Transductions

Emmanuel Filiot[a] and Pierre-Alain Reynier[b]

[a] Université libre de Bruxelles &FNRS
[b] LIS, Aix-Marseille Université & CNRS

EJCIM School

Specification

Describe properties

Computation

Decide those properties

Specification

Describe properties

Computation

Decide those properties

Regular languages

$L \subseteq \Sigma^*$

Logic

Automata

MSO[S] $\longleftrightarrow$ FA

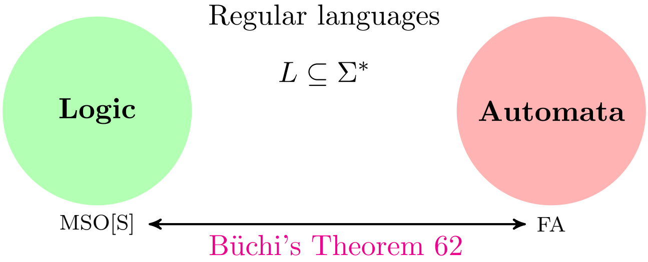
Büchi's Theorem 62

Specification

Describe properties

Computation

Decide those properties



Many extensions: infinite words, finite and infinite trees, graphs, other logics ...

Specification

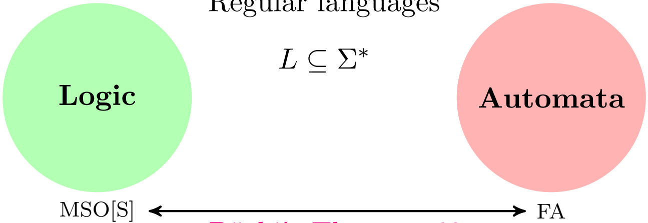
Describe properties

Computation

Decide those properties

Regular languages

$L \subseteq \Sigma^*$



Büchi's Theorem 62

Many extensions: infinite words, finite and infinite trees, graphs, other logics ...

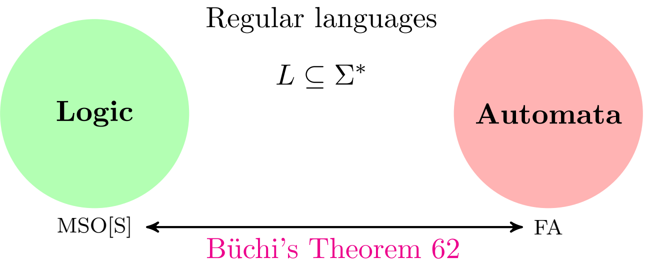
Famous application: Model-checking $A \models \phi$?

Specification

Describe properties

Computation

Decide those properties



Many extensions: infinite words, finite and infinite trees, graphs, other logics ...
Famous application: Model-checking $A \models \phi$?

$$L(A) \cap L(A_{\neg\phi}) = \varnothing$$

Objective of the talk

What about transductions ?

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Objective of the talk

What about transductions ?

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a # $\quad$ $abbab \mapsto abbab\#$

Objective of the talk

What about transductions ?

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$

Objective of the talk

What about transductions ?

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$
Squeeze all white space sequences ≥ 2	$ejcim\text{␣␣␣}19 \mapsto ejcim\text{␣}19$

Objective of the talk

What about transductions ?

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$
Squeeze all white space sequences ≥ 2	$ejcim\text{␣␣␣}19 \mapsto ejcim\text{␣}19$
Add a parity bit	$0100101 \mapsto \mathbf{1}0100101$

Objective of the talk

What about transductions ?

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$
Squeeze all white space sequences ≥ 2	$ejcim\textvisiblespace\textvisiblespace\textvisiblespace 19 \mapsto ejcim\textvisiblespace 19$
Add a parity bit	$0100101 \mapsto \mathbf{1}0100101$
Mirror the input word	$ejcim \mapsto micje$

Objective of the talk

What about transductions ?

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$
Squeeze all white space sequences ≥ 2	$ejcim\text{␣␣␣}19 \mapsto ejcim\text{␣}19$
Add a parity bit	$0100101 \mapsto \mathbf{1}0100101$
Mirror the input word	$ejcim \mapsto micje$
Copy the input word	$yes \mapsto yesyes$

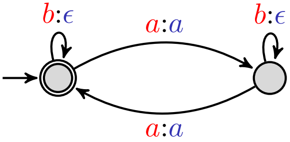
Outline

1. automata for transductions
2. closure properties and decision problems
3. logics for transductions
4. a more expressive class of functions
5. recent results

Automata models for transductions

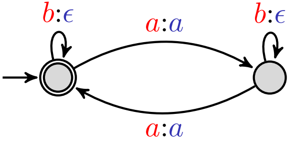
Automata for transductions: transducers

f_{del} :



Automata for transductions: transducers

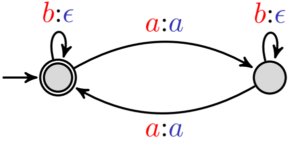
f_{del} :



$$aabaa \mapsto aaaa$$

Automata for transductions: transducers

f_{del} :

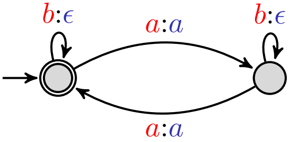


$$aabaa \mapsto aaaa$$

$$aaba \mapsto \text{undefined}$$

Automata for transductions: transducers

f_{del} :



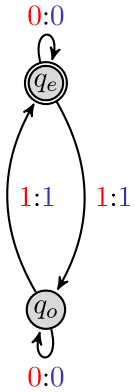
$$aabaa \mapsto aaaa$$

$$aaba \mapsto \text{undefined}$$

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

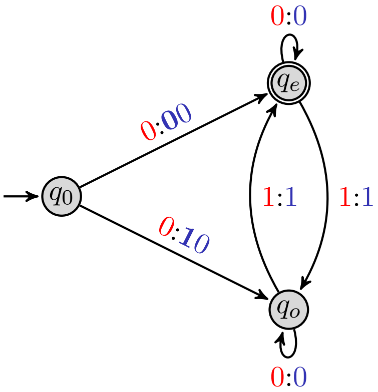
Parity bit

$01101 \mapsto \mathbf{1}01101, 01111 \mapsto \mathbf{0}01111$



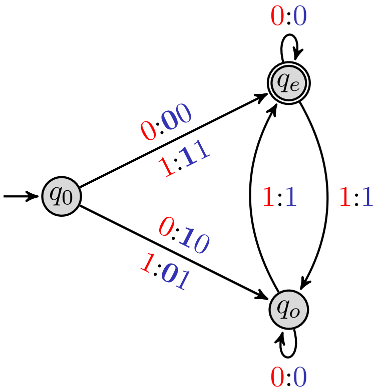
Parity bit

$01101 \mapsto \mathbf{1}01101, 01111 \mapsto \mathbf{0}01111$



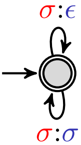
Parity bit

$01101 \mapsto \mathbf{1}01101, 01111 \mapsto \mathbf{0}01111$



Non-determinism and relations

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



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

Formal Definition

Definition

A transducer is a tuple $T = (\Sigma, Q, I, F, \Delta)$ where:

- Σ is a finite alphabet
- Q is a finite set of states
- $I \subseteq Q$ are the initial states and $F \subseteq Q$ are the final states
- $\Delta \subseteq Q \times \Sigma \times \Sigma^* \times Q$ is the transition relation.

Formal Definition

Definition

A transducer is a tuple $T = (\Sigma, Q, I, F, \Delta)$ where:

- Σ is a finite alphabet
- Q is a finite set of states
- $I \subseteq Q$ are the initial states and $F \subseteq Q$ are the final states
- $\Delta \subseteq Q \times \Sigma \times \Sigma^* \times Q$ is the transition relation.

Semantics

A run is a sequence of transitions

$$r = q_0 \xrightarrow{\sigma_1 : v_1} q_1 \dots q_{n-1} \xrightarrow{\sigma_n : v_n} q_n \qquad \sigma_i \in \Sigma$$

Its input is $in(r) = \sigma_1 \dots \sigma_n$ and its output $out(r) = v_1 \dots v_n$.
The (rational) relation defined by T is:

$$[\![T]\!] = \{(in(r), out(r)) \mid r \text{ is an accepting run}\}$$

Closure Properties: Domain and Co-Domain

Given $R \subseteq \Sigma^* \times \Sigma^*$:

- $dom(R) = \{u \mid \exists(u, v) \in R\}$
- $codom(R) = \{v \mid \exists(u, v) \in R\}$

Closure Properties: Domain and Co-Domain

Given $R \subseteq \Sigma^* \times \Sigma^*$:

- $dom(R) = \{u \mid \exists (u, v) \in R\}$
- $codom(R) = \{v \mid \exists (u, v) \in R\}$

Proposition

The domain and co-domain of a rational relation are regular.

Closure Properties: Union

Proposition

Rational relations are closed under union.

Closure Properties: Intersection

1. show that $\{(a^n b^m, a^n) \mid n, m \geq 0\}$ is rational.

Closure Properties: Intersection

1. show that $\{(a^n b^m, a^n) \mid n, m \geq 0\}$ is rational.

2. show that $\{(a^n b^m, a^m) \mid n, m \geq 0\}$ is rational.

Closure Properties: Intersection

1. show that $\{(a^n b^m, a^n) \mid n, m \geq 0\}$ is rational.

2. show that $\{(a^n b^m, a^m) \mid n, m \geq 0\}$ is rational.

3. are rational relations closed under intersection ? why ?

Closure Properties: Intersection

Proposition

Rational relations are not closed under intersection.

Closure Properties: Intersection

Proposition

Rational relations are not closed under intersection.

What about complement ?

Closure Properties: Intersection

Proposition

Rational relations are not closed under intersection.

What about complement ?

Proposition

Rational relations are not closed under complement.

Closure Properties: Composition

Def: $R_2 \circ R_1 = \{(u, w) \mid \exists (u, v) \in R_1, (v, w) \in R_2\}.$

Closure Properties: Composition

Def: $R_2 \circ R_1 = \{(u,w) \mid \exists (u,v) \in R_1, (v,w) \in R_2\}$.

Proposition

Rational relations are closed under composition.

Proof

Let $T_1 = (\Sigma, Q_1, I_1, F_1, \Delta_1)$ and $T_2 = (\Sigma, Q_2, I_2, F_2, \Delta_2)$.

- For all $u \in \Sigma^*$ and $p_2 \in Q_2$, let

$$\mathrm{Prod}_2(u, p_2) = \{(v, q_2) \in \Sigma^* \times Q_2 \mid p_2 \xrightarrow{u|v}_{T_2} q_2\}$$

Proof

Let $T_1 = (\Sigma, Q_1, I_1, F_1, \Delta_1)$ and $T_2 = (\Sigma, Q_2, I_2, F_2, \Delta_2)$.

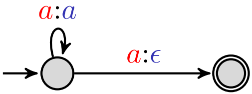
- For all $u \in \Sigma^*$ and $p_2 \in Q_2$, let

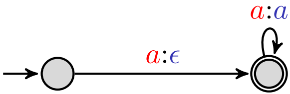
$$\mathrm{Prod}_2(u, p_2) = \{(v, q_2) \in \Sigma^* \times Q_2 \mid p_2 \xrightarrow{u|v}_{T_2} q_2\}$$

- The composition $[\![T_2]\!] \circ [\![T_1]\!]$ is realised by the transducer $T = (\Sigma, Q_1 \times Q_2, I_1 \times I_2, F_1 \times F_2, \Delta)$ where:

$$\Delta = \{(p_1, p_2) \xrightarrow{\sigma|v} (q_1, q_2) \mid \exists p_1 \xrightarrow{\sigma|u}_{T_1} q_1 \wedge (v, q_2) \in \mathrm{Prod}_2(u, p_2)\}$$

Transducer vs Automata

$T:$ 



Transducer vs Automata

$T :$

- Consider r_1, r_2 two runs on a^3. We have $(in(r_1), out(r_1)) = (in(r_2), out(r_2))$ but different in-out words:

$$(a, a)(a, a)(a, \epsilon) \neq (a, \epsilon)(a, a)(a, a)$$

Transducer vs Automata

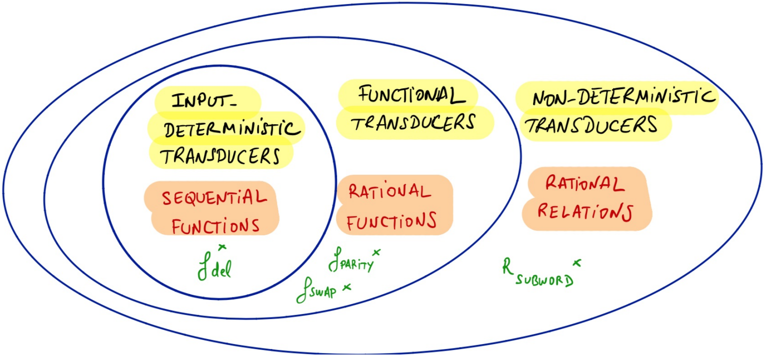
T :

- Consider r_1, r_2 two runs on a^3. We have $(in(r_1), out(r_1)) = (in(r_2), out(r_2))$ but different in-out words:

$$(a, a)(a, a)(a, \epsilon) \neq (a, \epsilon)(a, a)(a, a)$$

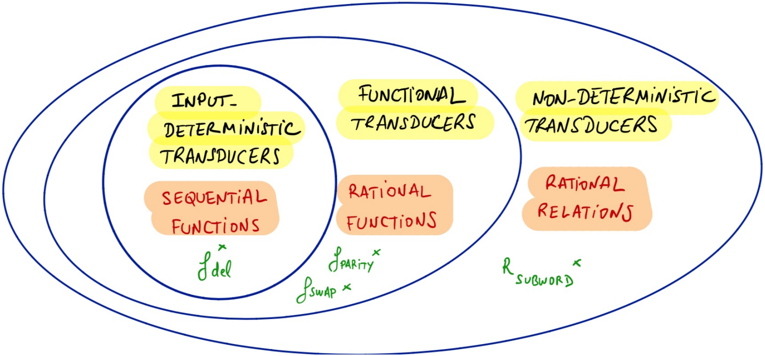
- Transducers are **asynchronous**
- Make most transducer problems conceptually difficult (and even computationally).

Different classes of transductions



$$f_{\mathrm{SWAP}} \;:\; u\sigma \to \sigma u \qquad \sigma \in \{a, b\}, u \in \{a, b\}^*$$

Different classes of transductions



Are the classes of sequential and rational functions decidable ?

Class Membership Problems (for transductions)

Given T a non-deterministic transducer:

Functionality decide if $[\![T]\!]$ is a function,

Determinizability decide if T is equivalent to some input-deterministic transducer.

Some application of the functionality problem

Testing unambiguity of NFA.

Another Fundamental Problem: Equivalence

Def Given two transducers T_1, T_2, does $[\![T_1]\!] = [\![T_2]\!]$ hold?

Case of functional transducers

Equivalence reduces to functionality:

1. test whether $dom(T_1) = dom(T_2)$
2. test whether $T_1 \uplus T_2$ is functional.

Functionality problem: Results

Lem (Schützenberger) Non-functionality is witnessed by runs r_1, r_2 such that

(1) r_1, r_2 are over the same input
(2) r_1, r_2 produce different outputs
(3) r_1, r_2 have polynomial length

Functionality problem: Results

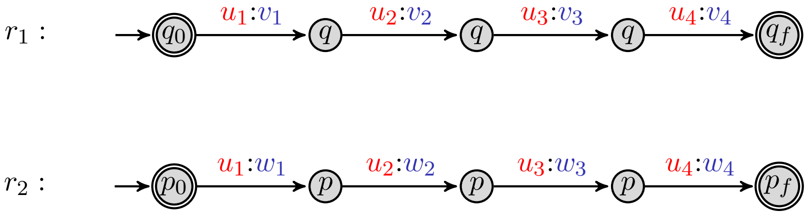
Lem (Schützenberger) Non-functionality is witnessed by runs r_1, r_2 such that

(1) r_1, r_2 are over the same input
(2) r_1, r_2 produce different outputs
(3) r_1, r_2 have polynomial length

Coro Functionality is decidable in PSPACE.

Proof of the Lemma

Assume $(u,v),(u,w) \in R$ where $v \neq w$, given by runs r_1, r_2 resp. If u is long enough:



Proof of the Lemma

Assume $(u, v), (u, w) \in R$ where $v \neq w$, given by runs r_1, r_2 resp. If u is long enough:

$$r_1 : \quad \to q_0 \xrightarrow{u_1 : v_1} q \xrightarrow{u_2 : v_2} q \xrightarrow{u_3 : v_3} q \xrightarrow{u_4 : v_4} q_f$$

$$r_2 : \quad \to p_0 \xrightarrow{u_1 : w_1} p \xrightarrow{u_2 : w_2} p \xrightarrow{u_3 : w_3} p \xrightarrow{u_4 : w_4} p_f$$

Show $v_1 v_2 v_3 v_4 \neq w_1 w_2 w_3 w_4 \implies$

$$v_1 v_4 \neq w_1 w_4 \vee v_1 v_2 v_4 \neq w_1 w_2 w_4 \vee v_1 v_3 v_4 \neq w_1 w_3 w_4$$

Proof (Ced)

$$(v_1v_4 = w_1w_4 \wedge v_1v_2v_4 = w_1w_2w_4 \wedge v_1v_3v_4 = w_1w_3w_4) \implies v_1v_2v_3v_4 = w_1w_2w_3w_4$$

1. Wlog, assume that $v_1 = \epsilon$. If not, assume v_1 prefix of w_1, i.e. $w_1 = v_1w_1'$ and eliminate v_1 from all rhs (the case w_1 prefix of v_1 is symmetric). So, we want

   $$(v_4 = w_1w_4 \wedge v_2v_4 = w_1w_2w_4 \wedge v_3v_4 = w_1w_3w_4) \implies v_2v_3v_4 = w_1w_2w_3w_4$$

Proof (Ced)

$$(v_1v_4 = w_1w_4 \wedge v_1v_2v_4 = w_1w_2w_4 \wedge v_1v_3v_4 = w_1w_3w_4) \implies v_1v_2v_3v_4 = w_1w_2w_3w_4$$

1. Wlog, assume that $v_1 = \epsilon$. If not, assume v_1 prefix of w_1, i.e. $w_1 = v_1w_1'$ and eliminate v_1 from all rhs (the case w_1 prefix of v_1 is symmetric). So, we want

   $$(v_4 = w_1w_4 \wedge v_2v_4 = w_1w_2w_4 \wedge v_3v_4 = w_1w_3w_4) \implies v_2v_3v_4 = w_1w_2w_3w_4$$

2. In $v_2v_4 = w_1w_2w_4$, replace v_4 by w_1w_4, then we get $v_2w_1w_4 = w_1w_2w_4$. Similarly, one gets $v_3w_1w_4 = w_1w_3w_4$. Simplify by w_4 and we get:

   $$v_2w_1 = w_1w_2 \quad (1) \qquad v_3w_1 = w_1w_3 \quad (2)$$

Proof (Ced)

$$(v_1v_4 = w_1w_4 \wedge v_1v_2v_4 = w_1w_2w_4 \wedge v_1v_3v_4 = w_1w_3w_4) \implies v_1v_2v_3v_4 = w_1w_2w_3w_4$$

1. Wlog, assume that $v_1 = \epsilon$. If not, assume v_1 prefix of w_1, i.e. $w_1 = v_1w_1'$ and eliminate v_1 from all rhs (the case w_1 prefix of v_1 is symmetric). So, we want

   $$(v_4 = w_1w_4 \wedge v_2v_4 = w_1w_2w_4 \wedge v_3v_4 = w_1w_3w_4) \implies v_2v_3v_4 = w_1w_2w_3w_4$$

2. In $v_2v_4 = w_1w_2w_4$, replace v_4 by w_1w_4, then we get $v_2w_1w_4 = w_1w_2w_4$. Similarly, one gets $v_3w_1w_4 = w_1w_3w_4$. Simplify by w_4 and we get:

   $$v_2w_1 = w_1w_2 \quad (1) \qquad v_3w_1 = w_1w_3 \quad (2)$$

3. Finally:

   $$\begin{aligned} v_2v_3v_4 &= v_2v_3w_1w_4 && \text{by } v_4 = w_1w_4 \\ &= v_2w_1w_3w_4 && \text{by } (2) \\ &= w_1w_2w_3w_4 && \text{by } (1) \end{aligned}$$

$\square$

Functionality Problem in PTIME

Thm (Gurari,Ibarra,83). Functionality is decidable in PTIME.

- reversal-bounded counter machines
- emptiness in PTIME if fixed number of counters
- later shown with a direct proof by Carton,Beal,Prieur,Sakarovitch (Squaring Transducers)

Squaring Transducers Carton,Beal,Prieur,Sakarovitch, 01

Some Definitions

Given a transducer $T = (\Sigma, Q, I, F, \Delta)$,

- $q \in Q$ is co-accessible by u if $q \xrightarrow{u|v} q_f \in F$ for some v
- $\text{CoAcc} = \{(p, q) \in Q^2 \mid p, q \text{ co-accessible by some } u\}$

Squaring Transducers Carton,Beal,Prieur,Sakarovitch, 01

Some Definitions

Given a transducer $T = (\Sigma, Q, I, F, \Delta)$,

- $q \in Q$ is co-accessible by u if $q \xrightarrow{u|v} q_f \in F$ for some v
- $\text{CoAcc} = \{(p,q) \in Q^2 \mid p, q \text{ co-accessible by some } u\}$
- Let $u, v \in \Sigma^*$, $u \wedge v$ is the longest common prefix of u and v

Squaring Transducers Carton,Beal,Prieur,Sakarovitch, 01

Some Definitions

Given a transducer $T = (\Sigma, Q, I, F, \Delta)$,

- $q \in Q$ is co-accessible by u if $q \xrightarrow{u|v} q_f \in F$ for some v
- $\mathrm{CoAcc} = \{(p,q) \in Q^2 \mid p, q \text{ co-accessible by some } u\}$
- Let $u, v \in \Sigma^*$, $u \wedge v$ is the longest common prefix of u and v
- $\mathsf{delay}(u,v) = (u', v')$ where $u = (u \wedge v)u'$ and $v = (u \wedge v)v'$

Squaring Transducers Carton,Beal,Prieur,Sakarovitch, 01

Some Definitions

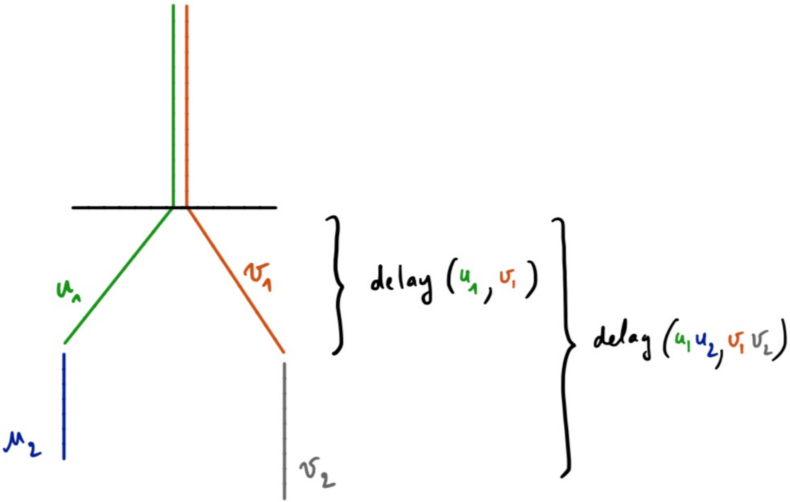
Given a transducer $T = (\Sigma, Q, I, F, \Delta)$,

- $q \in Q$ is <u>co-accessible</u> by u if $q \xrightarrow{u|v} q_f \in F$ for some v
- $\text{CoAcc} = \{(p,q) \in Q^2 \mid p, q \text{ co-accessible by some } u\}$
- Let $u, v \in \Sigma^*$, $u \wedge v$ is the longest common prefix of u and v
- $\mathsf{delay}(u,v) = (u',v')$ where $u = (u \wedge v)u'$ and $v = (u \wedge v)v'$

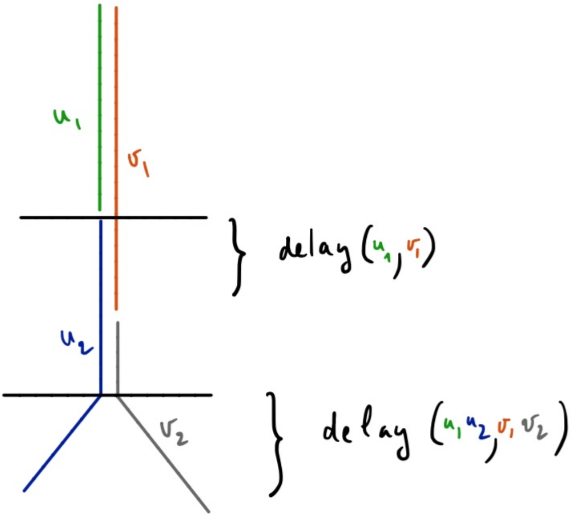
Lemma (The delay is compositional)

$\mathsf{delay}(u_1u_2, v_1v_2) = \mathsf{delay}(\mathsf{delay}(u_1,u_2).(v_1,v_2))$.

The Delay is Compositional ("Proof")



The Delay is Compositional ("Proof")



Squaring Transducers

State delays

$$\mathsf{delays}(p,q) = \{\mathsf{delay}(v,w) \mid \exists u \in \Sigma^* \exists p_0 \xrightarrow{u|v} p \exists q_0 \xrightarrow{u|w} q\}$$

Squaring Transducers

State delays

$$\mathsf{delays}(p,q) = \{\mathsf{delay}(v,w) \mid \exists u \in \Sigma^* \exists p_0 \xrightarrow{u|v} p \exists q_0 \xrightarrow{u|w} q\}$$

First observation

T is functional iff for all states $(p_f, q_f) \in F^2 \cap \mathrm{Acc}$, $\mathsf{delays}(p_f, q_f) = \{(\epsilon, \epsilon)\}$

Squaring Transducers

State delays

$$\mathsf{delays}(p,q) = \{\mathsf{delay}(v,w) \mid \exists u \in \Sigma^* \exists p_0 \xrightarrow{u|v} p \exists q_0 \xrightarrow{u|w} q\}$$

First observation

T is functional iff for all states $(p_f, q_f) \in F^2 \cap \mathrm{Acc}$, $\mathsf{delays}(p_f, q_f) = \{(\epsilon, \epsilon)\}$

Second observation

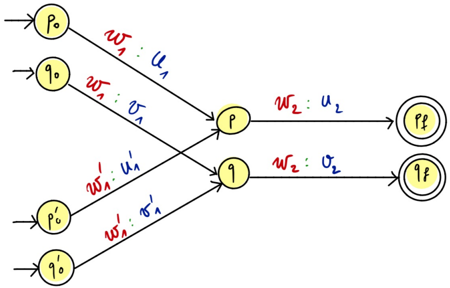
Let $(p,q) \in \mathrm{CoAcc}$. If $|\mathsf{delays}(p,q)| > 1$ then T is not functional.

Proof of the Second Observation

Second observation

Let $(p, q) \in \text{CoAcc}$. If $|\mathsf{delays}(p, q)| > 1$ then T is not functional.

Proof of the second observation



Assume

$$(\alpha_1, \beta_1) := \mathsf{delay}(u_1, v_1) \neq (\alpha_1', \beta_1') := \mathsf{delay}(u_1', v_1').$$

By contradiction, assume that $u_1 u_2 = v_1 v_2$ and $u_1' u_2 = v_1' v_2$. Then $\alpha_1 u_2 = \beta_1 v_2$ and $\alpha_1' u_2 = \beta_1' v_2$. As (α_1, β_1) and (α_1', β_1') are delays, the following cases may arise:

1. α_1 and β_1 start with distinct letters. Impossible.
2. α_1' and β_1' start with distinct letters. Impossible.
3. $\alpha_1 = \alpha_1' = \epsilon$: then $u_2 = \beta_1 v_2 = \beta_1' v_2$, hence $\beta_1 = \beta_1'$, impossible.
4. $\alpha_1 = \beta_1' = \epsilon$, then $u_2 = \beta_1 v_2$ and $v_2 = \alpha_1' u_2$. Then $u_2 = \beta_1 \alpha_1' u_2$, hence $\beta_1 = \alpha_1 = \alpha_1' = \beta_1' = \epsilon$, impossible.
5. cases $\beta_1 = \beta_1' = \epsilon$ and $\beta_1 = \alpha_1' = \epsilon$ are symmetrical.

Squaring Transducers: Algorithm for functionality

compute CoAcc (quadratic time)
Visited $= \{(p_0, q_0, (\epsilon, \epsilon)) \mid (p_0, q_0) \in \text{CoAcc} \cap I^2\}$
Waiting $=$ Visited
While (Waiting$\neq \varnothing$)
    Remove some $(p, q, d) \in$Waiting
    For all $p \xrightarrow{\sigma:v} p'$, $q \xrightarrow{\sigma:w} q'$ s.t. $(p', q') \in \text{CoAcc}$ **do**:
        $d' = \mathsf{delay}(d.(v, w))$
        if $(p', q', d') \notin$ Visited:
            if $\exists (p', q', d'') \in$ Visited s.t. $d' \neq d''$ **return** NO
            if $(p', q') \in F^2$ and $d' \neq (\epsilon, \epsilon)$ **return** NO
            add (p', q', d') to Waiting and to Visited
return YES

Invariant

Lemma

For all $(p, q, d) \in Visited$*, there exist* $p_0, q_0 \in I$*,* $u, v, w \in \Sigma^*$ *such that:*

1. $p_0 \xrightarrow{u:v} p$
2. $q_0 \xrightarrow{u:w} q$
3. $d = \mathsf{delay}(v, w)$
4. $(p, q) \in CoAcc$

Correctness of the algorithm

1. if it returns NO, then by the Invariant and the two observations, T is not functional

Correctness of the algorithm

1. if it returns NO, then by the Invariant and the two observations, T is not functional
2. conversely, if it returns YES, then we show that in the end, we have $(\star)$

   $$\text{Visited} \supseteq \{(p, q, \mathsf{delay}(v, w)) \mid \exists p_0 \xrightarrow{u:v} p, q_0 \xrightarrow{u:w} q,\ (p, q) \in \text{CoAcc}\}$$

   and in particular, Visited contains all such $(p, q, \mathsf{delay}(v, w))$ such that $(p, q) \in F^2 \cap \text{Acc}$.

Correctness of the algorithm

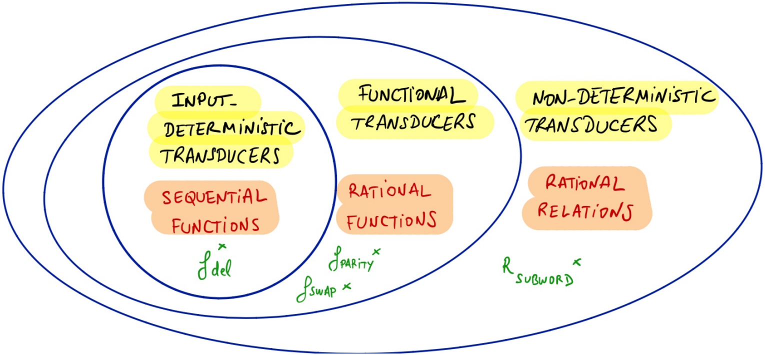
1. if it returns NO, then by the Invariant and the two observations, T is not functional
2. conversely, if it returns YES, then we show that in the end, we have $(\star)$

   $$\text{Visited} \supseteq \{(p, q, \mathsf{delay}(v, w)) \mid \exists p_0 \xrightarrow{u:v} p, q_0 \xrightarrow{u:w} q,\ (p, q) \in \text{CoAcc}\}$$

   and in particular, Visited contains all such $(p, q, \mathsf{delay}(v, w))$ such that $(p, q) \in F^2 \cap \text{Acc}$.
   If T is not functional, by Obs1 there exists (p, q, d) such that $(p, q) \in F^2 \cap \text{Acc}$ and $d \neq (\epsilon, \epsilon)$, hence the the test at line 10 eventually fails. Contradiction.

To show $\star$, use induction on $|u|$ and delay compositionality.

Summary of yesterday's talk



Given T a non-deterministic transducer:

Functionality decide if $[\![T]\!]$ is a function,

Determinizability decide if T is equivalent to some input-deterministic transducer.

➜ We have seen that Functionality can be decided in PTime.

Equivalence Problem: Results

For functional transducers T_1, T_2

- PSpace-c (hardness by automata equivalence)
- PTime if $dom(T_1) = dom(T_2)$ is known.

[1] $\exists K \forall u \; |T(u)| \leq K$

Equivalence Problem: Results

For functional transducers T_1, T_2

- PSPACE-c (hardness by automata equivalence)
- PTIME if $dom(T_1) = dom(T_2)$ is known.

In general

- Undecidable (Griffith 68), even if one alphabet is unary (Ibarra 78)

[1] $\exists K \forall u \; |T(u)| \leq K$

Equivalence Problem: Results

For functional transducers T_1, T_2

- PSPACE-c (hardness by automata equivalence)
- PTIME if $dom(T_1) = dom(T_2)$ is known.

In general

- Undecidable (Griffith 68), even if one alphabet is unary (Ibarra 78)
- Decidable for finite-valued transducers (Culik Karhumäki 86) [1].

[1] $\exists K \forall u \; |T(u)| \leq K$

Summary – Transducers

Expressiveness:

input-deterministic		functional		non-deterministic
f_{del}	$<$	f_{swap}	$<$	$R_{subword}$

???

PTime

Summary – Transducers

Expressiveness:

input-deterministic	functional	non-deterministic
f_{del} $<$	f_{swap} $<$	$R_{subword}$

??? PTime

Equivalence: ($dom(T_1) = dom(T_2)$ is known)

input-deterministic	functional	non-deterministic
PTime	PTime	undec

Determinizability

Def Given a transducer T, does there exist an input-deterministic transducer T' such that $[\![T]\!] = [\![T']\!]$?

Remark: we now assume that T is:

- functional (otherwise the answer is NO)
- trim (can be achieved in PTime)

Determinizability

Def Given a transducer T, does there exist an input-deterministic transducer T' such that $[\![T]\!] = [\![T']\!]$?

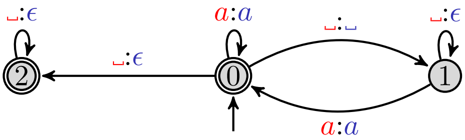
Remark: we now assume that T is:

- functional (otherwise the answer is NO)
- trim (can be achieved in PTime)

Thm (Choffrut77, Weber,Klemm,95). Determinizability is decidable in PTime.

- equivalent input-deterministic transducer of exp. size
- characterization based on a pattern of the transducer (twinning property) due to (Choffrut77)
- PTime membership due to (Weber,Klemm,95)

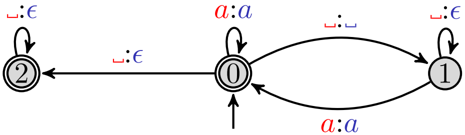
Determinizability: example



␣ = white space

Determinizability: example

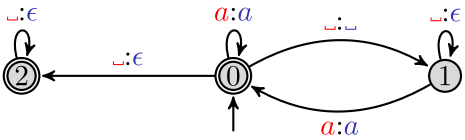
␣ = white space



␣␣aa␣␣␣a␣␣ $\mapsto$ ␣aa␣a

Determinizability: example

␣ = white space

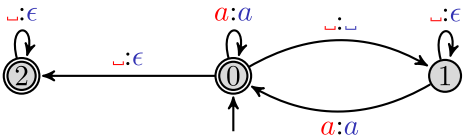


␣␣aa␣␣␣a␣␣ $\mapsto$ ␣aa␣a

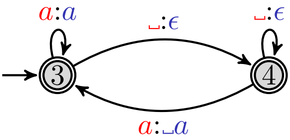
Is non-determinism needed ?

Determinizability: example

␣ = white space

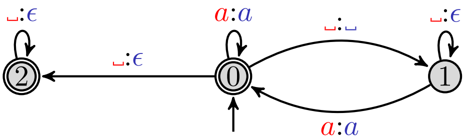


␣␣aa␣␣␣a␣␣ $\mapsto$ ␣aa␣a



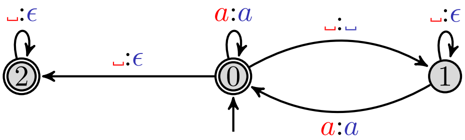
Is non-determinism needed ? No.

How to get a deterministic FT ?

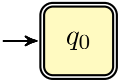


- extend automata subset construction with outputs
- output the longest common prefix

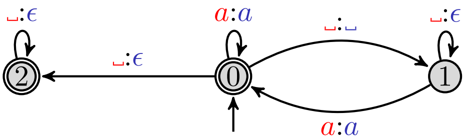
How to get a deterministic FT ?



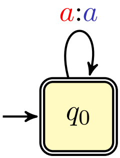
- extend automata subset construction with outputs
- output the longest common prefix



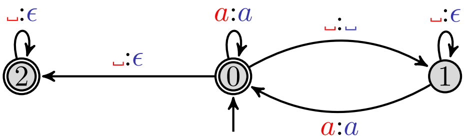
How to get a deterministic FT ?



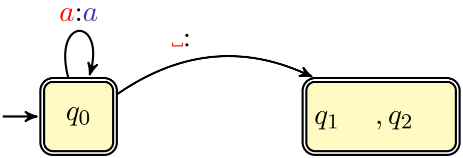
- extend automata subset construction with outputs
- output the longest common prefix



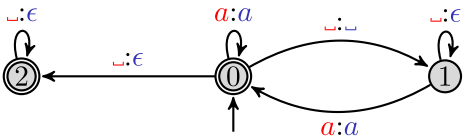
How to get a deterministic FT ?



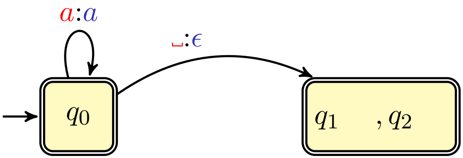
- extend automata subset construction with outputs
- output the longest common prefix



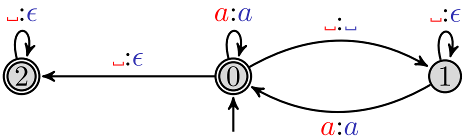
How to get a deterministic FT ?



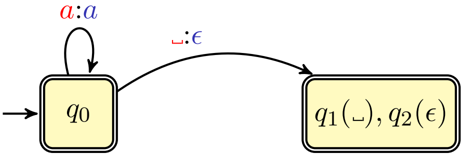
- extend automata subset construction with outputs
- output the longest common prefix



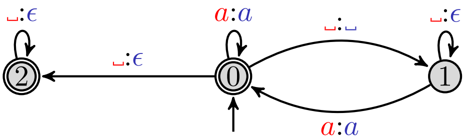
How to get a deterministic FT ?



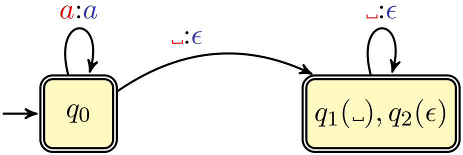
- extend automata subset construction with outputs
- output the longest common prefix



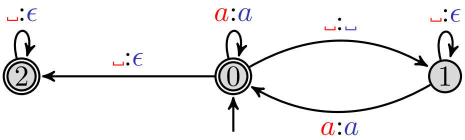
How to get a deterministic FT ?



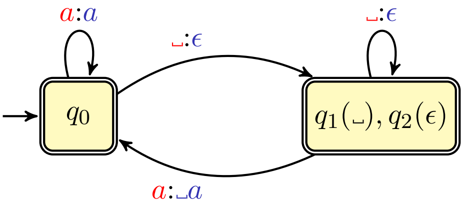
- extend automata subset construction with outputs
- output the longest common prefix



How to get a deterministic FT ?



- extend automata subset construction with outputs
- output the longest common prefix



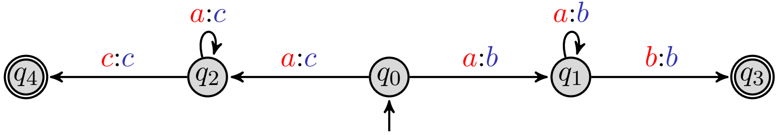
Can we always get an equivalent deterministic FT ?

Can we always get an equivalent deterministic FT ?

- not in general: input-deterministic transducers are less expressive than functional ones

Can we always get an equivalent deterministic FT ?

- not in general: input-deterministic transducers are less expressive than functional ones

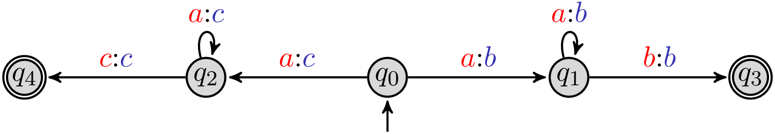


Semantics

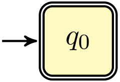
$$[\![T]\!] : \begin{cases} a^n b \mapsto b^{n+1} \\ a^n c \mapsto c^{n+1} \end{cases}$$

functional but not determinizable

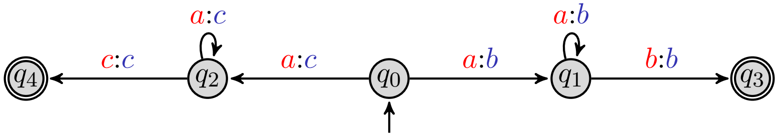
Subset construction fails ...



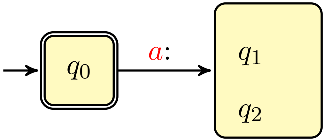
Subset construction:



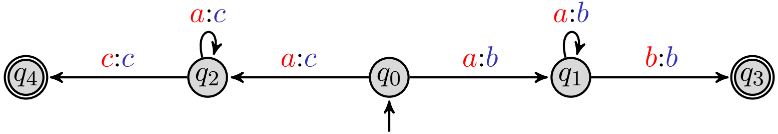
Subset construction fails ...



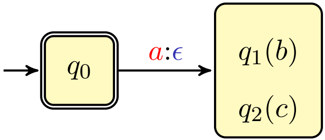
Subset construction:



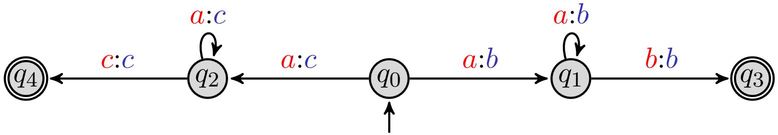
Subset construction fails ...



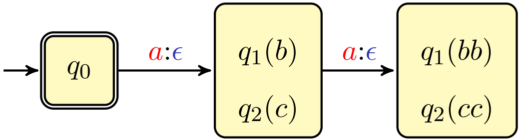
Subset construction:



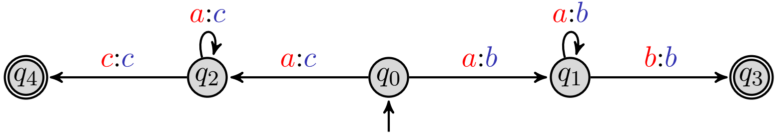
Subset construction fails ...



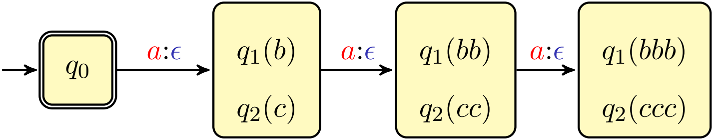
Subset construction:



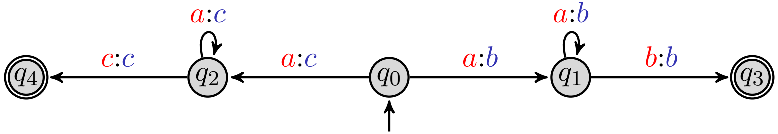
Subset construction fails ...



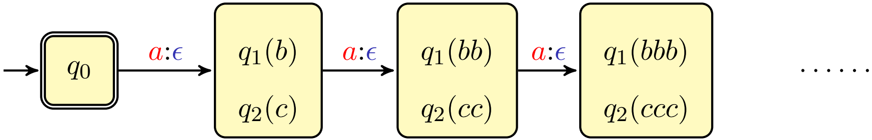
Subset construction:



Subset construction fails ...



Subset construction:



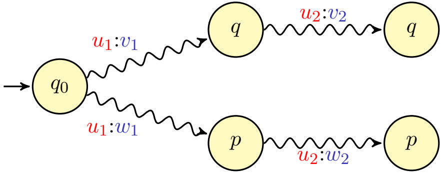
How to guarantee termination of subset construction?

Reminder:
$\mathsf{delay}(u, v) = (u', v')$ such that $u = \ell u'$, $v = \ell v'$ and $\ell = u \wedge v$.

How to guarantee termination of subset construction?

Reminder:
$\mathsf{delay}(u, v) = (u', v')$ such that $u = \ell u'$, $v = \ell v'$ and $\ell = u \wedge v$.

We say that T satisfies the Twinning Property iff for all situations

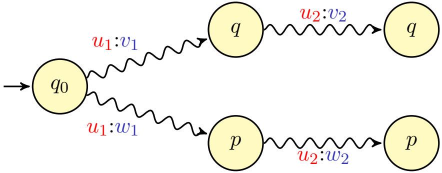


it is the case that $\mathsf{delay}(v_1, w_1) = \mathsf{delay}(v_1 v_2, w_1 w_2)$.

How to guarantee termination of subset construction?

Reminder:
$\mathsf{delay}(u, v) = (u', v')$ such that $u = \ell u'$, $v = \ell v'$ and $\ell = u \wedge v$.

We say that T satisfies the Twinning Property iff for all situations



it is the case that $\mathsf{delay}(v_1, w_1) = \mathsf{delay}(v_1v_2, w_1w_2)$.

Lemma[Characterization] T is determinizable iff it satisfies the Twinning Property.

Proof of the characterization ($T \models \mathrm{TP} \Rightarrow T$ det.)

(n: number of states, M: maximal length of output word)

Lemma If T satisfies the Twinning Property, then for all runs $p_0 \xrightarrow{u:v} p$ and $q_0 \xrightarrow{u:w} q$, we have $|\mathsf{delay}(v, w)| \leq 2n^2M$.

Proof of the characterization ($T \models \mathrm{TP} \Rightarrow T$ det.)

(n: number of states, M: maximal length of output word)

Lemma If T satisfies the Twinning Property, then for all runs $p_0 \xrightarrow{u:v} p$ and $q_0 \xrightarrow{u:w} q$, we have $|\mathsf{delay}(v, w)| \leq 2n^2 M$.

Proof: We proceed by contradiction, and consider a counter-example of minimal length, with input word u. Two cases:

- If $|u| \leq n^2$, then $|\mathsf{delay}(v, w)| \leq |v| + |w| \leq 2n^2 M$.
- If $|u| > n^2$, then there is a loop. By the twinning property, and the compositionality of delay, we obtain a shorter counter-example. Contradiction. □

Proof of the characterization ($T \models \mathrm{TP} \Rightarrow T$ det.)

(n: number of states, M: maximal length of output word)

Lemma If T satisfies the Twinning Property, then for all runs $p_0 \xrightarrow{u:v} p$ and $q_0 \xrightarrow{u:w} q$, we have $|\mathsf{delay}(v, w)| \leq 2n^2M$.

Proof: We proceed by contradiction, and consider a counter-example of minimal length, with input word u. Two cases:

- If $|u| \leq n^2$, then $|\mathsf{delay}(v, w)| \leq |v| + |w| \leq 2n^2M$.
- If $|u| > n^2$, then there is a loop. By the twinning property, and the compositionality of delay, we obtain a shorter counter-example. Contradiction. □

We have: $T \models$ Twinning Property $\Rightarrow$ Subset constr. terminates

$\Rightarrow T$ determinizable

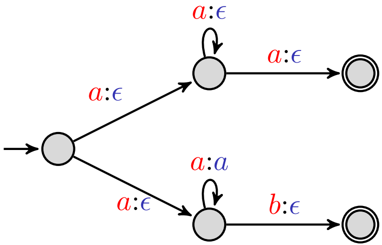
Proof of the characterization (T det. $\Rightarrow T \models \mathrm{TP}$)

We only give some intuition.

Proof of the characterization (T det. $\Rightarrow T \models$ TP)

We only give some intuition.

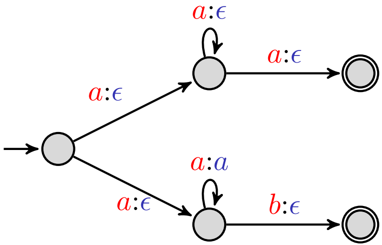
By contraposition, suppose that $T \not\models$ Twinning Property.
For instance:



Proof of the characterization (T det. $\Rightarrow T \models$ TP)

We only give some intuition.

By contraposition, suppose that $T \not\models$ Twinning Property.
For instance:

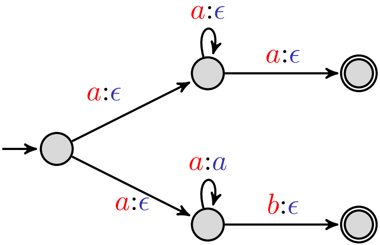


Iterating the loop yields an infinite number of distinct delays.
$(\epsilon, a), (\epsilon, aa), (\epsilon, aaa) \dots$

Proof of the characterization (T det. $\Rightarrow T \models$ TP)

We only give some intuition.

By contraposition, suppose that $T \not\models$ Twinning Property.
For instance:



Iterating the loop yields an infinite number of distinct delays.
$(\epsilon, a), (\epsilon, aa), (\epsilon, aaa) \ldots$

Any equivalent input-deterministic transducer should store them, impossible with finitely many states.

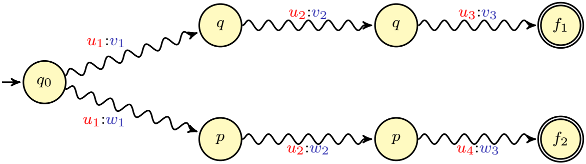
Proof of the characterization (T det. $\Rightarrow T \models$ TP)

We prove now T determinizable $\Rightarrow T \models$ Twinning Property.

Proof of the characterization (T det. $\Rightarrow T \models$ TP)

We prove now T determinizable $\Rightarrow T \models$ Twinning Property.

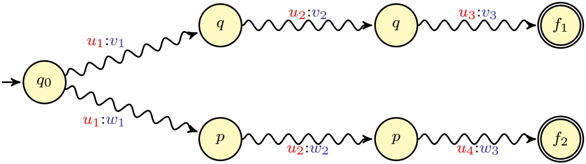
Consider an input-deterministic transducer D s.t. $[\![T]\!] = [\![D]\!]$ and an instance of the Twinning Property:



Proof of the characterization (T det. $\Rightarrow T \models$ TP)

We prove now T determinizable $\Rightarrow T \models$ Twinning Property.

Consider an input-deterministic transducer D s.t. $[\![T]\!] = [\![D]\!]$ and an instance of the Twinning Property:

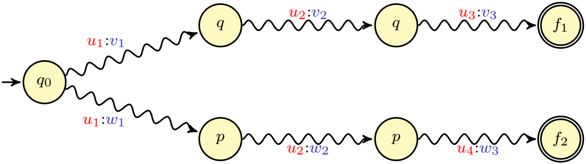


$\forall i, u_1 u_2^i u_3 \in dom([\![T]\!]) \Rightarrow \exists i \geq 1, j \geq 0 \mid$ there is a loop in D on u_2^i after $u_1 u_2^j$

Proof of the characterization (T det. $\Rightarrow T \models$ TP)

We prove now T determinizable $\Rightarrow T \models$ Twinning Property.

Consider an input-deterministic transducer D s.t. $[\![T]\!] = [\![D]\!]$ and an instance of the Twinning Property:



$\forall i, u_1 u_2^i u_3 \in dom([\![T]\!]) \Rightarrow \exists i \geq 1, j \geq 0 \mid$ there is a loop in D on u_2^i after $u_1 u_2^j$

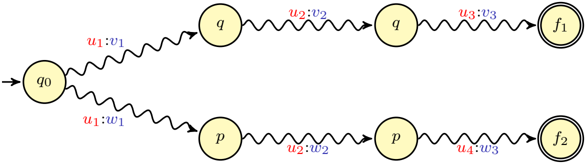
We obtain : (for some output words x_1, x_2, x_3, y_3 in D)

$$\forall \ell \geq 0, \quad \begin{array}{lll} [\![T]\!](u_1 u_2^j\ u_2^{\ell i}\ u_3) & = v_1 v_2^j\ v_2^{\ell i} v_3 & = x_1 x_2^\ell x_3 \\ [\![T]\!](u_1 u_2^j\ u_2^{\ell i}\ u_4) & = w_1 w_2^j\ w_2^{\ell i} w_3 & = x_1 x_2^\ell y_3 \end{array}$$

Proof of the characterization (T det. $\Rightarrow T \models$ TP)

We prove now T determinizable $\Rightarrow T \models$ Twinning Property.

Consider an input-deterministic transducer D s.t. $[\![T]\!] = [\![D]\!]$ and an instance of the Twinning Property:



$\forall i, u_1 u_2^i u_3 \in dom([\![T]\!]) \Rightarrow \exists i \geq 1, j \geq 0 \mid$ there is a loop in D on u_2^i after $u_1 u_2^j$

We obtain : (for some output words x_1, x_2, x_3, y_3 in D)

$$\forall \ell \geq 0, \quad \begin{aligned} [\![T]\!](u_1 u_2^j \ u_2^{\ell i} \ u_3) &= v_1 v_2^j \ v_2^{\ell i} v_3 &&= x_1 x_2^\ell x_3 \\ [\![T]\!](u_1 u_2^j \ u_2^{\ell i} \ u_4) &= w_1 w_2^j \ w_2^{\ell i} w_3 &&= x_1 x_2^\ell y_3 \end{aligned}$$

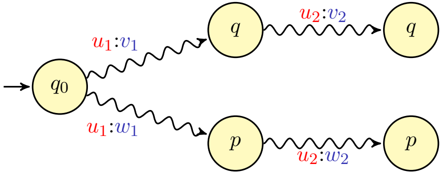
Thus $v_1 v_2^\omega = x_1 x_2^\omega = w_1 w_2^\omega$ and $|v_2| = |w_2|$.

This entails $\mathsf{delay}(v_1, w_1) = \mathsf{delay}(v_1 v_2, w_1 w_2)$. $\square$

Decidability of determinizability

Lemma[Characterization] T is determinizable iff it satisfies the Twinning Property.

For all situations

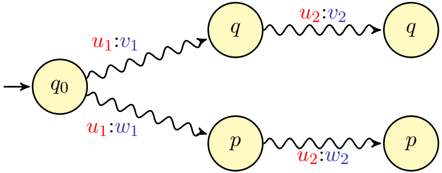


it is the case that $\mathsf{delay}(v_1, w_1) = \mathsf{delay}(v_1 v_2, w_1 w_2)$.

Decidability of determinizability

Lemma[Characterization] T is determinizable iff it satisfies the Twinning Property.

For all situations



it is the case that $\mathsf{delay}(v_1, w_1) = \mathsf{delay}(v_1 v_2, w_1 w_2)$.

Claim: T <u>violates</u> the TP iff there exists situation as above such that:

1. $|v_2| \neq |w_2|$, or
2. $|v_2| = |w_2| \neq 0$ and there is a mismatch between v_1 and w_1

Decidability of determinizability (2)

T violates the TP iff there exists situation as above such that:

1. $|v_2| \neq |w_2|$, or
2. $|v_2| = |w_2| \neq 0$ and there is a mismatch between v_1 and w_1

Decidability of determinizability (2)

T violates the TP iff there exists situation as above such that:

1. $|v_2| \neq |w_2|$, or
2. $|v_2| = |w_2| \neq 0$ and there is a mismatch between v_1 and w_1

To decide 1.:
consider a weighted graph with vertices (p, q) such that $(p, q) \xrightarrow{n} (p', q')$ iff $\exists \sigma$, $p \xrightarrow{\sigma : v} p'$, $q \xrightarrow{\sigma : w} q'$ and $n = |v| - |w|$
$\rightsquigarrow$ verify that every cycle has weight 0

Decidability of determinizability (2)

T violates the TP iff there exists situation as above such that:

1. $|v_2| \neq |w_2|$, or
2. $|v_2| = |w_2| \neq 0$ and there is a mismatch between v_1 and w_1

To decide 1.:
consider a weighted graph with vertices (p, q) such that $(p, q) \xrightarrow{n} (p', q')$ iff $\exists\sigma$, $p \xrightarrow{\sigma:v} p'$, $q \xrightarrow{\sigma:w} q'$ and $n = |v| - |w|$
$\rightsquigarrow$ verify that every cycle has weight 0

To decide 2.:

- Compute $X = \{(p, q) \text{ s.t. } \exists u_2.(p, q) \xrightarrow{u_2:(v_2,w_2)} (p, q), |v_2| = |w_2| \neq 0\}$

Decidability of determinizability (2)

T violates the TP iff there exists situation as above such that:

1. $|v_2| \neq |w_2|$, or
2. $|v_2| = |w_2| \neq 0$ and there is a mismatch between v_1 and w_1

To decide 1.:
consider a weighted graph with vertices (p, q) such that $(p, q) \xrightarrow{n} (p', q')$ iff $\exists \sigma$, $p \xrightarrow{\sigma : v} p'$, $q \xrightarrow{\sigma : w} q'$ and $n = |v| - |w|$
$\rightsquigarrow$ verify that every cycle has weight 0

To decide 2.:

- Compute $X = \{(p, q) \text{ s.t. } \exists u_2.(p, q) \xrightarrow{u_2 : (v_2, w_2)} (p, q), |v_2| = |w_2| \neq 0\}$
- non-deterministically guess a path $(p_0, p_0) \xrightarrow{u_1 : (v_1, w_1)} (p, q) \in X$ such that there is a mismatch between v_1 and w_1

Decidability of determinizability (2)

T violates the TP iff there exists situation as above such that:

1. $|v_2| \neq |w_2|$, or
2. $|v_2| = |w_2| \neq 0$ and there is a mismatch between v_1 and w_1

To decide 1.:
consider a weighted graph with vertices (p,q) such that $(p,q) \xrightarrow{n} (p',q')$ iff $\exists\sigma$, $p \xrightarrow{\sigma:v} p'$, $q \xrightarrow{\sigma:w} q'$ and $n = |v| - |w|$
$\rightsquigarrow$ verify that every cycle has weight 0

To decide 2.:

- Compute $X = \{(p,q) \text{ s.t. } \exists u_2.(p,q) \xrightarrow{u_2:(v_2,w_2)} (p,q), |v_2| = |w_2| \neq 0\}$
- non-deterministically guess a path $(p_0,p_0) \xrightarrow{u_1:(v_1,w_1)} (p,q) \in X$ such that there is a mismatch between v_1 and w_1
  - its length can be bounded by $2n^2$
  - non-deterministically guess the mismatch
  - position of the mismatch stored using two registers, whose values are bounded by $2n^2M$
  - $\rightsquigarrow$ NLogSpace

Decidability of determinizability (2)

T violates the TP iff there exists situation as above such that:

1. $|v_2| \neq |w_2|$, or
2. $|v_2| = |w_2| \neq 0$ and there is a mismatch between v_1 and w_1

To decide 1.:
consider a weighted graph with vertices (p, q) such that $(p, q) \xrightarrow{n} (p', q')$ iff $\exists \sigma$, $p \xrightarrow{\sigma : v} p'$, $q \xrightarrow{\sigma : w} q'$ and $n = |v| - |w|$
$\rightsquigarrow$ verify that every cycle has weight 0

To decide 2.:

- Compute $X = \{(p, q) \text{ s.t. } \exists u_2.(p, q) \xrightarrow{u_2 : (v_2, w_2)} (p, q), |v_2| = |w_2| \neq 0\}$
- non-deterministically guess a path $(p_0, p_0) \xrightarrow{u_1 : (v_1, w_1)} (p, q) \in X$ such that there is a mismatch between v_1 and w_1
    - its length can be bounded by $2n^2$
    - non-deterministically guess the mismatch
    - position of the mismatch stored using two registers, whose values are bounded by $2n^2 M$
    - $\rightsquigarrow$ NLogSpace

All together: decidable in PTime (and even in NLogSpace)

Summary – Transducers

Expressiveness:

input-deterministic		functional		non-deterministic
f_{del}	$<$	f_{swap}	$<$	$R_{subword}$

PTime PTime

Summary – Transducers

Expressiveness:

input-deterministic	functional	non-deterministic
f_{del} $<$	f_{swap} $<$	$R_{subword}$

PTIME PTIME

Equivalence: ($dom(T_1) = dom(T_2)$ is known)

input-deterministic	functional	non-deterministic
PTIME	PTIME	undec

Logics for transductions

MSO on words

Over some finite alphabet Σ:

$$\varphi \ ::= \ \varphi \wedge \varphi \mid \neg\varphi \mid \exists x \varphi \mid \exists X \varphi \mid x \in X \mid \sigma(x) \mid S(x,y) \qquad \sigma \in \Sigma$$

Over finite words, (set) variables interpreted by (sets of) positions.
Notation: $\leq$ is the transitive closure of S

MSO on words

Over some finite alphabet Σ:

$$\varphi \ ::= \ \varphi \wedge \varphi \mid \neg\varphi \mid \exists x \varphi \mid \exists X \varphi \mid x \in X \mid \sigma(x) \mid S(x,y) \qquad \sigma \in \Sigma$$

Over finite words, (set) variables interpreted by (sets of) positions.
Notation: $\leq$ is the transitive closure of S

Some examples

- first position is an a: $\exists x \ a(x) \wedge \forall y (x \leq y)$
- does not contain ab: $\neg \exists x \exists y . S(x,y) \wedge a(x) \wedge b(y)$
- counting modulo: odd number of a, even length, ...

MSO on words

Over some finite alphabet Σ:

$$\varphi \ ::= \ \varphi \wedge \varphi \mid \neg\varphi \mid \exists x \varphi \mid \exists X \varphi \mid x \in X \mid \sigma(x) \mid S(x,y) \qquad \sigma \in \Sigma$$

Over finite words, (set) variables interpreted by (sets of) positions.
Notation: $\leq$ is the transitive closure of S

Some examples

- first position is an a: $\exists x\ a(x) \wedge \forall y(x \leq y)$
- does not contain ab: $\neg\exists x \exists y. S(x,y) \wedge a(x) \wedge b(y)$
- counting modulo: odd number of a, even length, ...

$$L_\phi = \{w \in \Sigma^* \mid w \models \phi\}$$

MSO on words

Over some finite alphabet Σ:

$$\varphi \ ::= \ \varphi \wedge \varphi \mid \neg\varphi \mid \exists x \varphi \mid \exists X \varphi \mid x \in X \mid \sigma(x) \mid S(x,y) \qquad \sigma \in \Sigma$$

Over finite words, (set) variables interpreted by (sets of) positions.
Notation: $\leq$ is the transitive closure of S

Some examples

- first position is an a: $\exists x\ a(x) \wedge \forall y (x \leq y)$
- does not contain ab: $\neg \exists x \exists y . S(x,y) \wedge a(x) \wedge b(y)$
- counting modulo: odd number of a, even length, ...

$$L_\phi = \{ w \in \Sigma^* \mid w \models \phi \}$$

Büchi-Elgot-Trakhenbrot

$L \subseteq \Sigma^*$ is MSO-definable iff it is recognisable by some FA.

Examples of MSO formulae

- The first position:
  $\phi_{first}(x) =$

Examples of MSO formulae

- The first position:
  $\phi_{first}(x) =$

- Sets X, Y partition the set of positions:
  $\phi_{partition}(X, Y) =$

Examples of MSO formulae

- The first position:
  $\phi_{first}(x) =$

- Sets X, Y partition the set of positions:
  $\phi_{partition}(X, Y) =$

- Set X is the set of even positions:
  $\phi_{even}(X) =$

Extension to transductions

$\phi(x)$: MSO formula with one free FO variable x

$w \in \Sigma^*$, $i \in \{1, \ldots, |w|\}$

Notation: $w, i \models \phi(x)$: $\phi(x)$ evaluates to True at position i in w

Extension to transductions

$\phi(x)$: MSO formula with one free FO variable x
$w \in \Sigma^*$, $i \in \{1, \ldots, |w|\}$
Notation: $w, i \models \phi(x)$: $\phi(x)$ evaluates to True at position i in w

Consider formula $\phi_{v_1}, \ldots, \phi_{v_k}$:
$w = a_1 \ldots a_n \rightsquigarrow v_{j_1} \ldots v_{j_n} \iff \forall i \in \{1, \ldots, n\}, w, i \models \phi_{v_{j_i}}(x)$

Extension to transductions

$\phi(x)$: MSO formula with one free FO variable x
$w \in \Sigma^*$, $i \in \{1, \ldots, |w|\}$
Notation: $w, i \models \phi(x)$: $\phi(x)$ evaluates to True at position i in w

Consider formula $\phi_{v_1}, \ldots, \phi_{v_k}$:
$w = a_1 \ldots a_n \rightsquigarrow v_{j_1} \ldots v_{j_n} \iff \forall i \in \{1, \ldots, n\}, w, i \models \phi_{v_{j_i}}(x)$

Example (Delete a's)

$$\phi_b(x) \equiv b(x)$$

$$\phi_\epsilon(x) \equiv a(x)$$

Realizes the function $f : u \in \{a, b\}^* \mapsto b^{|u|_b}$

Extension to transductions

Example (Append #)

- replace label σ of x by σ if x is **not** the last position
- replace label σ of x by $\sigma\#$ if x is the last position

Extension to transductions

Example (Append #)

- replace label σ of x by σ if x is **not** the last position

$$\phi_{\sigma}(x) \equiv \sigma(x) \wedge \exists y\ S(x, y)$$

- replace label σ of x by $\sigma\#$ if x is the last position

Extension to transductions

Example (Append #)

- replace label σ of x by σ if x is **not** the last position

$$\phi_{\sigma}(x) \equiv \sigma(x) \wedge \exists y \; S(x, y)$$

- replace label σ of x by $\sigma\#$ if x is the last position

$$\phi_{\sigma\#}(x) \equiv \sigma(x) \wedge \forall y \; \neg S(x, y)$$

Extension to transductions

Example (Add a parity bit)

- replace label σ of x by 1σ if x is the first position and odd number of 1

- replace label σ of x by 0σ if x is the first position and even number of 1

- replace label σ of x by σ if x is not the first position

Extension to transductions

Example (Add a parity bit)

- replace label σ of x by 1σ if x is the first position and odd number of 1

$$\phi_{1\sigma}(x) \equiv \sigma(x) \wedge \forall y \, \neg S(y,x) \wedge \phi_{odd_1}$$

- replace label σ of x by 0σ if x is the first position and even number of 1

- replace label σ of x by σ if x is not the first position

Extension to transductions

Example (Add a parity bit)

- replace label σ of x by 1σ if x is the first position and odd number of 1

$$\phi_{1\sigma}(x) \equiv \sigma(x) \wedge \forall y\ \neg S(y,x) \wedge \phi_{odd_1}$$

- replace label σ of x by 0σ if x is the first position and even number of 1

$$\phi_{0\sigma}(x) \equiv \sigma(x) \wedge \forall y\ \neg S(y,x) \wedge \phi_{even_1}$$

- replace label σ of x by σ if x is not the first position

Extension to transductions

Example (Add a parity bit)

- replace label σ of x by 1σ if x is the first position and odd number of 1

$$\phi_{1\sigma}(x) \equiv \sigma(x) \wedge \forall y\ \neg S(y,x) \wedge \phi_{odd_1}$$

- replace label σ of x by 0σ if x is the first position and even number of 1

$$\phi_{0\sigma}(x) \equiv \sigma(x) \wedge \forall y\ \neg S(y,x) \wedge \phi_{even_1}$$

- replace label σ of x by σ if x is not the first position

$$\phi_{\sigma}(x) \equiv \sigma(x) \wedge \exists y\ S(y,x)$$

Büchi Theorem for Rational Transductions

Def $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable if it can be "described" by a finite set of formulas $\phi_{v_1}(x), \dots, \phi_{v_k}(x)$ $(v_1, \dots, v_k \in \Sigma^*)$.

$$w = a_1 \dots a_n \rightsquigarrow v_{j_1} \dots v_{j_n} \iff \forall i \in \{1, \dots, n\}, w, i \models \phi_{v_{j_i}}(x)$$

Büchi Theorem for Rational Transductions

Def $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable if it can be "described" by a finite set of formulas $\phi_{v_1}(x), \ldots, \phi_{v_k}(x)$ $(v_1, \ldots, v_k \in \Sigma^*)$.

$$w = a_1 \ldots a_n \rightsquigarrow v_{j_1} \ldots v_{j_n} \iff \forall i \in \{1, \ldots, n\}, w, i \models \phi_{v_{j_i}}(x)$$

Thm $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable iff it is realisable by a finite state transducer.

Büchi Theorem for Rational Transductions

Def $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable if it can be "described" by a finite set of formulas $\phi_{v_1}(x), \ldots, \phi_{v_k}(x)$ $(v_1, \ldots, v_k \in \Sigma^*)$.

$$w = a_1 \ldots a_n \rightsquigarrow v_{j_1} \ldots v_{j_n} \iff \forall i \in \{1, \ldots, n\}, w, i \models \phi_{v_{j_i}}(x)$$

Thm $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable iff it is realisable by a finite state transducer.

What about mirror ? $ejcim \mapsto micje$

Büchi Theorem for Rational Transductions

Def $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable if it can be "described" by a finite set of formulas $\phi_{v_1}(x), \dots, \phi_{v_k}(x)$ $(v_1, \dots, v_k \in \Sigma^*)$.

$w = a_1 \dots a_n \rightsquigarrow v_{j_1} \dots v_{j_n} \iff \forall i \in \{1, \dots, n\}, w, i \models \phi_{v_{j_i}}(x)$

Thm $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable iff it is realisable by a finite state transducer.

What about mirror ? $\qquad$ *ejcim* $\mapsto$ *micje*

Replace label of position x by σ if $last - x$ is labeled σ.

Büchi Theorem for Rational Transductions

Def $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable if it can be "described" by a finite set of formulas $\phi_{v_1}(x), \ldots, \phi_{v_k}(x)$ $(v_1, \ldots, v_k \in \Sigma^*)$.

$$w = a_1 \ldots a_n \rightsquigarrow v_{j_1} \ldots v_{j_n} \iff \forall i \in \{1, \ldots, n\}, w, i \models \phi_{v_{j_i}}(x)$$

Thm $f : \Sigma^* \hookrightarrow \Sigma^*$ is MSO-definable iff it is realisable by a finite state transducer.

What about mirror ? $ejcim \mapsto micje$

Replace label of position x by σ if $last - x$ is labeled σ.
Not MSO-definable.

Proof: from transducers to MSO

W.l.o.g., we assume T unambiguous.

For each transition $t = (p, a, v, q)$ of T, we define the language $L_t \subseteq (\Sigma \times \{0,1\})^*$ such that:

$$\overline{w} \in L_t \iff \exists \text{ run } q_0 \xrightarrow{u|v} q_f \text{ s.t. } \pi_2(\overline{w})[i] = 1 \text{ iff } t \text{ used at position } i$$

Proof: from transducers to MSO

W.l.o.g., we assume T unambiguous.

For each transition $t = (p, a, v, q)$ of T, we define the language $L_t \subseteq (\Sigma \times \{0, 1\})^*$ such that:

$$\overline{w} \in L_t \iff \exists \text{ run } q_0 \xrightarrow{u|v} q_f \text{ s.t. } \pi_2(\overline{w})[i] = 1 \text{ iff } t \text{ used at position } i$$

L_t is regular, recognized by A_t, obtained as follows:

- $p \xrightarrow{(a,1)} q$
- $p' \xrightarrow{(b,0)} q'$ for each transition $t' = (p', b, v', q') \neq t$

Proof: from transducers to MSO

W.l.o.g., we assume T unambiguous.

For each transition $t = (p, a, v, q)$ of T, we define the language $L_t \subseteq (\Sigma \times \{0, 1\})^*$ such that:

$$\overline{w} \in L_t \iff \exists \text{ run } q_0 \xrightarrow{u|v} q_f \text{ s.t. } \pi_2(\overline{w})[i] = 1 \text{ iff } t \text{ used at position } i$$

L_t is regular, recognized by A_t, obtained as follows:

- $p \xrightarrow{(a,1)} q$
- $p' \xrightarrow{(b,0)} q'$ for each transition $t' = (p', b, v', q') \neq t$

By the previous result, L_t can be translated into $\phi_t(x)$.

Proof: from transducers to MSO

W.l.o.g., we assume T unambiguous.

For each transition $t = (p, a, v, q)$ of T, we define the language $L_t \subseteq (\Sigma \times \{0, 1\})^*$ such that:

$$\overline{w} \in L_t \iff \exists \text{ run } q_0 \xrightarrow{u|v} q_f \text{ s.t. } \pi_2(\overline{w})[i] = 1 \text{ iff } t \text{ used at position } i$$

L_t is regular, recognized by A_t, obtained as follows:

- $p \xrightarrow{(a,1)} q$
- $p' \xrightarrow{(b,0)} q'$ for each transition $t' = (p', b, v', q') \neq t$

By the previous result, L_t can be translated into $\phi_t(x)$.

We let $\phi_v(x) = \bigvee_{t=(p,a,v,q)} \phi_t(x)$.

Proof: from MSO to transducers

For each $\phi_v(x)$, build an automaton A_v that recognizes words $\overline{w} \in (\Sigma \times \{0,1\})^*$ such that $\pi_2(\overline{w})[i] = 1$ iff $w, i \models \phi_v(x)$.

Proof: from MSO to transducers

For each $\phi_v(x)$, build an automaton A_v that recognizes words $\overline{w} \in (\Sigma \times \{0,1\})^*$ such that $\pi_2(\overline{w})[i] = 1$ iff $w, i \models \phi_v(x)$.

Claim: As $\phi_{v_1}, \ldots, \phi_{v_k}$ define a function f, for each word $w \in dom(f)$, and for each position $i \in \{1, \ldots, |w|\}$, there is exactly one j such that $w, i \models \phi_{v_j}$.

Proof: from MSO to transducers

For each $\phi_v(x)$, build an automaton A_v that recognizes words $\overline{w} \in (\Sigma \times \{0,1\})^*$ such that $\pi_2(\overline{w})[i] = 1$ iff $w, i \models \phi_v(x)$.

Claim: As $\phi_{v_1}, \dots, \phi_{v_k}$ define a function f, for each word $w \in dom(f)$, and for each position $i \in \{1, \dots, |w|\}$, there is exactly one j such that $w, i \models \phi_{v_j}$.

Consider the automaton $A = \Pi_{j=1}^{k} A_{v_j}$, synchronised on Σ. Transform it into a transducer by outputting v_j if transition $(a, 1)$ is used in A_{v_j}.

Proof: from MSO to transducers

For each $\phi_v(x)$, build an automaton A_v that recognizes words $\overline{w} \in (\Sigma \times \{0,1\})^*$ such that $\pi_2(\overline{w})[i] = 1$ iff $w, i \models \phi_v(x)$.

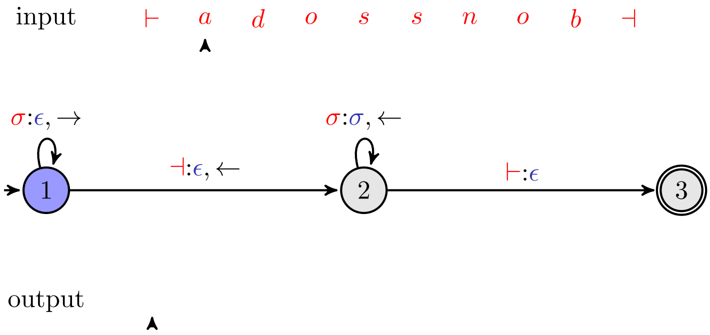
Claim: As $\phi_{v_1}, \ldots, \phi_{v_k}$ define a function f, for each word $w \in dom(f)$, and for each position $i \in \{1, \ldots, |w|\}$, there is exactly one j such that $w, i \models \phi_{v_j}$.

Consider the automaton $A = \Pi_{j=1}^{k} A_{v_j}$, synchronised on Σ. Transform it into a transducer by outputting v_j if transition $(a, 1)$ is used in A_{v_j}.

This construction is correct thanks to previous claim. $\square$

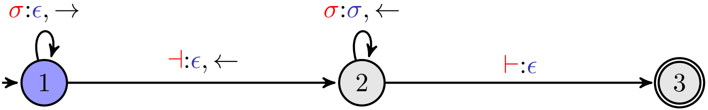
The class of regular functions

Two-way finite transducers (2FT)



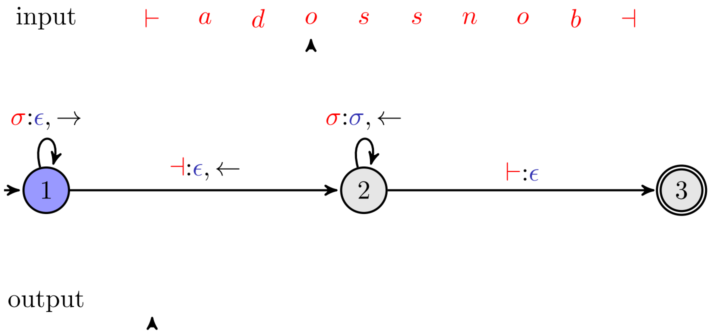
Two-way finite transducers (2FT)

input $\vdash$ a d o s s n o b $\dashv$

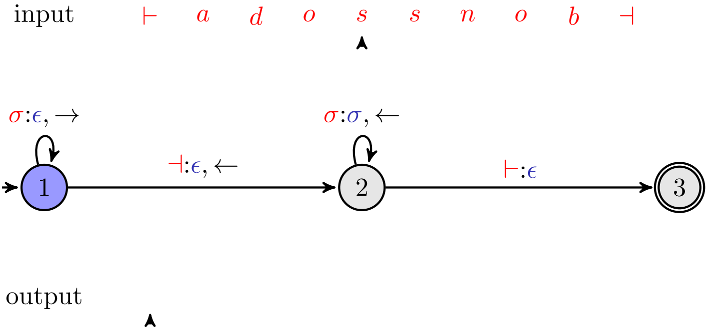


output

Two-way finite transducers (2FT)

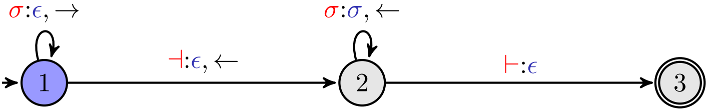


Two-way finite transducers (2FT)



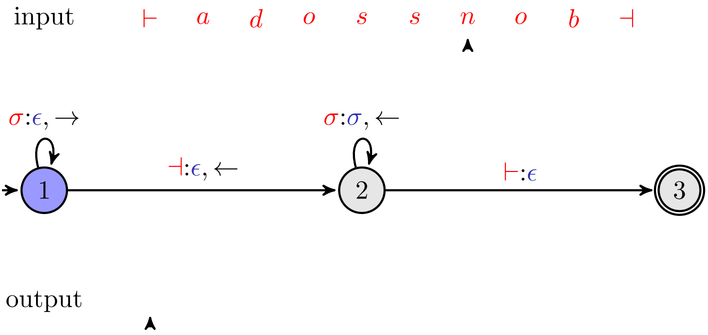
Two-way finite transducers (2FT)

input $\vdash$ a d o s s n o b $\dashv$



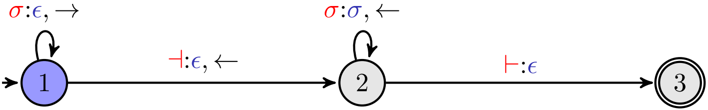
output

Two-way finite transducers (2FT)



Two-way finite transducers (2FT)

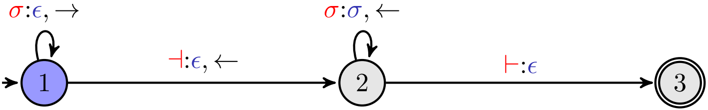
input $\vdash$ a d o s s n o b $\dashv$



output

Two-way finite transducers (2FT)

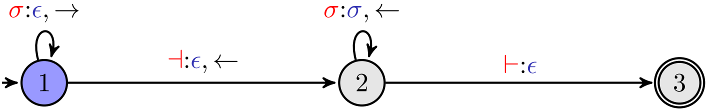
input $\vdash$ a d o s s n o b $\dashv$



output

Two-way finite transducers (2FT)

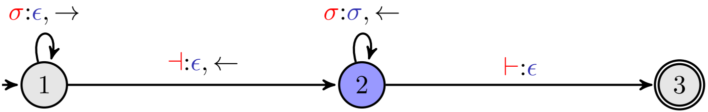
input $\vdash$ a d o s s n o b $\dashv$



output

Two-way finite transducers (2FT)

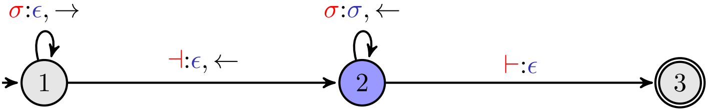
input $\vdash$ a d o s s n o b $\dashv$



output

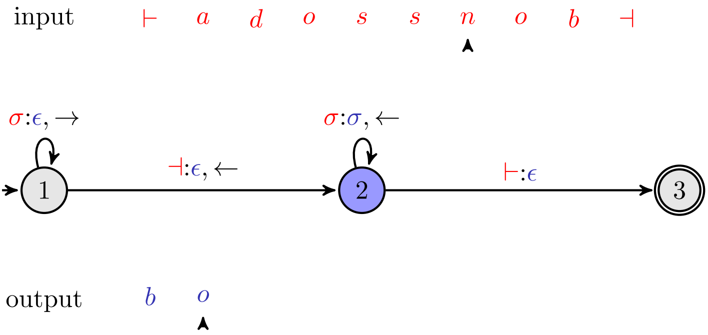
Two-way finite transducers (2FT)

input $\vdash$ a d o s s n o b $\dashv$



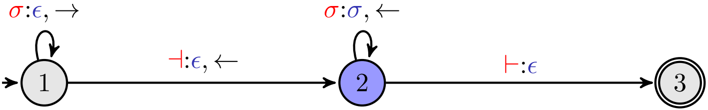
output b

Two-way finite transducers (2FT)



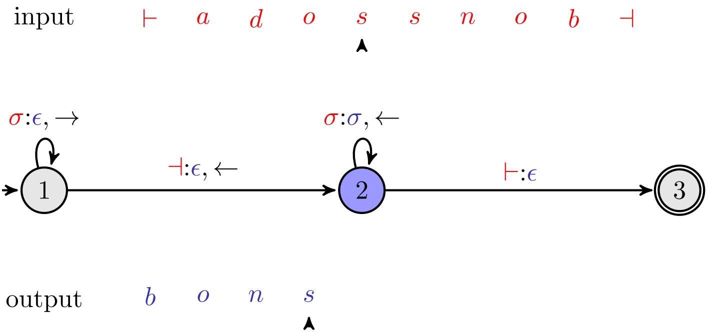
Two-way finite transducers (2FT)

input $\vdash$ a d o s s n o b $\dashv$



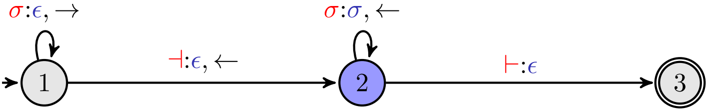
output b o n

Two-way finite transducers (2FT)



Two-way finite transducers (2FT)

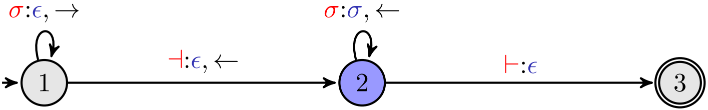
input $\vdash \quad a \quad d \quad o \quad s \quad s \quad n \quad o \quad b \quad \dashv$



output $b \quad o \quad n \quad s \quad s$

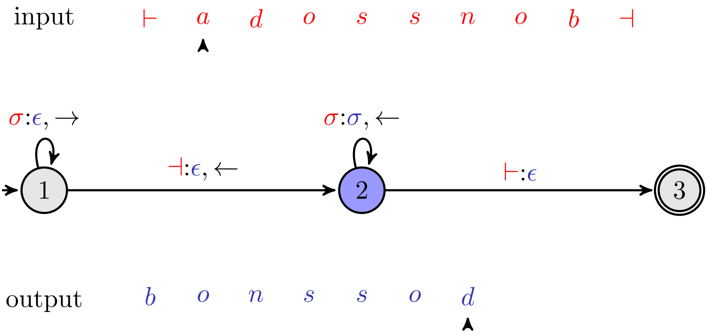
Two-way finite transducers (2FT)

input $\vdash$ a d o s s n o b $\dashv$



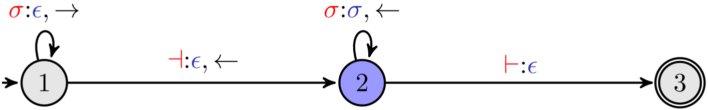
output b o n s s o

Two-way finite transducers (2FT)



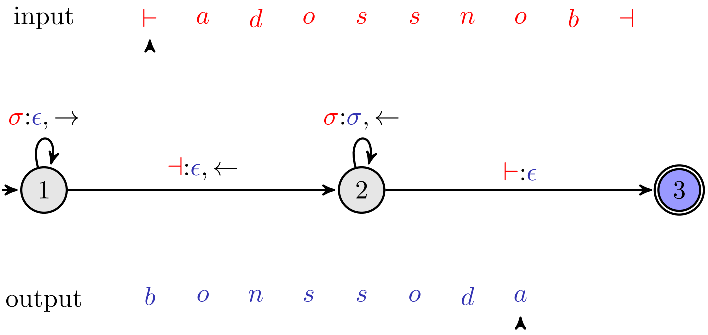
Two-way finite transducers (2FT)

input $\vdash$ a d o s s n o b $\dashv$



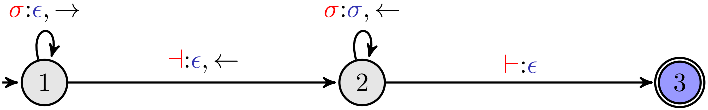
output b o n s s o d a

Two-way finite transducers (2FT)



Two-way finite transducers (2FT)

input $\vdash$ a d o s s n o b $\dashv$

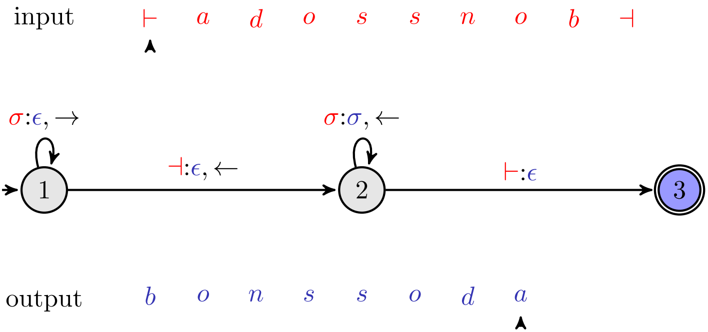


output b o n s s o d a

Other examples

$march, 2019 \mapsto 2019, march$

Two-way finite transducers (2FT)

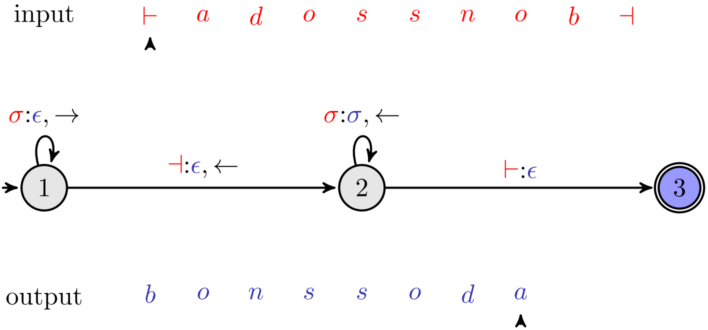


Other examples

$march, 2019 \mapsto 2019, march$

$u \mapsto uu$ (copy)

Two-way finite transducers (2FT)



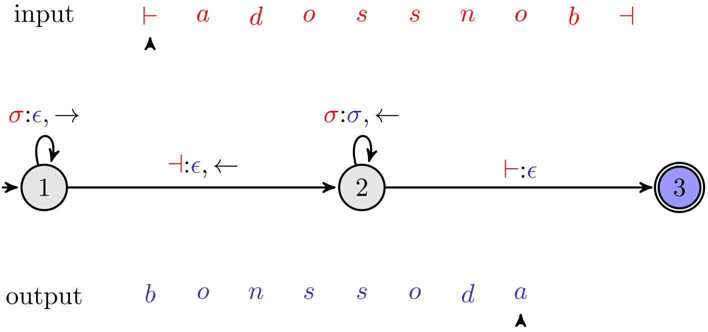
Other examples

$march, 2019 \mapsto 2019, march$

$u \mapsto uu$ (copy)

$u \mapsto f_1(u) f_2(u)$ (copy+trans)

Two-way finite transducers (2FT)



Other examples

$march, 2019 \mapsto 2019, march$

$u \mapsto uu$ (copy)

$u \mapsto f_1(u)f_2(u)$ (copy+trans)

$u_1\#u_2\#\ldots\#u_k \mapsto \overline{u_1}\#\ldots\#\overline{u_k}$ (local reverse)

Some important results on two-way transducers

Over (functional) transductions:

- equivalence is decidable in PSPACE

(Gurari 82) (Culik, Karhumäki,87)

[2] $\Sigma^* a \Sigma^*$ is not definable by any one-way reversible automaton

Some important results on two-way transducers

Over (functional) transductions:

- equivalence is decidable in PSPACE

  (Gurari 82) (Culik, Karhumäki,87)

- closed under composition

  (Chytil, Jakl, 77)

[2] $\Sigma^* a \Sigma^*$ is not definable by any one-way reversible automaton

Some important results on two-way transducers

Over (functional) transductions:

- equivalence is decidable in PSPACE

  (Gurari 82) (Culik, Karhumäki,87)
- closed under composition

  (Chytil, Jakl, 77)
- equivalent to **reversible**[2] two-way transducers

  (Dartois,Fournier,Jecker,Lhote,17)

[2] $\Sigma^* a \Sigma^*$ is not definable by any one-way reversible automaton

Some important results on two-way transducers

Over (functional) transductions:

- equivalence is decidable in PSPACE

  (Gurari 82) (Culik, Karhumäki,87)

- closed under composition

  (Chytil, Jakl, 77)

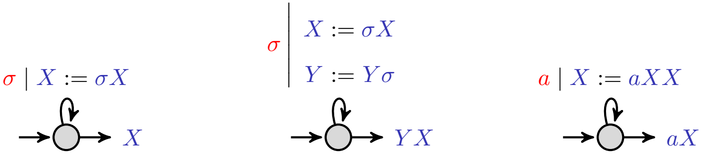
- equivalent to **reversible**[2] two-way transducers

  (Dartois,Fournier,Jecker,Lhote,17)

- and to many other models ...

[2] $\Sigma^* a \Sigma^*$ is not definable by any one-way reversible automaton

Transducers with registers



mirror *id.mirror* *exp:* $a^n \mapsto a^{2^n}$

- deterministic one-way
- equivalent to 2FT if linear updates

(Alur, Cerny, 10)

- decidable equivalence problem

(F., R.,17) (Benedikt et. al., 17)

(Courcelle) MSO Transducers

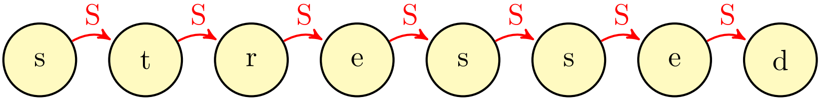
"interpreting the output structure in the input structure"

- output predicates defined by MSO formulas interpreted over the input structure

(Courcelle) MSO Transducers

"interpreting the output structure in the input structure"

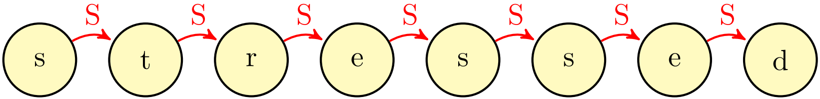
- output predicates defined by MSO formulas interpreted over the input structure



(Courcelle) MSO Transducers

"interpreting the output structure in the input structure"

- output predicates defined by MSO formulas interpreted over the input structure



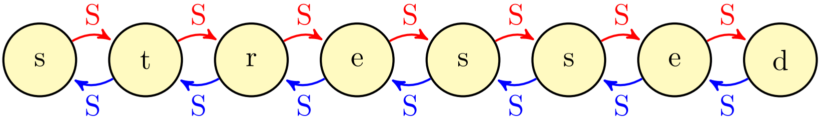
$$\phi_S(x, y) \equiv S(y, x)$$

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

(Courcelle) MSO Transducers

"interpreting the output structure in the input structure"

- output predicates defined by MSO formulas interpreted over the input structure



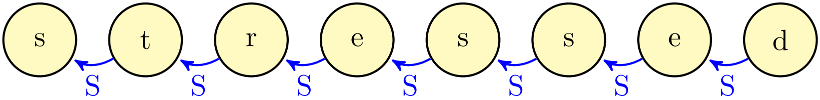
$$\phi_S(x, y) \equiv S(y, x)$$

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

(Courcelle) MSO Transducers

"interpreting the output structure in the input structure"

- output predicates defined by MSO formulas interpreted over the input structure



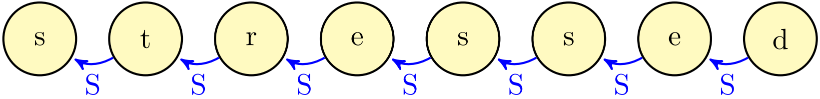
$$\phi_S(x,y) \equiv S(y,x)$$

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

(Courcelle) MSO Transducers

"interpreting the output structure in the input structure"

- output predicates defined by MSO formulas interpreted over the input structure

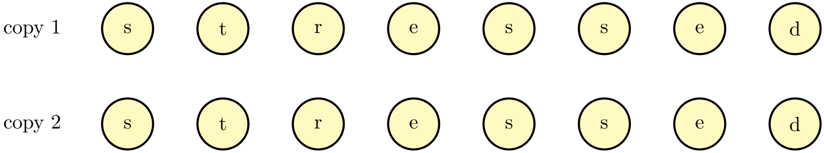


$$\phi_S(x, y) \equiv S(y, x)$$

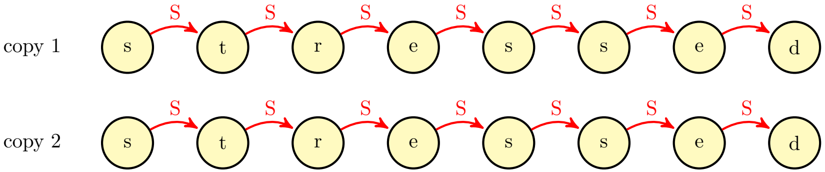
$$\phi_\sigma(x) \equiv \sigma(x)$$

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

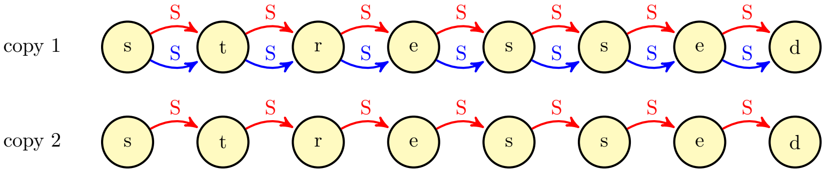
Other example : $u \mapsto u.\mathrm{mirror}(u)$



Other example : $u \mapsto u.\text{mirror}(u)$



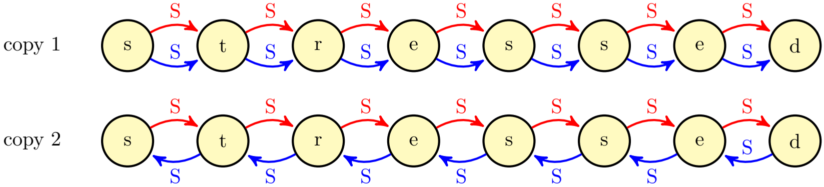
Other example : $u \mapsto u.\mathrm{mirror}(u)$



Formulas

copy 1: $\phi^{\mathbf{1}}_{S}(x,y) \quad \equiv \quad S(x,y)$

Other example : $u \mapsto u.\mathrm{mirror}(u)$

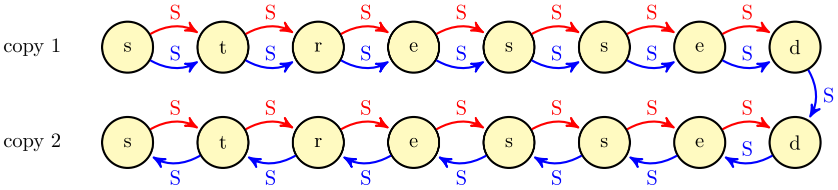


Formulas

$$\textbf{copy 1:} \qquad \phi^{\mathbf{1}}_{S}(x,y) \equiv S(x,y)$$

$$\textbf{copy 2:} \qquad \phi^{\mathbf{2}}_{S}(x,y) \equiv S(y,x)$$

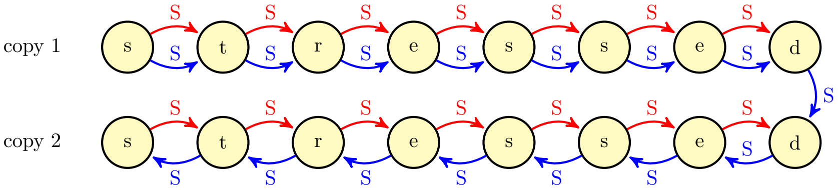
Other example : $u \mapsto u.\text{mirror}(u)$



Formulas

$$
\begin{array}{rcl}
\textbf{copy 1:} & \phi^{\mathbf{1}}_{S}(x,y) \equiv & S(x,y) \\
\textbf{copy 2:} & \phi^{\mathbf{2}}_{S}(x,y) \equiv & S(y,x) \\
\textbf{copy 1 to copy 2:} & \phi^{\mathbf{1}\to\mathbf{2}}_{S}(x,y) \equiv & x = y \wedge last(x)
\end{array}
$$

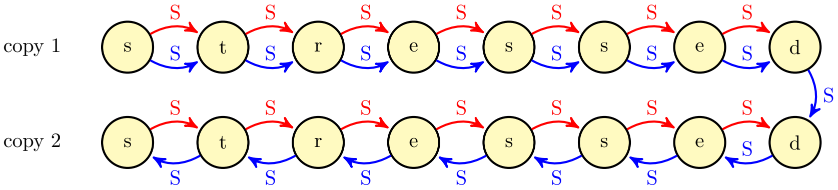
Other example : $u \mapsto u.\mathrm{mirror}(u)$



Formulas

$$
\begin{array}{rrcl}
\textbf{copy 1:} & \phi_S^{\mathbf{1}}(x,y) & \equiv & S(x,y) \\
\textbf{copy 2:} & \phi_S^{\mathbf{2}}(x,y) & \equiv & S(y,x) \\
\textbf{copy 1 to copy 2:} & \phi_S^{\mathbf{1}\to\mathbf{2}}(x,y) & \equiv & x = y \wedge last(x) \\
\textbf{copy 2 to copy 1:} & \phi_S^{\mathbf{2}\to\mathbf{1}}(x,y) & \equiv & \perp
\end{array}
$$

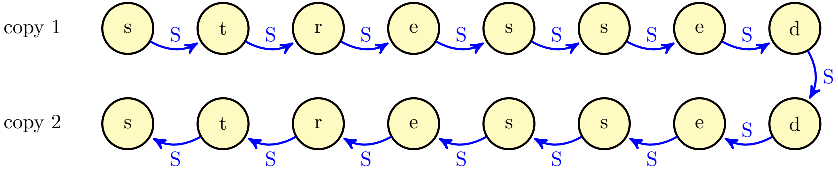
Other example : $u \mapsto u.\mathrm{mirror}(u)$



Formulas

$$\begin{array}{rrcl}
\textbf{copy 1:} & \phi_S^{\mathbf{1}}(x,y) & \equiv & S(x,y) \\
\textbf{copy 2:} & \phi_S^{\mathbf{2}}(x,y) & \equiv & S(y,x) \\
\textbf{copy 1 to copy 2:} & \phi_S^{\mathbf{1}\to\mathbf{2}}(x,y) & \equiv & x = y \wedge last(x) \\
\textbf{copy 2 to copy 1:} & \phi_S^{\mathbf{2}\to\mathbf{1}}(x,y) & \equiv & \perp \\
\textbf{for all copies } i\textbf{:} & \phi_\sigma^{\mathbf{i}}(x) & \equiv & \sigma(x)
\end{array}$$

Other example : $u \mapsto u.\mathrm{mirror}(u)$



Formulas

copy 1:	$\phi^{\mathbf{1}}_{S}(x,y)$	$\equiv$	$S(x,y)$
copy 2:	$\phi^{\mathbf{2}}_{S}(x,y)$	$\equiv$	$S(y,x)$
copy 1 to copy 2:	$\phi^{\mathbf{1}\to\mathbf{2}}_{S}(x,y)$	$\equiv$	$x = y \wedge last(x)$
copy 2 to copy 1:	$\phi^{\mathbf{2}\to\mathbf{1}}_{S}(x,y)$	$\equiv$	$\perp$
for all copies i:	$\phi^{\mathbf{i}}_{\sigma}(x)$	$\equiv$	$\sigma(x)$

Büchi Theorem for Regular Transductions

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

Theorem (Engelfriet, Hoogeboom, 01)

The following are equivalent:

1. *f is definable by a deterministic two-way transducer*
2. *f is MSO-definable.*

Büchi Theorem for Regular Transductions

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

Theorem (Engelfriet, Hoogeboom, 01)

The following are equivalent:

1. *f is definable by a deterministic two-way transducer*
2. *f is MSO-definable.*

Consequence Equivalence is decidable for MSO-transducers, and they are closed under composition.

Büchi Theorem for Regular Transductions

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

Theorem (Engelfriet, Hoogeboom, 01)

The following are equivalent:

1. *f is definable by a deterministic two-way transducer*
2. *f is MSO-definable.*

Consequence Equivalence is decidable for MSO-transducers, and they are closed under composition.

Proof ideas: MSO-transducers are 2-way transducers with MSO jumps $\phi_S^{c \to c'}(x, y)$

- turn jumps into walks
- hold enough information to decide MSO-formulas locally: states = MSO-types

$f = \hat{f} \circ f_{types}$ (use composition closure of 2-way trans)

Summary – Expressiveness

	input-deterministic		functional		non-deterministic
1-way (rational)		$<$		$<$	
	$\wedge$		$\wedge$		$\wedge$
2-way (regular)		$=$		$<$	

Summary – Expressiveness

	input-deterministic	functional	non-deterministic
1-way (rational)	$\leftarrow$ PTIME	$\leftarrow$ PTIME	
2-way (regular)		$\leftarrow$ PSPACE	

Summary – Expressiveness

	input-deterministic	functional	non-deterministic
1-way (rational)	PTime	Dec	PTime Undec
2-way (regular)			PSpace

(F., Gauwin, R., Servais, 13)

(Baschenis, Gauwin, Muscholl, Puppis,17)

Summary – Equivalence problem

$dom(T_1) = dom(T_2)$ is known.

	input-deterministic	functional	non-deterministic
1-way (rational)	PTIME	PTIME	undec
2-way (regular)	PSPACE	PSPACE	undec

Some other (recent) results

Other specification languages

- FO-transducers
  - equivalent to aperiodic transducers with registers (F., Krishna, Trivedi, 14)
  - and to aperiodic 2-way transducers (Dartois, Jecker, R., 16)

Other specification languages

- FO-transducers
  - equivalent to aperiodic transducers with registers (F., Krishna, Trivedi, 14)
  - and to aperiodic 2-way transducers (Dartois, Jecker, R., 16)
- regular function expressions
  - iterated sum $f^*(u) = f(u_1)f(u_2)\dots f(u_n)$ for $u = u_1 \dots u_n$
  - chain sum $f^c(u) = f(u_1u_2)f(u_2u_3)\dots f(u_{n-1}u_n)$
  - introduced by Alur, Freilich, Raghothaman in 14
  - direct construction from 2FT by Baudru, R. in 18
  - extended to infinite words by Dave, Gastin, Krishna in 18
  - non-deterministic 2FT by Choffrut, Guillon in 14

Other specification languages

- FO-transducers
  - equivalent to aperiodic transducers with registers (F., Krishna, Trivedi, 14)
  - and to aperiodic 2-way transducers (Dartois, Jecker, R., 16)
- regular function expressions
  - iterated sum $f^*(u) = f(u_1)f(u_2)\dots f(u_n)$ for $u = u_1 \dots u_n$
  - chain sum $f^c(u) = f(u_1u_2)f(u_2u_3)\dots f(u_{n-1}u_n)$
  - introduced by Alur, Freilich, Raghothaman in 14
  - direct construction from 2FT by Baudru, R. in 18
  - extended to infinite words by Dave, Gastin, Krishna in 18
  - non-deterministic 2FT by Choffrut, Guillon in 14
- functions on lists, equipped with function composition $\circ$ by Bojanczyk, Daviaud and Krishna in 18

Other specification languages

- FO-transducers
  - equivalent to aperiodic transducers with registers (F., Krishna, Trivedi, 14)
  - and to aperiodic 2-way transducers (Dartois, Jecker, R., 16)
- regular function expressions
  - iterated sum $f^*(u) = f(u_1)f(u_2)\dots f(u_n)$ for $u = u_1 \dots u_n$
  - chain sum $f^c(u) = f(u_1u_2)f(u_2u_3)\dots f(u_{n-1}u_n)$
  - introduced by Alur, Freilich, Raghothaman in 14
  - direct construction from 2FT by Baudru, R. in 18
  - extended to infinite words by Dave, Gastin, Krishna in 18
  - non-deterministic 2FT by Choffrut, Guillon in 14
- functions on lists, equipped with function composition $\circ$ by Bojanczyk, Daviaud and Krishna in 18
- an expressive decidable logic tailored to (non-functional) transductions Dartois, F., Lhote, 18

Definability Problems

Definition

$\mathcal{F}$: logical fragment of MSOT (e.g. FOT)

Input: T an MSOT

Output: Is $[\![T]\!]$ FO-definable ?

Definability Problems

Definition

$\mathcal{F}$: logical fragment of MSOT (e.g. FOT)
Input: T an MSOT
Output: Is $[\![T]\!]$ FO-definable ?

Results

- Decidable for "rational" MSOT (=rational functions)
  F., Gauwin, Lhote, 16
- Open for MSOT

Register Minimization Problems

Rational functions = Register transd. with updates $X := Y u$

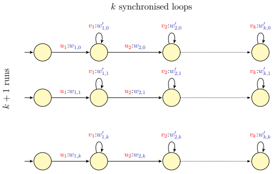
Register Minimization Problems

Rational functions = Register transd. with updates $X := Yu$

Theorem (Daviaud, R., Talbot, 16)

A transducer T can be expressed using k registers iff it satisfies the Twinning Property of order k.

For all situations like:



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

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

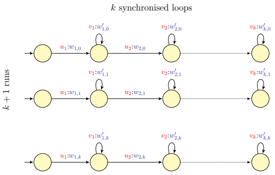
Register Minimization Problems

Rational functions = Register transd. with updates $X := Yu$

Theorem (Daviaud, R., Talbot, 16)

A transducer T can be expressed using k registers iff it satisfies the Twinning Property of order k.

For all situations like:



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

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

Other results:

- multi-sequential transducers Daviaud, Jecker, R., Villevalois, 17
- concatenation-free non-det reg. transducers Baschenis, Gauwin, Muscholl, Puppis, 16
- concatenation-free **det.** reg transducers R., Villevalois, 19

Origin semantics (Bojanczyk, 14)

$$a\ a\ a\ a \mapsto a\ a\ a \qquad \neq \qquad a\ a\ a\ a \mapsto a\ a\ a$$

Origin semantics $[\![T]\!]_o$ inherent to most transducer models T !

Origin semantics (Bojanczyk, 14)

$$a\ a\ a\ a \mapsto a\ a\ a \quad \neq \quad a\ a\ a\ a \mapsto a\ a\ a$$

Origin semantics $[\![T]\!]_o$ inherent to most transducer models T !

- existence of a canonical transducer if *origin* is taken into account (Bojanczyk, 14)
- decidable FO-definability of MSOT *with* origin

Origin semantics (Bojanczyk, 14)

$$a \; a \; a \; a \mapsto a \; a \; a \qquad \neq \qquad a \; a \; a \; a \mapsto a \; a \; a$$

Origin semantics $[\![T]\!]_o$ inherent to most transducer models T !

- existence of a canonical transducer if *origin* is taken into account (Bojanczyk, 14)
- decidable FO-definability of MSOT *with* origin
- algorithmic problems modulo origin ($[\![T_1]\!]_o = [\![T_2]\!]_o$)
- extended to "similar" origins through resynchronisers (F., Maneth, R., Talbot, 15) (F., Jecker, Löding, Winter,16), (Bose, Muscholl, Penelle, Puppis, 18)

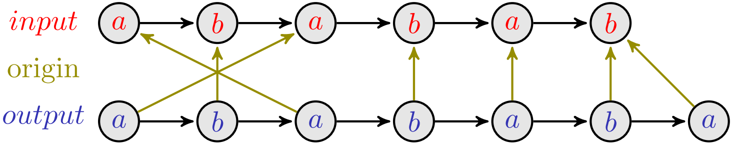
Origin semantics (Bojanczyk, 14)

$$a\ a\ a\ a \mapsto a\ a\ a \qquad \neq \qquad a\ a\ a\ a \mapsto a\ a\ a$$

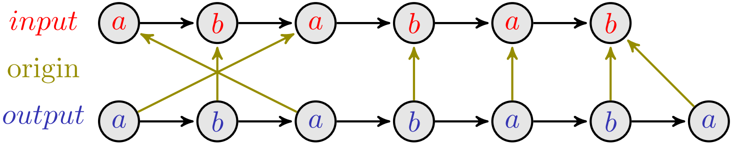
Origin semantics $[\![T]\!]_o$ inherent to most transducer models T !

- existence of a canonical transducer if *origin* is taken into account (Bojanczyk, 14)
- decidable FO-definability of MSOT *with* origin
- algorithmic problems modulo origin ($[\![T_1]\!]_o = [\![T_2]\!]_o$)
- extended to ”similar” origins through resynchronisers (F., Maneth, R., Talbot, 15) (F., Jecker, Löding, Winter,16), (Bose, Muscholl, Penelle, Puppis, 18)
- study of rational relation subclasses by control languages $REL(C)$, $C \subseteq \{\mathrm{in}, \mathrm{out}\}^*$ (Descotte, Figueira, Libkin, Puppis)

An expressive origin-based logic tailored to transductions (Dartois, F., Lhote, 18)

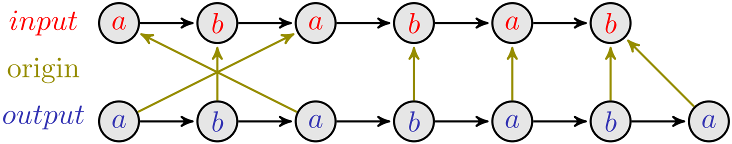


An expressive origin-based logic tailored to transductions (Dartois, F., Lhote, 18)



transduction ≈ set of origin graphs

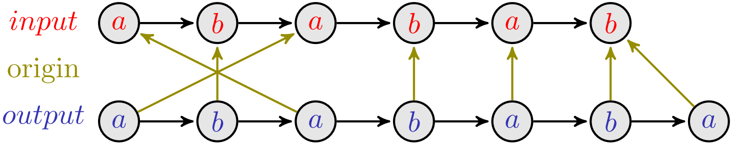
An expressive origin-based logic tailored to transductions (Dartois, F., Lhote, 18)



transduction ≈ set of origin graphs

$$\mathrm{MSO}[\leq_{in}, \leq_{out}, o]$$

An expressive origin-based logic tailored to transductions (Dartois, F., Lhote, 18)



transduction ≈ set of origin graphs

$$\mathrm{MSO}[\leq_{in}, \leq_{out}, o]$$

Results

- $T \models \phi$ decidable for 2-way transducers
- undecidable satisfiability
- decidable fragment with regular synthesis
- correspondence with data words

Some Other Results

- machine-independent characterisations
  Cadilhac, Krebs, Ludwig, Paperman, 15

Some Other Results

- machine-independent characterisations

  Cadilhac, Krebs, Ludwig, Paperman, 15
- uniformisation problems (Ismaël's Jecker and Sarah Winter's PhD thesis). E.g. given R rational, is there f sequential such that
  - $f \subseteq R$
  - $dom(f) = dom(R)$

Some Other Results

- machine-independent characterisations

  Cadilhac, Krebs, Ludwig, Paperman, 15

- uniformisation problems (Ismaël's Jecker and Sarah Winter's PhD thesis). E.g. given R rational, is there f sequential such that
  - $f \subseteq R$
  - $dom(f) = dom(R)$

- learning

  Boiret, Lemay, Niehren 12

Some Other Results

- machine-independent characterisations

  Cadilhac, Krebs, Ludwig, Paperman, 15
- uniformisation problems (Ismaël's Jecker and Sarah Winter's PhD thesis). E.g. given R rational, is there f sequential such that
  - $f \subseteq R$
  - $dom(f) = dom(R)$
- learning

  Boiret, Lemay, Niehren 12
- data word transducers Léo Exibard's PhD thesis

Some Other Results

- machine-independent characterisations

  Cadilhac, Krebs, Ludwig, Paperman, 15
- uniformisation problems (Ismaël's Jecker and Sarah Winter's PhD thesis). E.g. given R rational, is there f sequential such that
  - $f \subseteq R$
  - $dom(f) = dom(R)$
- learning

  Boiret, Lemay, Niehren 12
- data word transducers Léo Exibard's PhD thesis
- other structures: infinite strings, nested words, trees, graphs, data words ...

A Few Applications

- language and speech processing (M. Mohri)
- regular model-checking
- text analysis, document transformation
- reactive synthesis
- **Tools**: OpenFST, Vaucanson, DreX (Alur, d'Antoni, Raghothaman)
- line of works on symbolic transducers (d'Antoni, Veanes ...)

Thanks!

Thanks for your attention!

Announcements

RP'19, September 11-13, Brussels

- conference on reachability problems
- talks with submitted papers or w/o
- best papers invited for a journal issue
- deadline in June
- invited speakers: Henzinger, Protasov, Lasota, Sriram S., Raskin.

Announcements

RP'19, September 11-13, Brussels

- conference on reachability problems
- talks with submitted papers or w/o
- best papers invited for a journal issue
- deadline in June
- invited speakers: Henzinger, Protasov, Lasota, Sriram S., Raskin.

1 PostDoc position at ULB

- transducer and synthesis problems
- up to 2 years
- flexible starting date