finite index and $f[\equiv_f]$ is sequential
 - ▶ \equiv_f and $\sim_{f[\equiv_f]}$ have finite index.

A canonical look-ahead

Identify suffixes with a 'bounded' effect on prefixes.

Equivalent suffixes

- $u \equiv_f v$ if $\{d(f(wu), f(wv)) \mid wu, wv \in \text{dom}(f)\}$ is finite¹.

¹ $d(\alpha, \beta) = |\alpha| + |\beta| - 2|\text{lcp}(\alpha, \beta)|$.

A canonical look-ahead

Identify suffixes with a 'bounded' effect on prefixes.

Equivalent suffixes

- ▶ $u \equiv_f v$ if $\{d(f(wu), f(wv)) \mid wu, wv \in \text{dom}(f)\}$ is finite¹.
- ▶ \equiv_f is a left congruence

¹ $d(\alpha, \beta) = |\alpha| + |\beta| - 2|\text{lcp}(\alpha, \beta)|$.

A canonical look-ahead

Identify suffixes with a 'bounded' effect on prefixes.

Equivalent suffixes

- ▶ $u \equiv_f v$ if $\{d(f(wu), f(wv)) \mid wu, wv \in \text{dom}(f)\}$ is finite¹.
- ▶ \equiv_f is a left congruence
- ▶ look-ahead: return the class of the suffix.

Ex: For f_{sw} , $\Sigma^*/\equiv = \{\{ua \mid u \in \Sigma^*\}, \{ub \mid u \in \Sigma^*\}, \{\epsilon\}\}$.

¹ $d(\alpha, \beta) = |\alpha| + |\beta| - 2|lcp(\alpha, \beta)|$.

A canonical look-ahead

Identify suffixes with a 'bounded' effect on prefixes.

Equivalent suffixes

- ▶ $u \equiv_f v$ if $\{d(f(wu), f(wv)) \mid wu, wv \in \text{dom}(f)\}$ is finite¹.
- ▶ \equiv_f is a left congruence
- ▶ look-ahead: return the class of the suffix.

Ex: For f_{sw} , $\Sigma^*/\equiv = \{\{ua \mid u \in \Sigma^*\}, \{ub \mid u \in \Sigma^*\}, \{\epsilon\}\}$.

Original Result

- ▶ canonical machine called bimachine
- ▶ bimachine = left DFA + right DFA + output function
- ▶ right automaton = look-ahead

¹ $d(\alpha, \beta) = |\alpha| + |\beta| - 2|lcp(\alpha, \beta)|$.

Definability problems

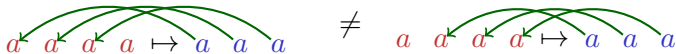
Rational Transductions

Given f defined by T , is it definable by some \mathcal{C} -transducer ?

- ▶ sufficient conditions on \mathcal{C} to get decidability (F., Gauwin, Lhote, LICS'16)
- ▶ includes aperiodic congruences: decidable FO-definability
- ▶ even PSPACE-C (F., Gauwin, Lhote, FSTTCS'16)

Regular Transductions

- ▶ existence of a canonical transducer if *origin* is taken into account (Bojanczyk, ICALP'14)



- ▶ decidable FO-definability *with* origin, open without

A new logic for transductions

joint with Luc Dartois and Nathan Lhote

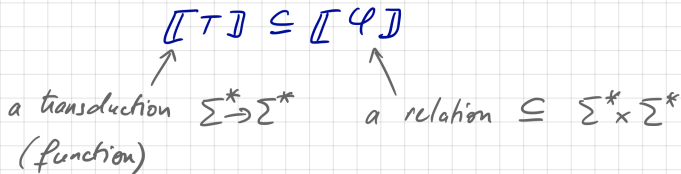
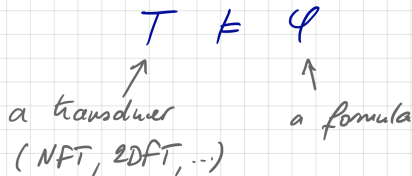
MOTIVATIONS

- specify properties of transductions in a high-level formalism : a logic
- decidable model-checking

$$\begin{array}{ccc} T & \models & \varphi \\ \uparrow & & \uparrow \\ \text{a transducer} & & \text{a formula} \\ (NFT, 2DFT, \dots) & & \end{array}$$

MOTIVATIONS

- specify properties of transductions in a high-level formalism: a logic
- decidable model-checking



EXAMPLES

● "there exists at least one 'a' in the output"

$$\{ (u, v_1 a v_2) \mid u, v_1, v_2 \in \Sigma^* \}$$

● "every request is processed exactly once"

$$\{ (r_1 \dots r_k, g_{\pi(i_1)} \dots g_{\pi(i_k)}) \mid \pi \text{ permutation} \}$$

● more generally : shuffle

$$\{ (\sigma_1 \dots \sigma_n, \sigma_{\pi(1)} \dots \sigma_{\pi(n)}) \mid \pi \text{ permutation} \}$$

NON-DETERMINISTIC MSOT (NMSOT) COURCELLE

use second-order parameters X_1, \dots, X_n

$\{ (u, v) \mid v \text{ is a subword of } u, |v| \text{ even} \}$



$$\varphi_{\text{dom}}(X) \equiv \text{even}(X)$$

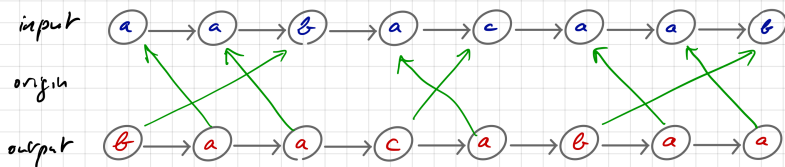
$$\begin{aligned} \varphi_S(x, y, X) = & x, y \in X \wedge x < y \\ & \wedge \neg (\exists z \in X. x < z < y) \end{aligned}$$

FROM NMSOT to NEW LOGIC

NMSOT is not satisfactory as a specification language

- * too "operational"
- * the previous examples are not NMSOT

IDEA: SEE TRANSDUCTIONS AS SINGLE STRUCTURES WITH ORIGIN



Predicates: \leq_{in} , \leq_{out} , σ

EXAMPLES

- "there exists at least one 'a' in the output"

$$\exists^{out}_x . a(x)$$

- shuffle

$$BIT(\sigma) \wedge \underbrace{\bigwedge_{\sigma \in \Sigma} \forall^{in}_x \forall^{out}_y . \frac{\sigma(x,y) \wedge}{\sigma(x)} \rightarrow \sigma(y)}_{LABPRES(\sigma)}$$

- identity

$$BIT(\sigma) \wedge LABPRES(\sigma) \wedge ORDERPRES(\sigma)$$

where $ORDERPRES(\sigma) \equiv \forall^{in}_{x,x'} \forall^{out}_{y,y'} \frac{\sigma(x,y) \wedge \sigma(x',y')}{x \leq_{in} x'} \rightarrow y \leq_{out} y'$

RESULTS ON $\text{MSO}[\leq_{\text{in}}, \leq_{\text{out}}, \emptyset]$

Decidable model-checking of 2DFT [Bojanczyk'17]
et. al.

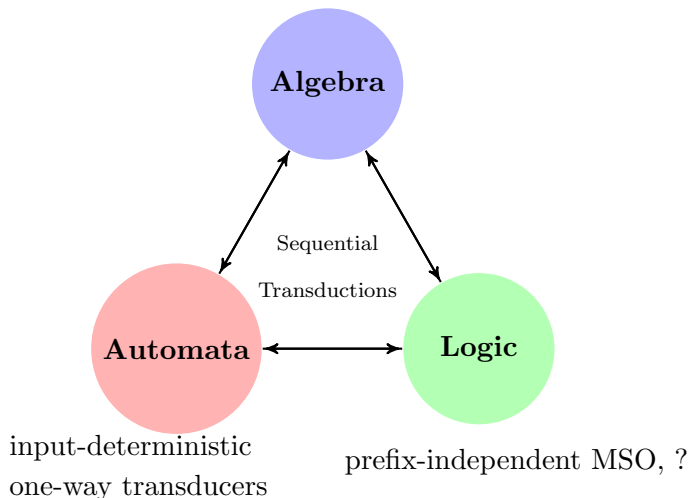
Undecidable satisfiability

Decidable fragment for satisfiability

} [Nathan's talk]

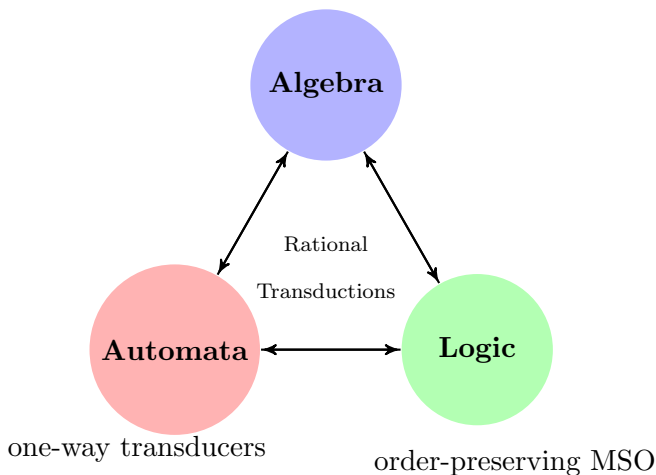
Summary: sequential transductions

finiteness of \sim_f (Choffrut)

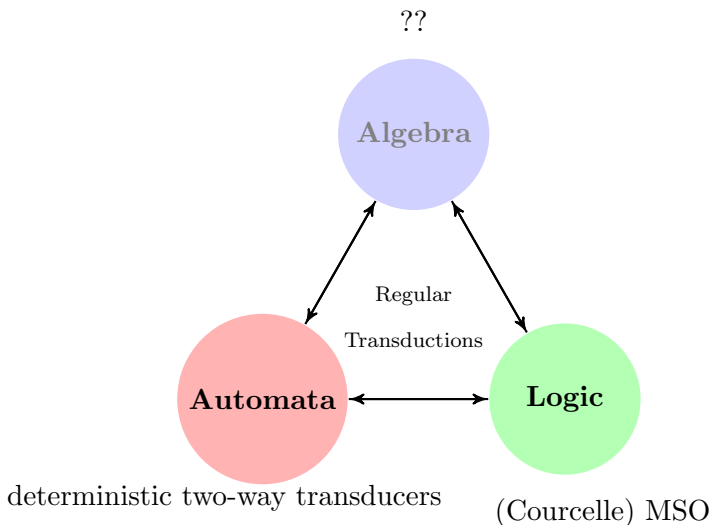


Summary: rational transductions

finiteness of \equiv_f and $\sim_{f[\equiv_f]}$ (Reutenauer, Schützenberger)



Summary: regular transductions



Other works

- ▶ AC^0 transductions (Cadilhac, Krebs, Ludwig, Paperman, 15)
- ▶ variants of two-way transducers (Guillon, Choffrut, 14, 15, 16), (Carton, 12) (McKenzie, Schwentick, Thérien, Vollmer, 06)
- ▶ learning rational transductions (Boiret, Lemay, Niehren, 12)
- ▶ inclusion, equivalence and uniformisation problems for rational transductions with “similar origins” (F., Jecker, Löding, Winter, 16)
- ▶ non-determinism
- ▶ infinite words, nested words, trees

SIGLOG News 9th



Transducers, Logic and Algebra for Functions of Finite Words

Emmanuel Filiot, Université Libre de Bruxelles
and Pierre-Alain Reynier, Aix-Marseille Université

The robust theory of regular languages is based on three important pillars: computation (automata) and algebra. In this paper, we survey old and recent results on extensions of these pillars to functions of finite words. We consider two important classes of word functions, the rational and regular functions, respectively defined by one-way and two-way automata with output words, called transducers.

ON the relation between computation, mathematical logic, and algebra of finite words. The class of regular languages is captured by finite automata, to the class of languages accepted by pushdown automata [Büchi 1960; Elgot 1968]. While automata theory has been a central pillar of the theory of computation, mathematical logic, and algebra have been less prominent. While automata theory has been a central pillar of the theory of computation, mathematical logic, and algebra have been less prominent. While automata theory has been a central pillar of the theory of computation, mathematical logic, and algebra have been less prominent.

SIGLOG News 9th



Thank You.

Transducers, Logic and Algebra for Functions of Finite Words

Emmanuel Filiot, Université Libre de Bruxelles
and Pierre-Alain Reynier, Aix-Marseille Université

The robust theory of regular languages is based on three important pillars: computation (automata) and algebra. In this paper, we survey old and recent results on extensions of these pillars to functions of finite words. We consider two important classes of word functions, the rational and regular functions, respectively defined by one-way and two-way automata with output words, called transducers.

ON the relation between computation, mathematical logic, and algebra of finite words. The class of regular languages is the class of languages accepted by finite automata, to the class of languages accepted by transducers [Büchi 1960; Elgot 1961]. While automata theory is a well-established theory, the theory of transducers is less developed. While automata theory is a well-established theory, the theory of transducers is less developed.