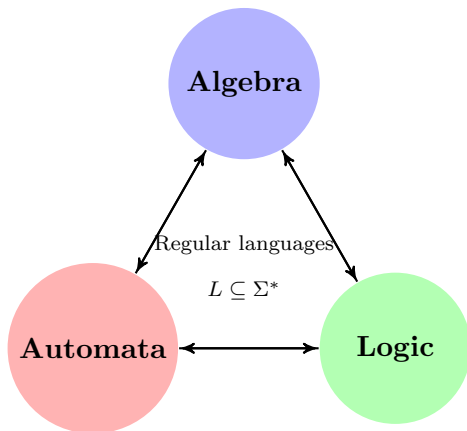


Automata and Logic for (Finite) Word Transductions

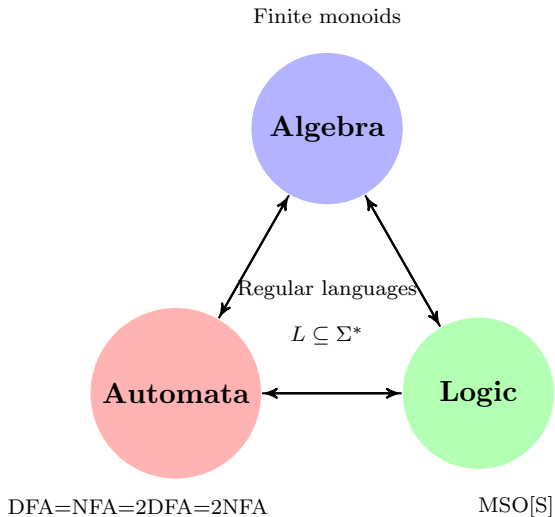
Emmanuel Filiot

Université libre de Bruxelles & FNRS

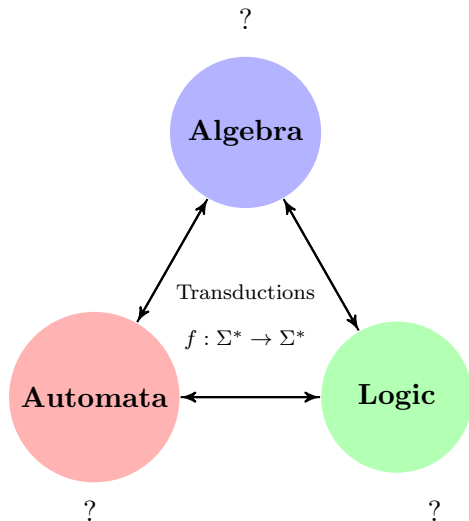
Trinity for Regular Languages



Trinity for Regular Languages

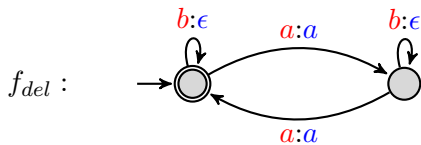


Objective of the talk

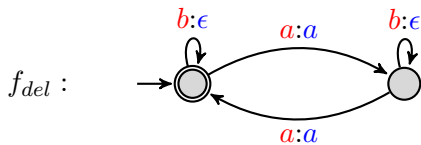


Automata models for transductions

Automata for transductions: transducers

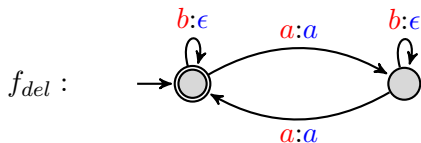


Automata for transductions: transducers



$aabaa \mapsto aaaa$

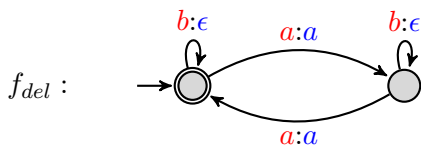
Automata for transductions: transducers



$aabaa \mapsto aaaa$

$aaba \mapsto \text{undefined}$

Automata for transductions: transducers



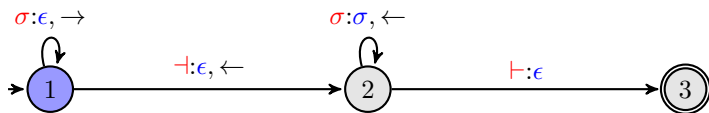
$aabaa \mapsto aaaa$

$aaba \mapsto \text{undefined}$

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

Two-way transducers

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

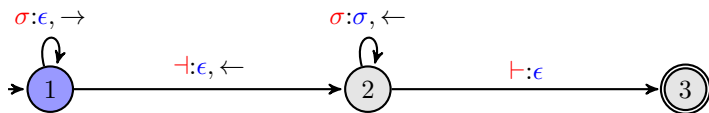


output



Two-way transducers

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

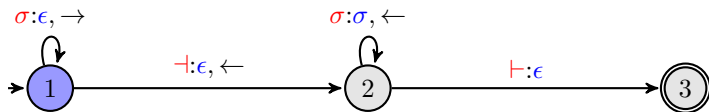


output



Two-way transducers

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

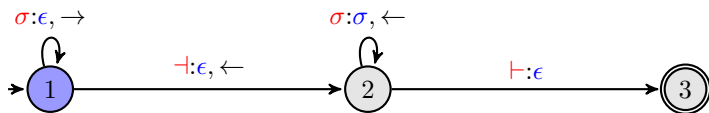


output



Two-way transducers

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

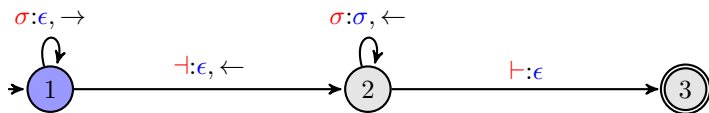


output



Two-way transducers

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

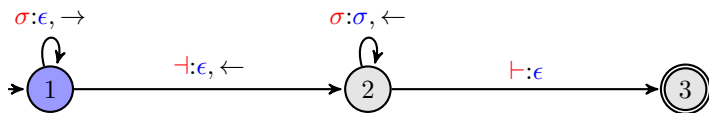


output



Two-way transducers

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

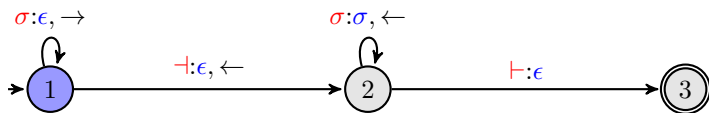


output



Two-way transducers

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

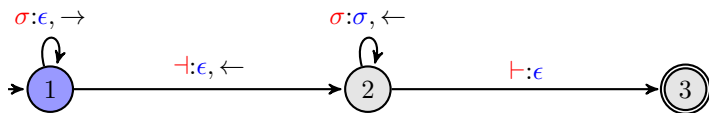


output



Two-way transducers

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

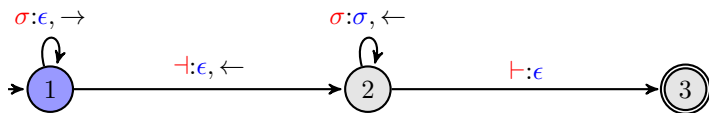


output



Two-way transducers

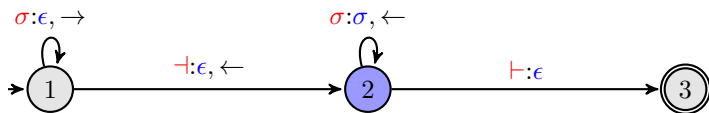
input \vdash *s t r e s s e d* \vdash
▲



output
▲

Two-way transducers

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

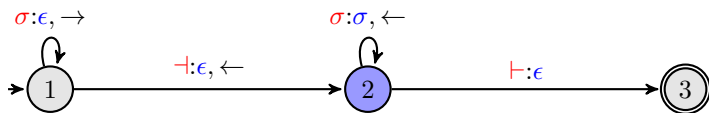


output



Two-way transducers

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



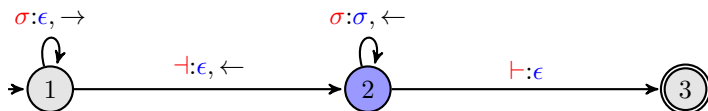
output

d



Two-way transducers

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



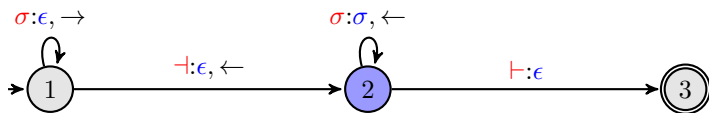
output *d* *e*



Two-way transducers

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

▲

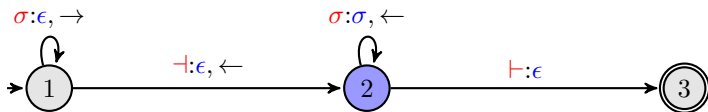


output d e s

▲

Two-way transducers

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

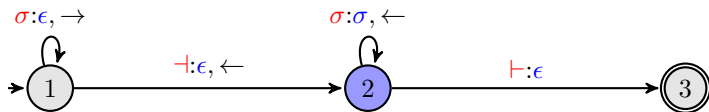


output d e s s



Two-way transducers

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



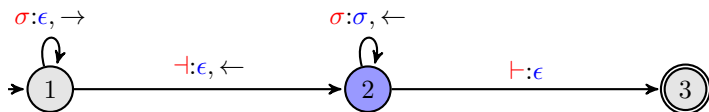
output

d e s s e



Two-way transducers

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

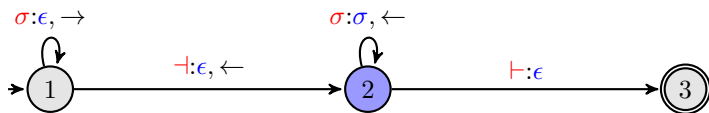


output d e s s e r



Two-way transducers

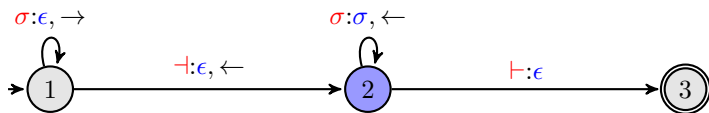
input \vdash *s t r e s s e d* \vdash
▲



output *d e s s e r t*
▲

Two-way transducers

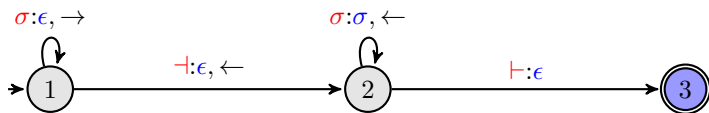
input \vdash s t r e s s e d \vdash
 \blacktriangle



output d e s s e r t s
 \blacktriangle

Two-way transducers

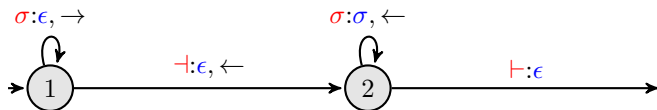
input \vdash s t r e s s e d \vdash
 \blacktriangle



output d e s s e r t s
 \blacktriangle

Two-way transducers

input s t r e s s e d ⊥



output d e s s e r t s

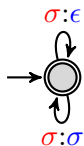
▲

one-way < two-way

- ☺ decidable equivalence problem (Culik, Karhumaki, 87).
- ☺ closed under composition \circ (Chytil, Jakl, 77) (Dartois, Fournier, Jecker, Lhote, 17)

Non-determinism and relations

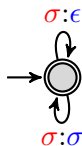
In general, transducers define binary *relations* in $\Sigma^* \times \Sigma^*$



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

Non-determinism and relations

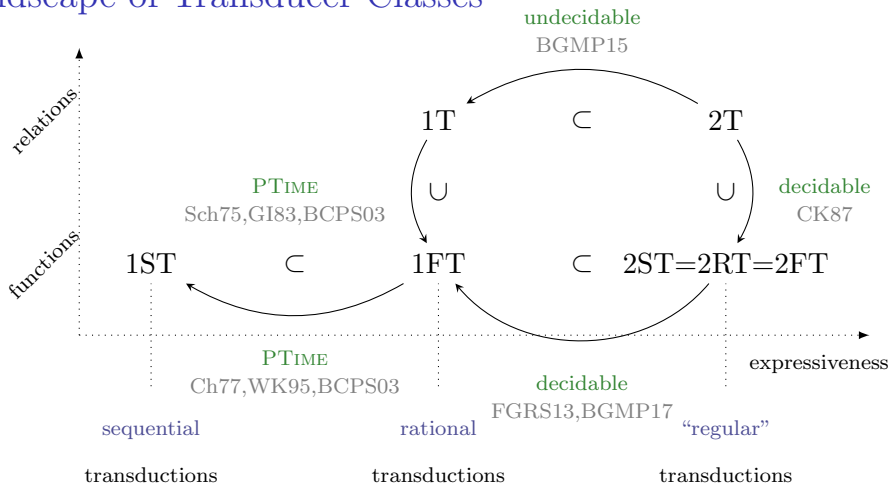
In general, transducers define binary *relations* in $\Sigma^* \times \Sigma^*$



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

- ▶ undecidable equivalence/inclusion problems for non-det 1-way transducers
- ▶ sequential transducer $=_{def}$ input-deterministic transducer

Landscape of Transducer Classes⁴



⁴**Sch**:Schutzenberger; **GI**:Gurari, Ibarra; **BCPS**:Beal, Carton, Prieur, Sakarovitch; **Ch**:Choffrut, **WK**:Weber, Klemm; **FGRS**., Gauwin, Reynier, Servais; **BGMP**:Baschenis, Gauwin, Muscholl, Puppis, **CK**:Culik, Karhumaki

Logic for transductions

(Courcelle) MSO Transducers

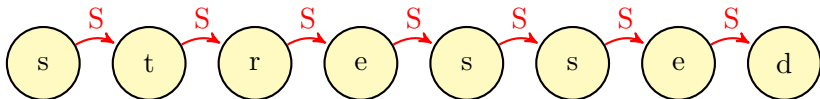
“interpreting the output structure in the input structure”

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

(Courcelle) MSO Transducers

“interpreting the output structure in the input structure”

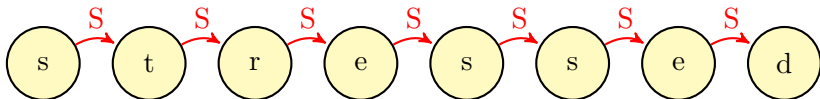
- ▶ output predicates defined by $\text{MSO}[S]$ formulas interpreted over the input structure



(Courcelle) MSO Transducers

“interpreting the output structure in the input structure”

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



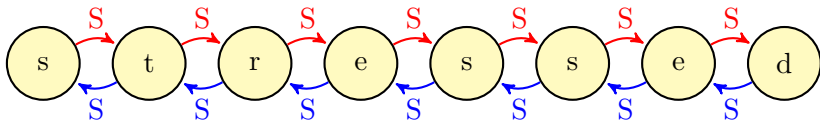
$$\phi_S(x, y) \equiv S(y, x)$$

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

(Courcelle) MSO Transducers

“interpreting the output structure in the input structure”

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



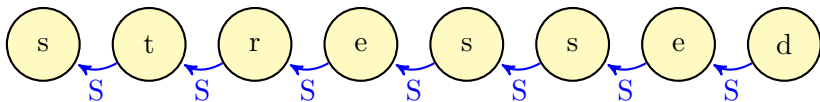
$$\phi_S(x, y) \equiv S(y, x)$$

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

(Courcelle) MSO Transducers

“interpreting the output structure in the input structure”

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



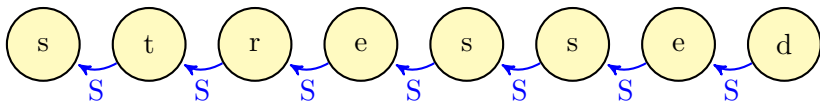
$$\phi_S(x, y) \equiv S(y, x)$$

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

(Courcelle) MSO Transducers

“interpreting the output structure in the input structure”

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

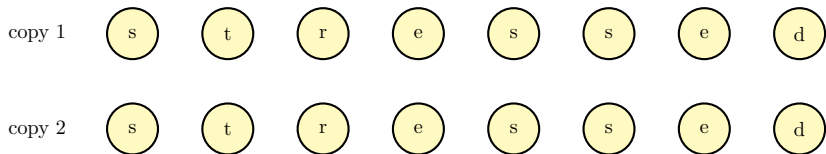


$$\phi_S(x, y) \equiv S(y, x)$$

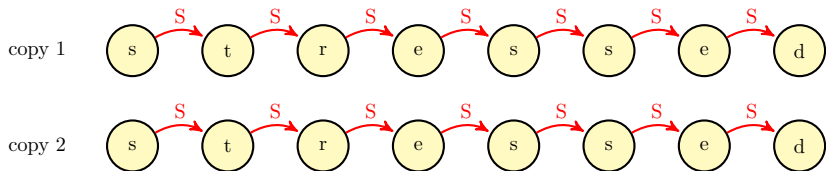
$$\phi_\sigma(x) \equiv \sigma(x)$$

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

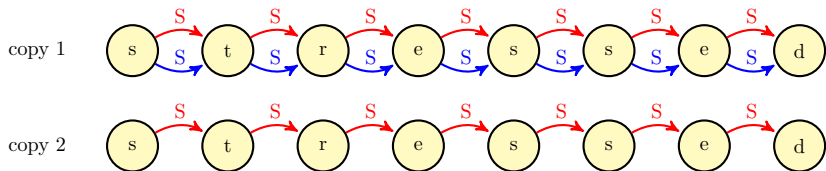
Other example : $u \mapsto u.mirror(u)$



Other example : $u \mapsto u.mirror(u)$



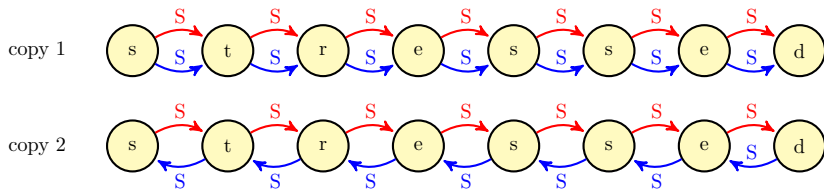
Other example : $u \mapsto u.mirror(u)$



Formulas

copy 1: $\phi_S^1(x, y) \equiv S(x, y)$

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

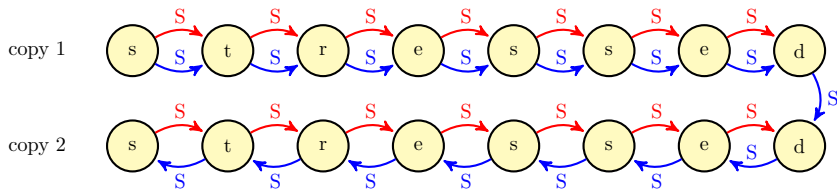


Formulas

copy 1: $\phi_S^1(x, y) \equiv S(x, y)$

copy 2: $\phi_S^2(x, y) \equiv S(y, x)$

Other example : $u \mapsto u.mirror(u)$



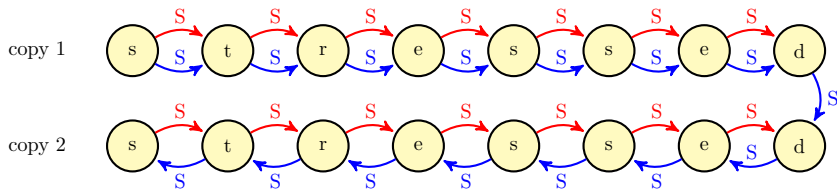
Formulas

$$\text{copy 1: } \phi_S^1(x, y) \equiv S(x, y)$$

$$\text{copy 2: } \phi_S^2(x, y) \equiv S(y, x)$$

$$\text{copy 1 to copy 2: } \phi_S^{1 \rightarrow 2}(x, y) \equiv x = y \wedge last(x)$$

Other example : $u \mapsto u.mirror(u)$



Formulas

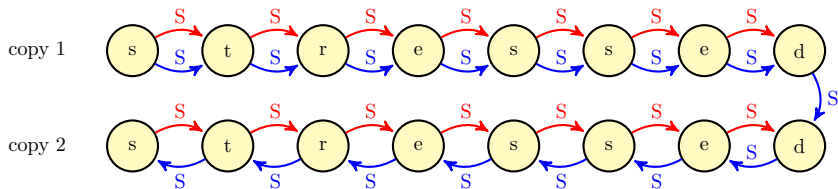
$$\text{copy 1: } \phi_S^1(x, y) \equiv S(x, y)$$

$$\text{copy 2: } \phi_S^2(x, y) \equiv S(y, x)$$

$$\text{copy 1 to copy 2: } \phi_S^{1 \rightarrow 2}(x, y) \equiv x = y \wedge \text{last}(x)$$

$$\text{copy 2 to copy 1: } \phi_S^{2 \rightarrow 1}(x, y) \equiv \perp$$

Other example : $u \mapsto u.mirror(u)$



Formulas

$$\text{copy 1: } \phi_S^1(x, y) \equiv S(x, y)$$

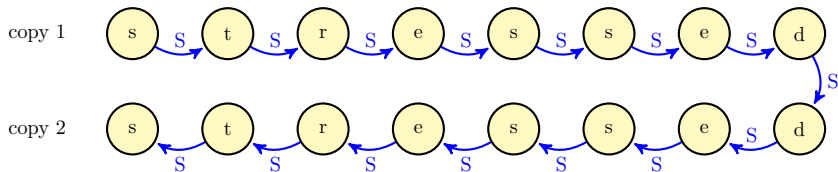
$$\text{copy 2: } \phi_S^2(x, y) \equiv S(y, x)$$

$$\text{copy 1 to copy 2: } \phi_S^{1 \rightarrow 2}(x, y) \equiv x = y \wedge last(x)$$

$$\text{copy 2 to copy 1: } \phi_S^{2 \rightarrow 1}(x, y) \equiv \perp$$

$$\text{for all copies } i: \phi_\sigma^i(x) \equiv \sigma(x)$$

Other example : $u \mapsto u.mirror(u)$



Formulas

$$\text{copy 1: } \phi_S^1(x, y) \equiv S(x, y)$$

$$\text{copy 2: } \phi_S^2(x, y) \equiv S(y, x)$$

$$\text{copy 1 to copy 2: } \phi_S^{1 \rightarrow 2}(x, y) \equiv x = y \wedge last(x)$$

$$\text{copy 2 to copy 1: } \phi_S^{2 \rightarrow 1}(x, y) \equiv \perp$$

$$\text{for all copies } i: \phi_\sigma^i(x) \equiv \sigma(x)$$

Büchi Theorems for Word Transductions

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

Theorem (Engelfriet, Hoogeboom, 01)

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

Büchi Theorems for Word Transductions

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

Theorem (Engelfriet, Hoogeboom, 01)

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

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

A DECIDABLE LOGIC FOR TRANSDUCTIONS OF FINITE WORDS

Joint with

Luc

Dartois

ULB

Nathan

Lhote

ULB - LABRI

MOTIVATION

MODEL-CHECKING :

IMPLEMENTATION \models SPECIFICATION ?

↑

↑

A functional transduction

A transduction

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

$$\mathcal{R} \subseteq \Sigma^* \times \Sigma^*$$

$$f \models \mathcal{R} \quad \text{if} \quad \forall u \in \text{dom}(f), (u, f(u)) \in \mathcal{R}$$

MOTIVATION

MODEL-CHECKING :

IMPLEMENTATION \models SPECIFICATION ?

↑

↑

A functional transduction

A transduction

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

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

$$f \models R \quad \text{if} \quad \forall u \in \text{dom}(f), (u, f(u)) \in R$$

SYNTHESIS :

?

\models SPECIFICATION

$$\exists f. f \models R \wedge \text{dom}(f) = \text{dom}(R) ?$$

EXAMPLES OF SPECIFICATIONS

- True: $\Sigma^*_x \Sigma^*$

EXAMPLES OF SPECIFICATIONS

• True: $\Sigma^*_x \Sigma^*$

• There exists an 'a' in the output $\Sigma^*_x \Sigma^*_a \Sigma^*$

EXAMPLES OF SPECIFICATIONS

• True: $\Sigma^* x \Sigma^*$

• There exists an 'a' in the output $\Sigma^* x \Sigma_a^* \Sigma^*$

• Output is a subword of input

$$\{ (u, v) \mid \exists i_1 < \dots < i_{|v|} \cdot v = u[i_1] \dots u[i_{|v|}] \}$$

EXAMPLES OF SPECIFICATIONS

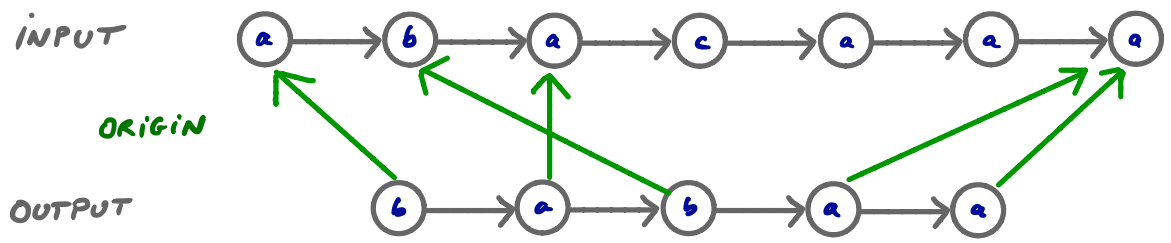
- True: $\Sigma^* \times \Sigma^*$
- There exists an 'a' in the output $\Sigma^* \times \Sigma_a^* \Sigma^*$
- Output is a subword of input
 $\{ (u, v) \mid \exists i_1 < \dots < i_{|v|} \cdot v = u[i_1] \dots u[i_{|v|}] \}$
- Every request is processed exactly once
 $\{ (r_1 \dots r_k, g_{\pi(1)} \dots g_{\pi(k)}) \mid \pi \text{ permutation} \}$

EXAMPLES OF SPECIFICATIONS

- True: $\Sigma^* \times \Sigma^*$
- There exists an 'a' in the output $\Sigma^* \times \Sigma_a^* \Sigma^*$
- Output is a subword of input
 $\{ (u, v) \mid \exists i_1 < \dots < i_{|v|} \cdot v = u[i_1] \dots u[i_{|v|}] \}$
- Every request is processed exactly once
 $\{ (r_1 \dots r_k, g_{\pi(1)} \dots g_{\pi(k)}) \mid \pi \text{ permutation} \}$
- More generally: shuffle
 $\{ (\sigma_1 \dots \sigma_k, \sigma_{\pi(1)} \dots \sigma_{\pi(k)}) \mid \pi \text{ permutation} \}$

TOWARDS A LOGIC FOR TRANSDUCTIONS

IDEA : SEE TRANSDUCTIONS AS SINGLE STRUCTURES WITH ORIGIN
(called origin graphs)

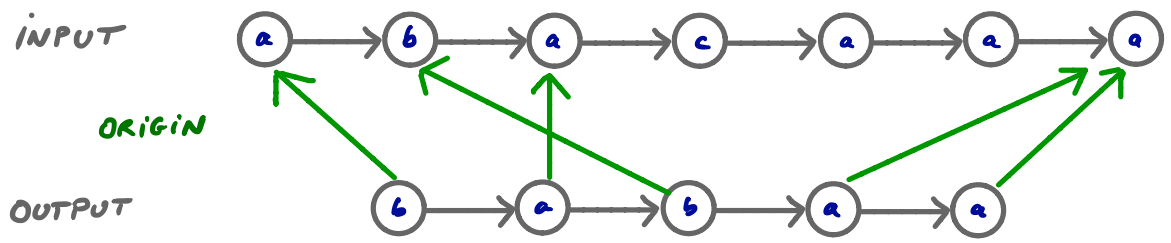


$$\text{MSO}[\leq_{\text{in}}, \leq_{\text{out}}, \theta]$$

↑ ↑ ↑
input order output order origin function

TOWARDS A LOGIC FOR TRANSDUCTIONS

IDEA : SEE TRANSDUCTIONS AS SINGLE STRUCTURES WITH ORIGIN (called origin graphs)



$$\text{MSO}[\leq_{\text{in}}, \leq_{\text{out}}, \theta]$$

input order output order origin function

Origin information [Bojańczyk, '14]

Characterizations of classes of origin graphs [Bojańczyk, Daviaud, Guillon, Penelle '17]

EXAMPLES

- True : T

EXAMPLES

• True : T

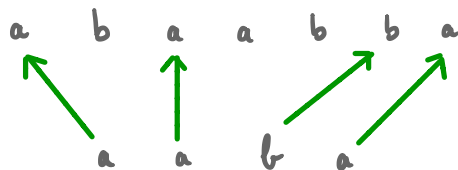
• Output contains 'a' : $\exists^{out} x. a(x)$ ($\exists^{out} x =_{def} \exists x. x \leq_{out} x$)

EXAMPLES

• True : T

• Output contains 'a' : $\exists^{out} x. a(x)$

• Output is a subword of input :



$$\underbrace{\forall^{out} x \forall^{out} y. x \leq_{out} y \rightarrow \sigma(x) \leq_{in} \sigma(y)}_{\sigma \text{ is order-preserving}}$$

$$\underbrace{\forall^{out} x. \bigwedge_{\sigma \in \Sigma} \sigma(x) \rightarrow \sigma(\sigma(x))}_{\sigma \text{ is label-preserving}}$$

$$\underbrace{\forall^{out} x \forall^{out} y. x \neq y \rightarrow \sigma(x) \neq \sigma(y)}_{\sigma \text{ is injective}}$$

EXAMPLES

- True : T
- Output contains 'a' : $\exists^{out} x. a(x)$
- Output is a subword of input :
- shuffle : $bijection(\sigma) \wedge label-pres(\sigma)$

EXAMPLES

- True : T
- Output contains 'a' : $\exists^{out} x. a(x)$
- Output is a subword of input :
- shuffle : bijection (σ) \wedge label-pres (σ)
- identity : shuffle \wedge order-preserving (σ)

MODEL-CHECKING

$T \models \varphi$ is decidable for $T \in 2DFT$ and $\varphi \in MSOL[\leq_{in}, \leq_{out}, \emptyset]$

Bojańczyk, Daviaud, Guillon, Penelle '17

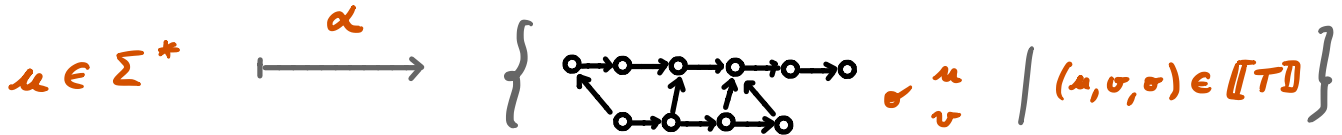
MODEL-CHECKING

$T \neq \varphi$ is decidable for $T \in \text{2DFT}$ and $\varphi \in \text{MSOL}[\leq_{in}, \leq_{out}, \theta]$

Bojańczyk, Dawid, Guillon, Penelle '17

WORD

SET OF ORIGIN GRAPHS



- ① $\alpha \in$ Courcelle's word-to-graph transductions
- ② Backward Translation Theorem: $\alpha^{-1}(\llbracket \neg \varphi \rrbracket) \in \text{REG}(\Sigma)$
- ③ Check $\alpha^{-1}(\llbracket \neg \varphi \rrbracket) = \emptyset$

SATISFIABILITY

Input : $\varphi \in \text{MSO}[\leq_{in}, \leq_{out}, \sigma]$ Output : $\exists G \cdot G \models \varphi ?$

RESULTS

- Undecidable (even for $\text{FO}_2[\leq_{out}, S_{out}, \leq_{in}, \sigma]$)
- Decidable for $\mathcal{L}_T := \text{FO}_2[\leq_{out}, \emptyset, \text{MSO}_{bin}[\leq_{in}]]$

Example : $\forall x^{\text{out}} \exists y^{\text{out}} P(\sigma(x), \sigma(y)) \rightarrow \forall x^{\text{out}} y^{\text{out}} x \succ_{\text{out}} y \rightarrow Q(\sigma(y), \sigma(x))$
 $P, Q \in \text{MSO}[\leq_{in}]$

SATISFIABILITY

Input : $\varphi \in \text{MSO}[\leq_{in}, \leq_{out}, \sigma]$

Output : $\exists G \cdot G \models \varphi ?$

RESULTS

- Undecidable (even for $\text{FO}_2[\leq_{out}, \leq_{out}, \leq_{in}, \sigma]$)
- Decidable for $\mathcal{L}_T := \text{FO}_2[\leq_{out}, \sigma, \text{MSO}_{bin}[\leq_{in}]]$

Example : $\forall x^{\text{out}} \exists y^{\text{out}} P(\sigma(x), \sigma(y)) \rightarrow \forall x^{\text{out}} y^{\text{out}} x \geq_{\text{out}} y \rightarrow Q(\sigma(y), \sigma(x))$
 $P, Q \in \text{MSO}[\leq_{in}]$

CONSEQUENCES

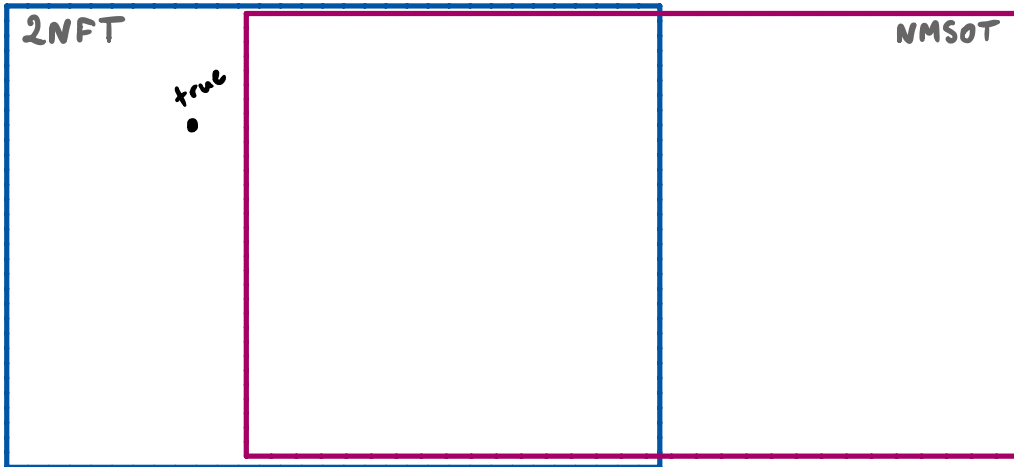
- Decidable equivalence problem for \mathcal{L}_T -transductions
- Decidable satisfiability for $\exists \mathcal{L}_T$

\uparrow
 $\exists x_1 \dots \exists x_n \cdot \mathcal{L}_T$

EXPRESSIVENESS OF \mathcal{L}_T

$$\mathcal{L}_T := FO_2[\leq_{out}, \theta, MSO_{bin}[\leq_{in}]]$$

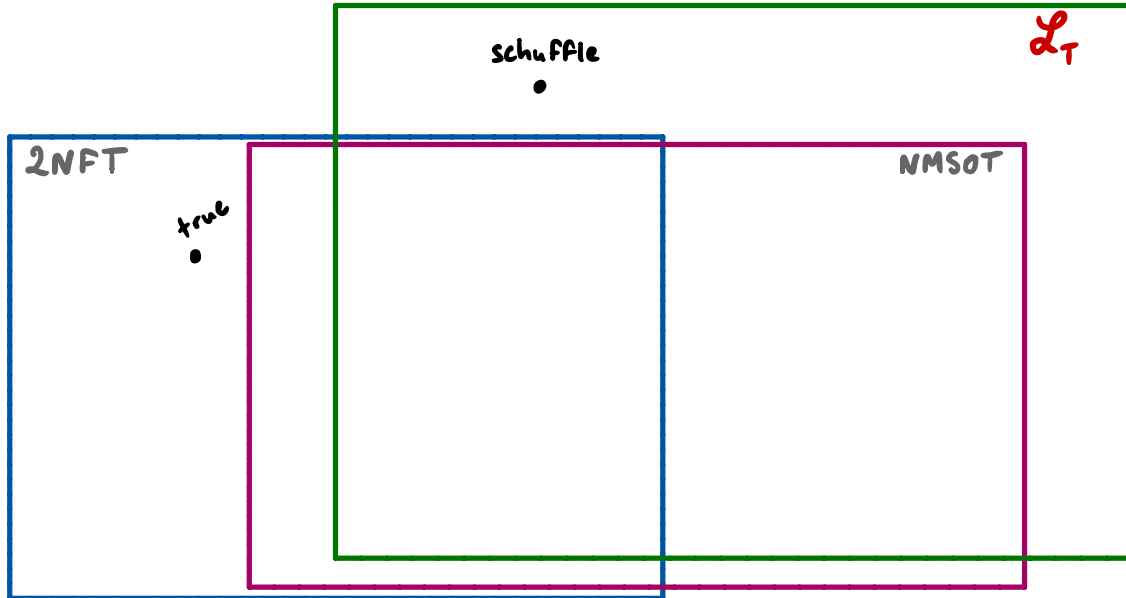
- all previous examples (including shuffle)
- $\mathcal{L}_T > 2DFT = MSOT$



EXPRESSIVENESS OF \mathcal{L}_T

$$\mathcal{L}_T := FO_2[\leq_{out}, \theta, MSO_{bin}[\leq_{in}]]$$

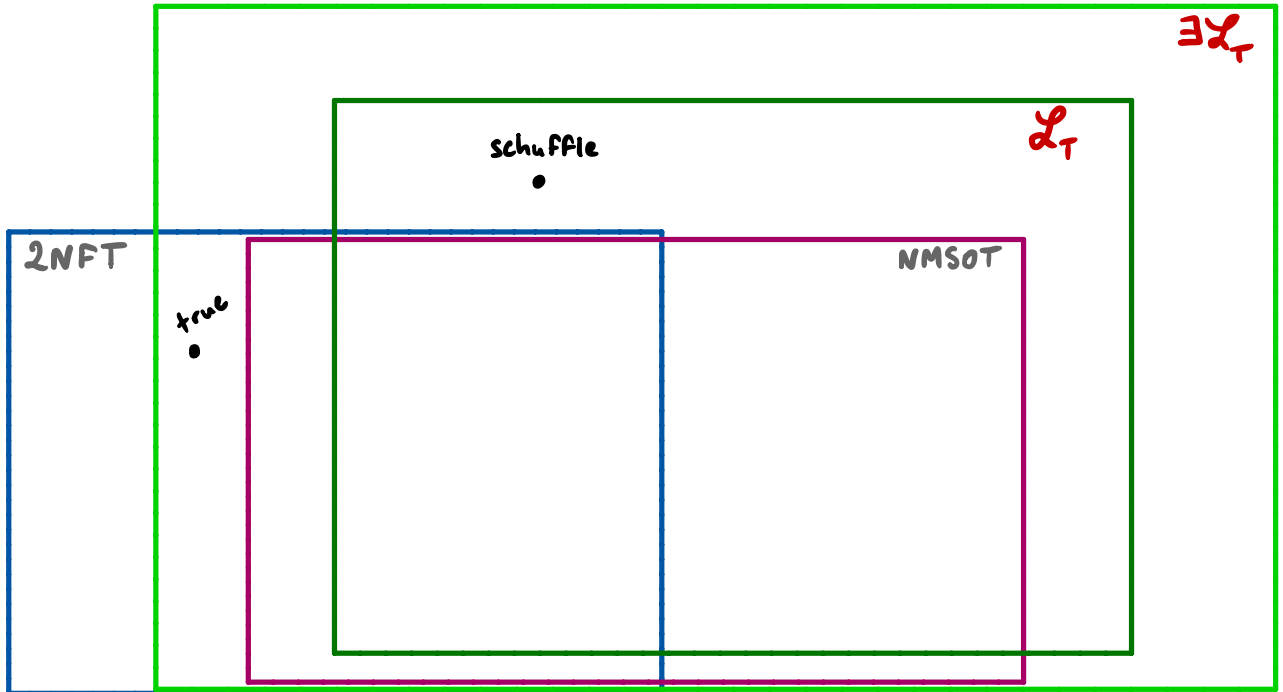
- all previous examples (including shuffle)
- $\mathcal{L}_T > 2DFT = MSOT$



EXPRESSIVENESS OF \mathcal{L}_T

$$\mathcal{L}_T := FO_2[\leq_{out}, \theta, MSO_{bin}[\leq_{in}]]$$

- all previous examples (including shuffle)
- $\mathcal{L}_T > 2DFT = MSOT$



SYNTHESIS

THEOREM

$\forall \varphi \in \exists \mathcal{L}_T. \exists f: \Sigma^* \rightarrow \Sigma^* .$

1. $f \in \text{MSOT}$
2. $\text{dom } f = \text{dom } \varphi$
3. $f \models \varphi$

SYNTHESIS

THEOREM

$$\forall \varphi \in \mathcal{L}_T. \exists f: \Sigma^* \rightarrow \Sigma^* . \begin{cases} 1. & f \in \text{MSOT} \\ 2. & \text{dom } f = \text{dom } \varphi \\ 3. & f \models \varphi \end{cases}$$

CONSEQUENCES

- New characterization of MSOT : $\text{MSOT} = \mathcal{L}_T \cap \text{functions}$
- Decidable functionality problem for \mathcal{L}_T

1. $\varphi: \mathcal{L}_T \xrightarrow{\text{synthesis}} \psi: \text{MSOT} \xrightarrow{\text{expressiveness}} \varphi': \mathcal{L}_T \cap \text{functions}$
2. Test $\varphi \equiv \varphi'$

DATA WORDS (OVER Σ, D)

DEFINITION

$$(\sigma_1, d_1) \dots (\sigma_n, d_n) \in (\Sigma \times D)^*$$

finite
alphabet

infinite set of
ordered data

DATA WORDS (OVER Σ, D)

DEFINITION

$$(\sigma_1, d_1) \dots (\sigma_n, d_n) \in (\Sigma \times D)^*$$

finite
alphabet

infinite set of
ordered data

DATA WORD AS FINITE STRUCTURE

finite word structure + data comparison

$\{1, \dots, n\}$: positions

\leq : linear-order

\preceq : preorder on positions

DATA WORDS (OVER Σ, D)

DEFINITION

$$(\sigma_1, d_1) \dots (\sigma_n, d_n) \in (\Sigma \times D)^*$$

finite
alphabet

infinite set of
ordered data

DATA WORD AS FINITE STRUCTURE

finite word structure + data comparison

$\{1, \dots, n\}$: positions

\preceq : preorder on positions

\leq : linear-order

LOGICS FOR DATA WORDS

$FO_2[\leq, S_p, \sim_d]$: decidable

$FO_2[\leq, S_d, \preceq]$: decidable

$FO_2[\leq, S_p, \preceq]$: undecidable

[Schwentick, Zeume, '10]

Bojanczyk
Muscholl
Schwentick '06
Legoufin
David

TYPED DATA WORDS (over $\Sigma, \mathcal{D}, \mathcal{T}$)

DEFINITION

$$(w, \tau : \mathcal{D} \rightarrow \mathcal{T})$$

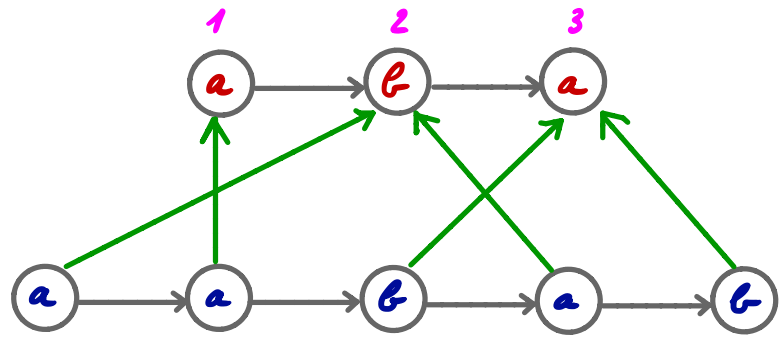
↑
data
word

↑
finite set
of types

closed if $\text{data}(w) = \downarrow \text{data}(w)$

$$\begin{pmatrix} d_1 \\ \tau_1 \end{pmatrix} \begin{pmatrix} d_2 \\ \tau_2 \end{pmatrix} \begin{pmatrix} d_3 \\ \tau_3 \end{pmatrix} \begin{pmatrix} d_4 \\ \tau_4 \end{pmatrix} \begin{pmatrix} d_5 \\ \tau_5 \end{pmatrix} \begin{pmatrix} d_6 \\ \tau_6 \end{pmatrix} \dots \begin{pmatrix} d_n \\ \tau_n \end{pmatrix}$$

TYPED DATA WORDS vs TRANSDUCTIONS



data = origin !

$\begin{pmatrix} 2 \\ b \\ a \end{pmatrix}$

$\begin{pmatrix} 1 \\ a \\ a \end{pmatrix}$

$\begin{pmatrix} 3 \\ a \\ b \end{pmatrix}$

$\begin{pmatrix} 2 \\ b \\ a \end{pmatrix}$

$\begin{pmatrix} 3 \\ a \\ b \end{pmatrix}$

MODELS

NON-ERASING TRANSDUCTIONS
WITH ORIGIN OVER Σ, Γ



CLOSED TYPED DATA WORDS
OVER Σ, N, Γ

MODELS

NON-ERASING TRANSDUCTIONS
WITH ORIGIN OVER Σ, Γ



CLOSED TYPED DATA WORDS
OVER Σ, N, Γ

LOGICS

$FO_2 [\leq_{out}, \sigma, MSO_{bin} [\leq_{in}]]$

TRANSDUCTIONS



$FO_2 [\leq, MSO_{bin} [\leq]]$

DATA WORDS

MODELS

NON-ERASING TRANSDUCTIONS
WITH ORIGIN OVER Σ, Γ



CLOSED TYPED DATA WORDS
OVER Σ, N, Γ

LOGICS

$FO_2[\leq_{out}, \sigma, MSO_{bin}[\leq_{in}]]$



$FO_2[\leq, MSO_{bin}[\leq]]$

TRANSDUCTIONS

DATA WORDS

CONSEQUENCE. $FO_2[\leq, MSO_{bin}[\leq]]$ is decidable on typed data words over N
(closedness does not matter).
• $EMSO_2[\leq, MSO_{bin}[\leq]]$ is decidable too.

SUMMARY

- new logic tailored to express non-functional computations
- decidable satisfiability / model-checking / equivalence / functionality
- MSOT synthesis
- new characterization of MSOT
- origin-sensitive! otherwise undecidable

SUMMARY

- new logic tailored to express non-functional transductions
- decidable satisfiability / model-checking / equivalence / functionality
- MSOT synthesis
- new characterization of MSOT
- origin-sensitive! otherwise undecidable

FUTURE WORK

- synthesis of "memory-efficient" machines from \mathcal{L}_T
- extensions to trees & data word transductions

SUMMARY

- new logic tailored to express non-functional transductions
- decidable satisfiability / model-checking / equivalence / functionality
- MSOT synthesis
- new characterization of MSOT
- origin-sensitive! otherwise undecidable

FUTURE WORK

- synthesis of "memory-efficient" machines from \mathcal{L}_T
- extensions to trees & data word transductions

CHALLENGE

- MSOT = 2DFT = 2RFT = SST = \mathcal{L}_T functions = ...
- canonical model?

NON-DETERMINISTIC MSO TRANSDUCERS

Use monadic second order parameters



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

$$\varphi_S(x, y, X) \equiv x, y \in X \wedge \exists z \in X. x < z < y$$

output : b a b



output : a b a