

Which classes of origin graphs  
are generated by transducers?

Mikołaj Bojańczyk, Laure Daviaud,  
Bruno Guillon and Vincent Penelle

Uniwersytet Warszawski – LaBRI

March 26, 2018

— *Réunion Delta* —

---

*This work is part of a project LIPA that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No.683080).*

---

## Definition

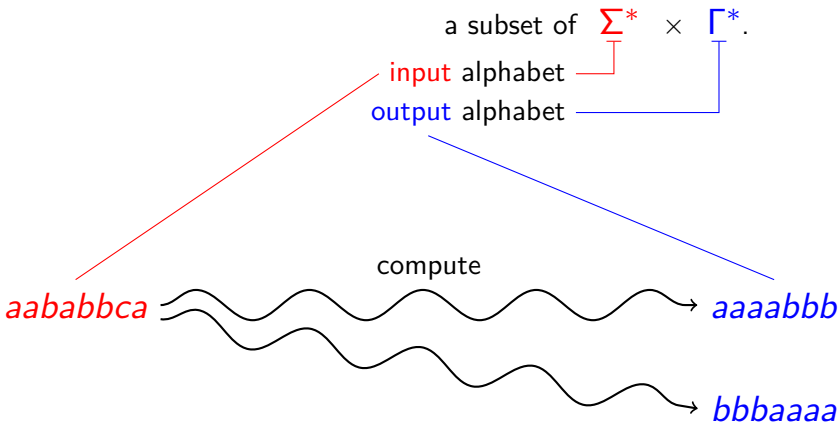
A **transduction** is a binary relation on words:

a subset of  $\Sigma^* \times \Gamma^*$ .

# Word transductions

## Definition

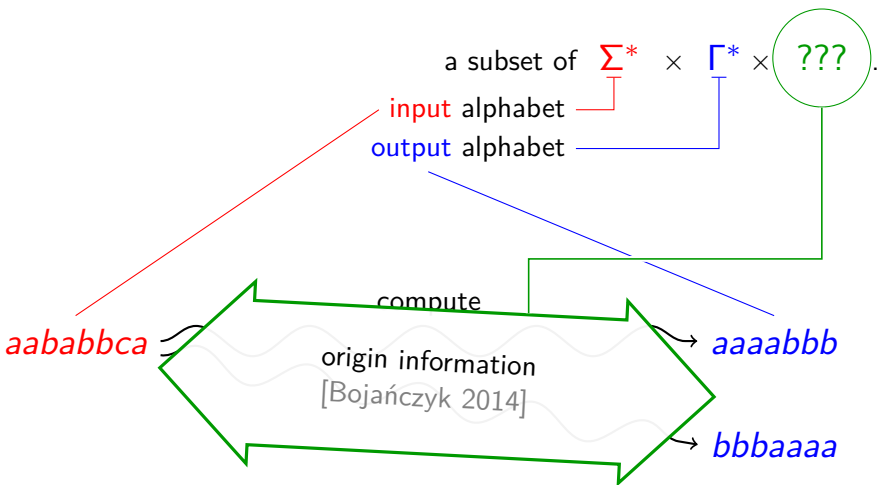
A **transduction** is a binary relation on words:



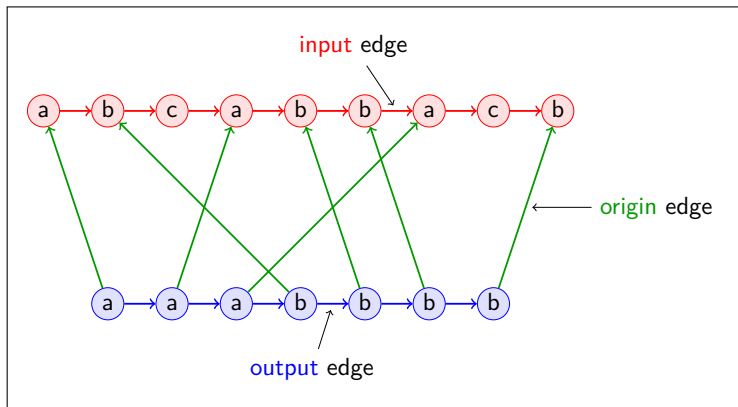
# Word transductions

## Definition

A **transduction** is a binary relation on words:



# Origin graphs



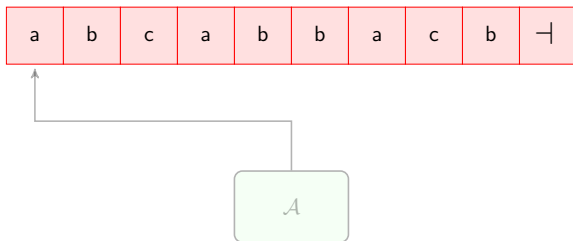
origin: a mapping from **output positions** into **input positions**

# What is the origin semantic of a transducer?

featuring SST and MSOT

# Streaming String Transducers (SST)

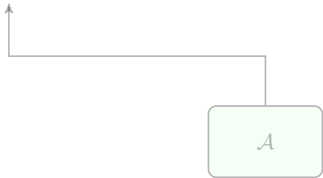
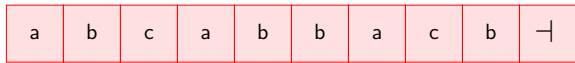
- a 1-way automaton  $\mathcal{A}$





# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished output register  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\begin{cases} X \leftarrow X \cdot a \\ Y \leftarrow \varepsilon \end{cases}$ ,  $\begin{cases} X \leftarrow a \cdot Y \\ Y \leftarrow b \cdot a \cdot X \end{cases}$ ,  $\begin{cases} X \leftarrow b \\ Y \leftarrow X \cdot a \cdot Y \end{cases}$ ,  ~~$\begin{cases} X \leftarrow X \cdot a \\ Y \leftarrow X \cdot Y \end{cases}$~~



register X: 

register Y: 



# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\left\{ \begin{array}{l} X \leftarrow X \cdot a \\ Y \leftarrow \varepsilon \end{array} \right.$  ,  $\left\{ \begin{array}{l} X \leftarrow a \cdot Y \\ Y \leftarrow b \cdot a \cdot X \end{array} \right.$  ,  $\left\{ \begin{array}{l} X \leftarrow b \\ Y \leftarrow X \cdot a \cdot Y \end{array} \right.$  ,  ~~$\left\{ \begin{array}{l} X \leftarrow X \cdot a \\ Y \leftarrow X \cdot Y \end{array} \right.$~~

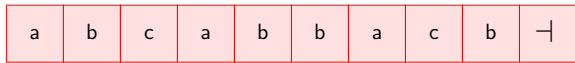
## Definition (Origin semantics for streaming string transducers)

- **origin** of an **output position**: the **position** of the **input** head when the **letter** was created.

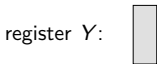
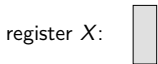
# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\begin{cases} X \leftarrow X \cdot a \\ Y \leftarrow \varepsilon \end{cases}$ ,  $\begin{cases} X \leftarrow a \cdot Y \\ Y \leftarrow b \cdot a \cdot X \end{cases}$ ,  $\begin{cases} X \leftarrow b \\ Y \leftarrow X \cdot a \cdot Y \end{cases}$ ,  ~~$\begin{cases} X \leftarrow X \cdot a \\ Y \leftarrow X \cdot Y \end{cases}$~~



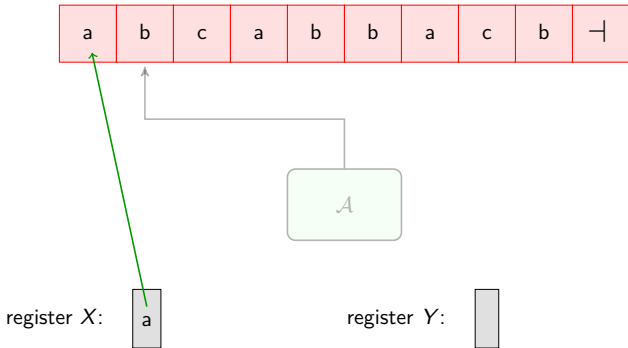
■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$



# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\begin{cases} X \leftarrow X \cdot a \\ Y \leftarrow \varepsilon \end{cases}$ ,  $\begin{cases} X \leftarrow a \cdot Y \\ Y \leftarrow b \cdot a \cdot X \end{cases}$ ,  $\begin{cases} X \leftarrow b \\ Y \leftarrow X \cdot a \cdot Y \end{cases}$ ,  ~~$\begin{cases} X \leftarrow X \cdot a \\ Y \leftarrow X \cdot Y \end{cases}$~~

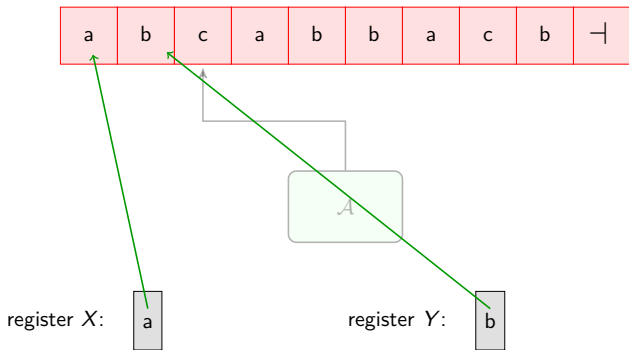


■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~

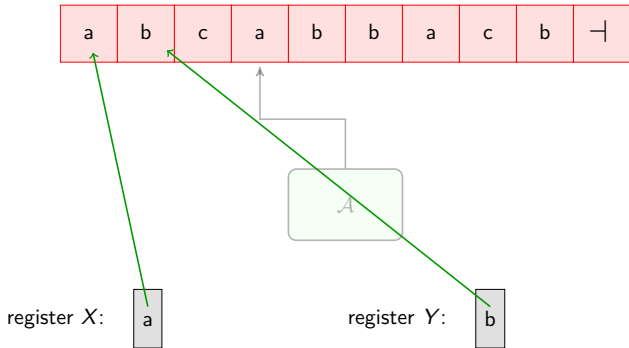


■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~

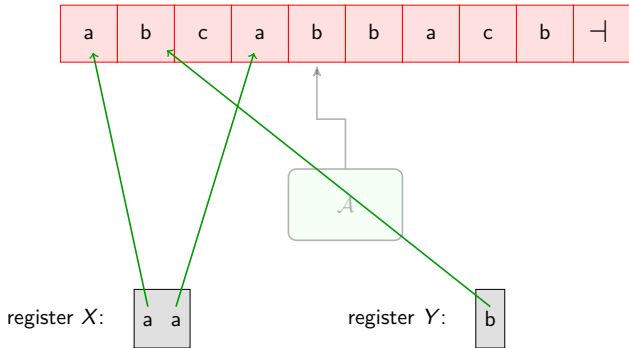


■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~

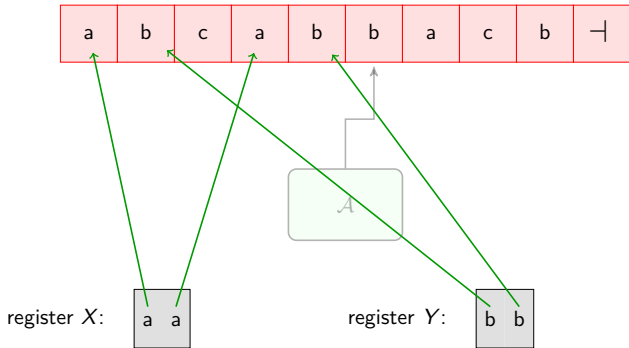


■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~

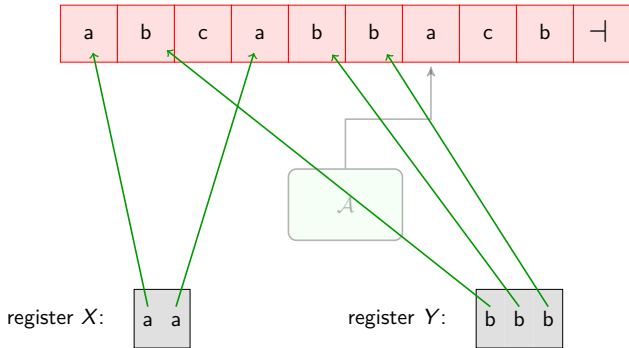


■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~



■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

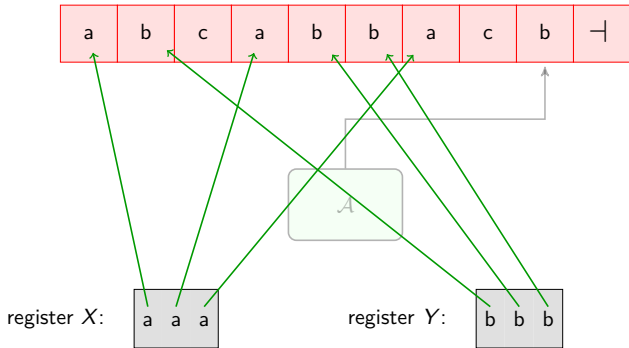




# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~

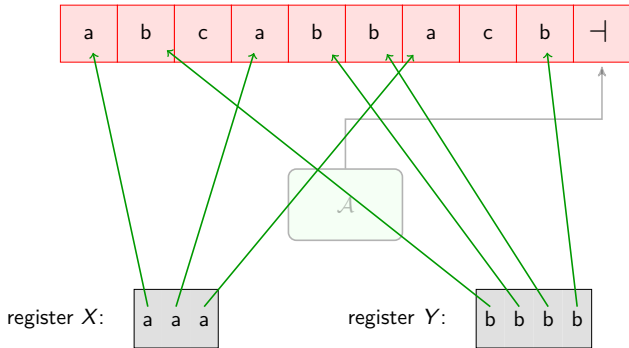


■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~

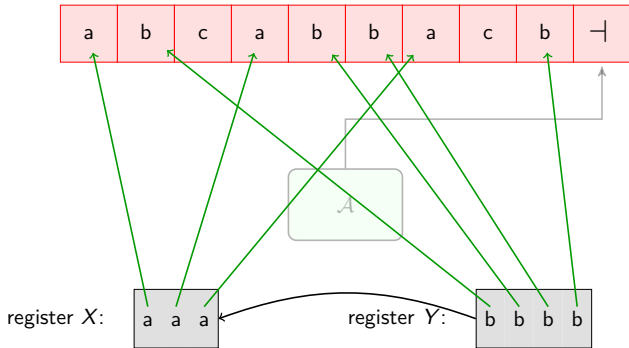


■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~

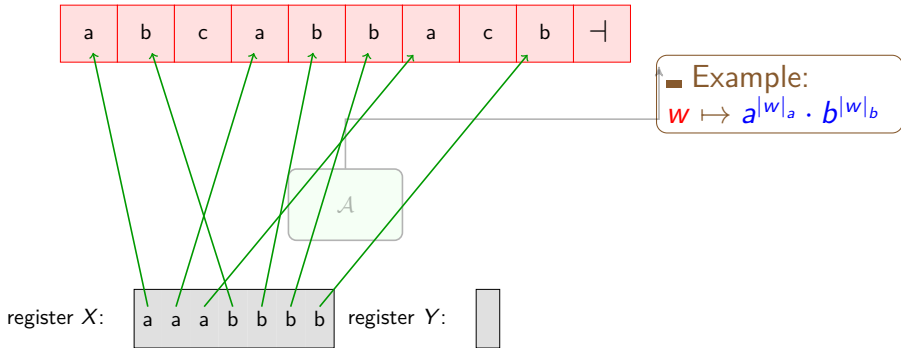


■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$

# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

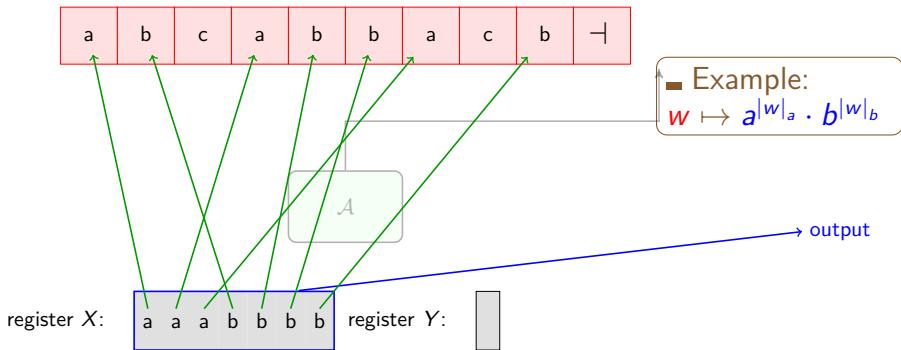
e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}$ ,  $\{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}$ ,  $\{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\}$ ,  ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~



# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

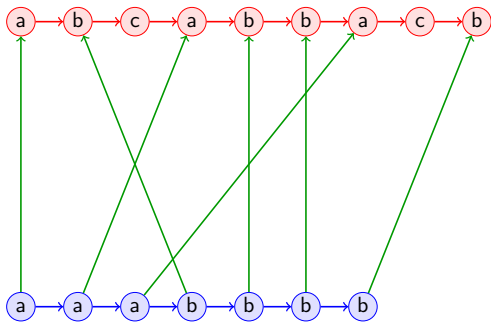
e.g.,  $\{X \leftarrow X \cdot a, Y \leftarrow \varepsilon\}, \{X \leftarrow a \cdot Y, Y \leftarrow b \cdot a \cdot X\}, \{X \leftarrow b, Y \leftarrow X \cdot a \cdot Y\},$   ~~$\{X \leftarrow X \cdot a, Y \leftarrow X \cdot Y\}$~~



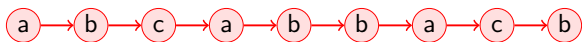
# Streaming String Transducers (SST)

- a 1-way automaton  $\mathcal{A}$
- a finite set  $R$  of registers including a distinguished **output register**  
e.g.,  $R = \{X, Y\}$
- a labelling of transitions by copyless register updates

e.g.,  $\begin{cases} X \leftarrow X \cdot a \\ Y \leftarrow \varepsilon \end{cases}$ ,  $\begin{cases} X \leftarrow a \cdot Y \\ Y \leftarrow b \cdot a \cdot X \end{cases}$ ,  $\begin{cases} X \leftarrow b \\ Y \leftarrow X \cdot a \cdot Y \end{cases}$ ,  ~~$\begin{cases} X \leftarrow X \cdot a \\ Y \leftarrow X \cdot Y \end{cases}$~~



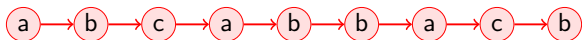
■ Example:  
 $w \mapsto a^{|w|_a} \cdot b^{|w|_b}$





# string-to-string MSO-transduction [Courcelle 1991]

- Nondeterministic MSO-colouring (nondeterministic case);
- Copy (finitely many copies of the **input**);

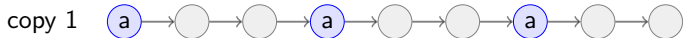


- Nondeterministic MSO-colouring (nondeterministic case);
- Copy (finitely many copies of the **input**);
- MSO-Interpretation
  - a formula for restricting the universe;
  - a formula for each predicate of the **output** vocabulary.



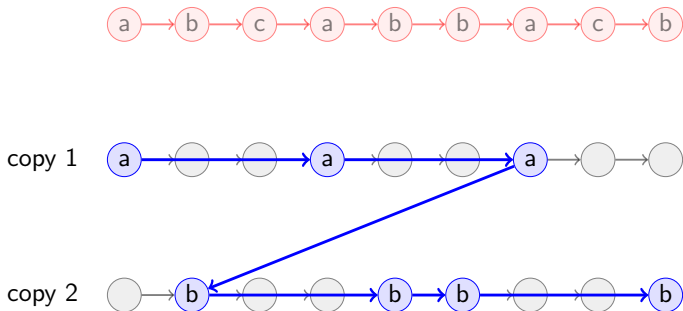
# string-to-string MSO-transduction [Courcelle 1991]

- Nondeterministic MSO-colouring (nondeterministic case);
- Copy (finitely many copies of the **input**);
- MSO-Interpretation
  - a formula for restricting the universe;
  - a formula for each predicate of the **output** vocabulary.



# string-to-string MSO-transduction [Courcelle 1991]

- Nondeterministic MSO-colouring (nondeterministic case);
- Copy (finitely many copies of the **input**);
- MSO-Interpretation
  - a formula for restricting the universe;
  - a formula for each predicate of the **output** vocabulary.



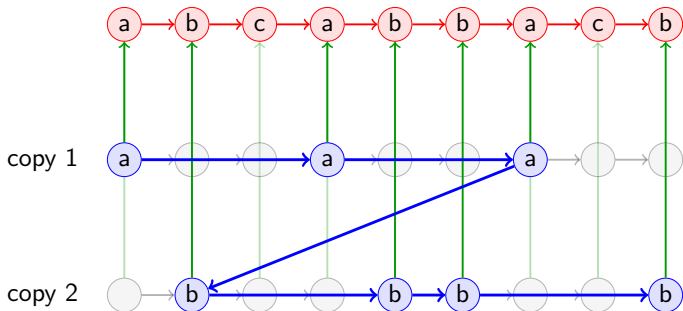
- Nondeterministic MSO-colouring (nondeterministic case);
- Copy (finitely many copies of the **input**);
- MSO-Interpretation
  - a formula for restricting the universe;
  - a formula for each predicate of the **output** vocabulary.

## Definition (origin semantics of MSO-transduction)

- **origin** of an **output position**: the **input vertex** of which **it** is a copy.

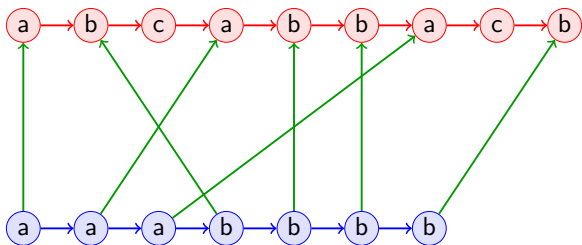
# string-to-origin graph MSO-transduction

- Nondeterministic MSO-colouring (nondeterministic case);
- Copy (finitely many copies of the **input**);
- MSO-Interpretation
  - a formula for restricting the universe;
  - a formula for each predicate of the **output** vocabulary.



# string-to-origin graph MSO-transduction

- Nondeterministic MSO-colouring (nondeterministic case);
- Copy (finitely many copies of the **input**);
- MSO-Interpretation
  - a formula for restricting the universe;
  - a formula for each predicate of the **output** vocabulary.

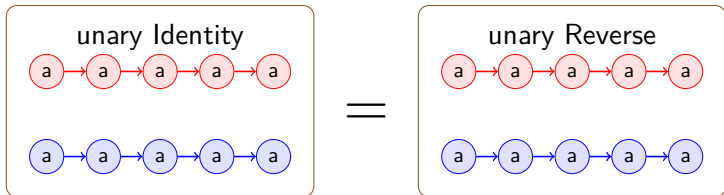


What do we get from **origin** information?



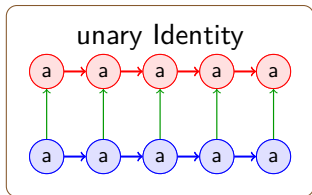
# Origin semantics is thinner grained

## Examples

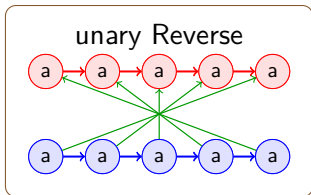


# Origin semantics is thinner grained

## Examples

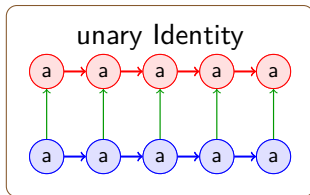


$\neq$

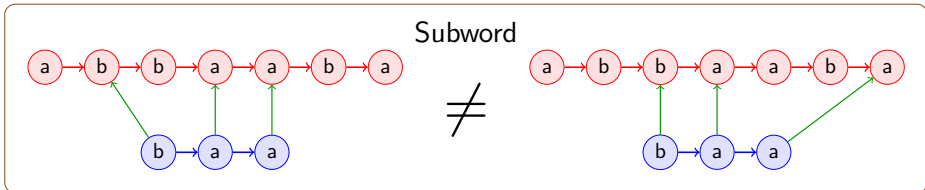
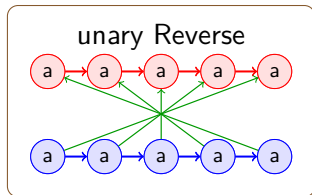


# Origin semantics is thinner grained

## Examples



≠



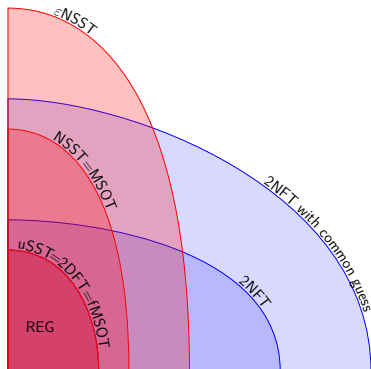
≠

# What is a *regular* transduction with origin?

This is still true with **origin** information.

[Bojańczyk 2014]

also true for closure under composition, decidability of equivalence. . .



- SST: Streaming String Transducer
- MSOT: MSO-transduction
- 2FT : 2-way finite transducer

**Theorem:** The following is decidable :

## Input

- an NSST  $\mathcal{A}$
- an MSO formula  $\phi$  over the corresponding origin vocabulary

## Question

- Is  $\phi$  true in some origin graph in the origin semantics of  $\mathcal{A}$ ?

origin vocabulary: binary predicates  $\rightarrow$ ,  $\rightarrow$ ,  $\rightarrow$   
and labelling in  $\Sigma \cup \Gamma$ ;

## Example

*“the **origin mapping** is bijective and letter-preserving.”*

*“the **output** may be split in two parts  
such that the **origin mapping** is order-preserving on each part.”*

**Theorem:** The following is decidable :

## Input

- an NSST  $\mathcal{A}$
- an MSO formula  $\phi$  over the corresponding origin vocabulary

## Question

- Is  $\phi$  true in some origin graph in the origin semantics of  $\mathcal{A}$ ?

Proof: Let  $\mathcal{A}$  and  $\phi$  be fixed.

- there is a string-to-origin graph MSO-transduction  $\rho$  equivalent to  $\mathcal{A}$

**Theorem:** The following is decidable :

## Input

- an NSST  $\mathcal{A}$
- an MSO formula  $\phi$  over the corresponding origin vocabulary

## Question

- Is  $\phi$  true in some origin graph in the origin semantics of  $\mathcal{A}$ ?

Proof: Let  $\mathcal{A}$  and  $\phi$  be fixed.

- there is a string-to-origin graph MSO-transduction  $\rho$  equivalent to  $\mathcal{A}$
- we consider  $\mathcal{G} = \{G \text{ origin graph} \mid \phi \text{ is true over } G\}$

**Theorem:** The following is decidable :

## Input

- an NSST  $\mathcal{A}$
- an MSO formula  $\phi$  over the corresponding origin vocabulary

## Question

- Is  $\phi$  true in some origin graph in the origin semantics of  $\mathcal{A}$ ?

Proof: Let  $\mathcal{A}$  and  $\phi$  be fixed.

- there is a **string-to-origin graph** MSO-transduction  $\rho$  equivalent to  $\mathcal{A}$
- we consider  $\mathcal{G} = \{G \text{ origin graph} \mid \phi \text{ is true over } G\}$
- by Backward Translation Theorem [Courcelle91],  
 $\rho^{-1}(\mathcal{G})$  is regular and can be tested for emptiness.  $\square$



Which properties of origin graphs  
characterise  
regular sets of origin graphs?

# Which sets of origin graphs are generated by transducers?

## Theorem:

A set of origin graphs is realised by an **unambiguous** SST if and only if it is

- **mso-definable:**

an MSO sentence using  $\rightarrow$ ,  $\rightarrow$ ,  $\rightarrow$  and labelling in  $\Sigma \cup \Gamma$ ;

- **functional:**

for each **input word**, there exists at most one origin graph;

- **bounded degree:**

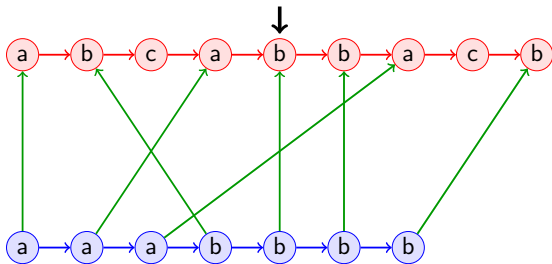
each **input position** is the **origin** of at most **m** **output positions**;

- **bounded crossing:**

NEXT SLIDE.

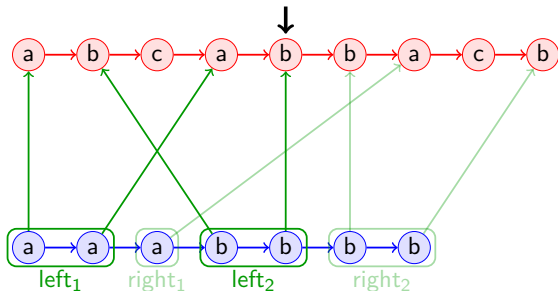
# Crossing

- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position



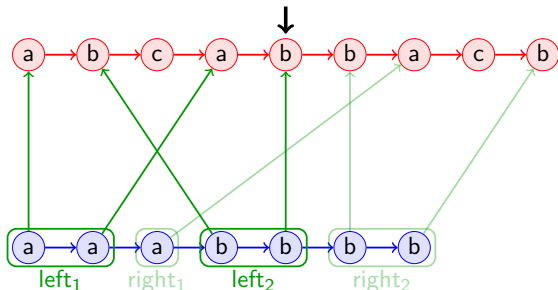
# Crossing

- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position



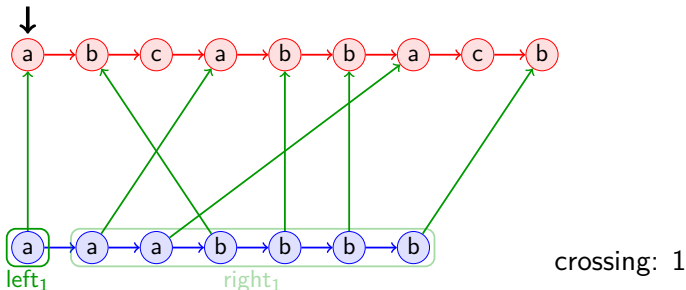
# Crossing

- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position



# Crossing

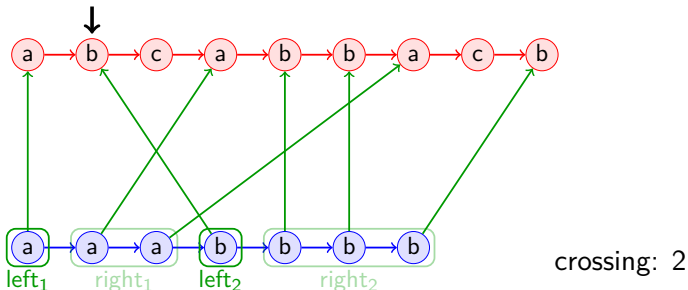
- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:           max of the crossings



( $\equiv$  particular path decomposition of bounded width)

# Crossing

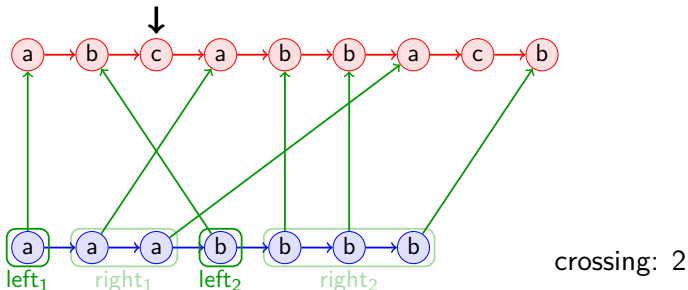
- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:           max of the crossings



( $\equiv$  particular path decomposition of bounded width)

# Crossing

- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:           max of the crossings

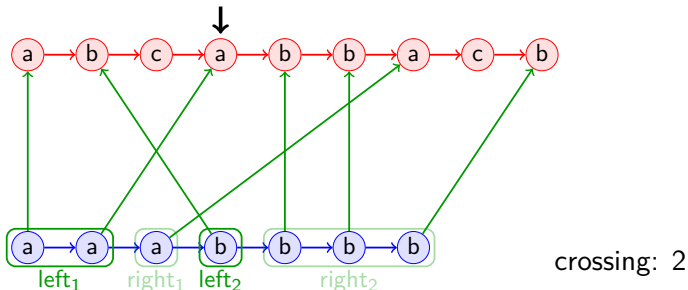


( $\equiv$  particular path decomposition of bounded width)



# Crossing

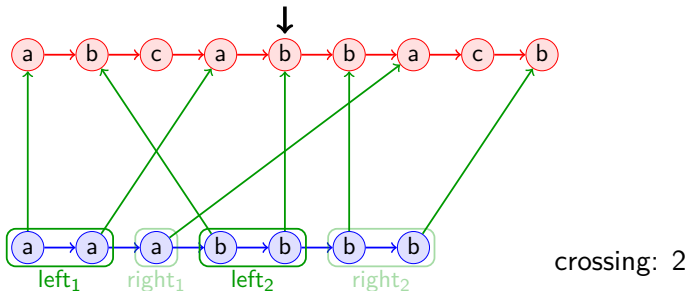
- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:           max of the crossings



( $\equiv$  particular path decomposition of bounded width)

# Crossing

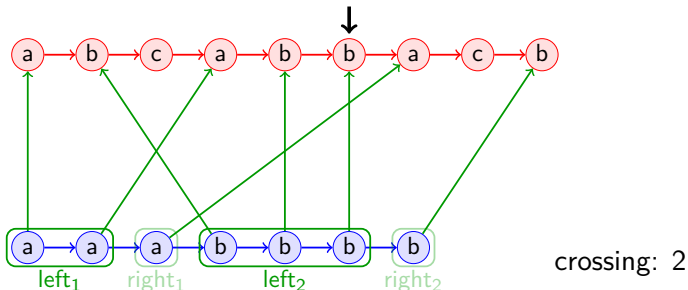
- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:           max of the crossings



( $\equiv$  particular path decomposition of bounded width)

# Crossing

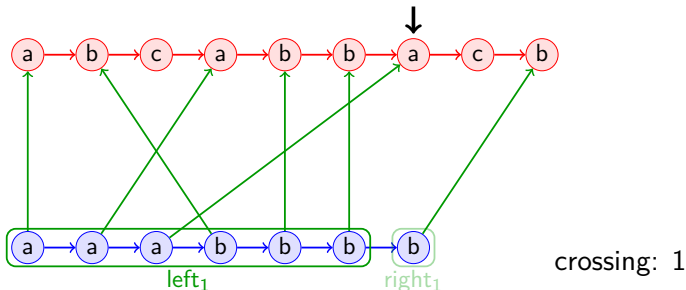
- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:           max of the crossings



( $\equiv$  particular path decomposition of bounded width)

# Crossing

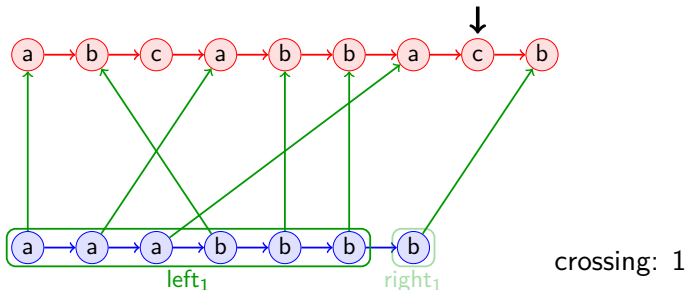
- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:           max of the crossings



( $\equiv$  particular path decomposition of bounded width)

# Crossing

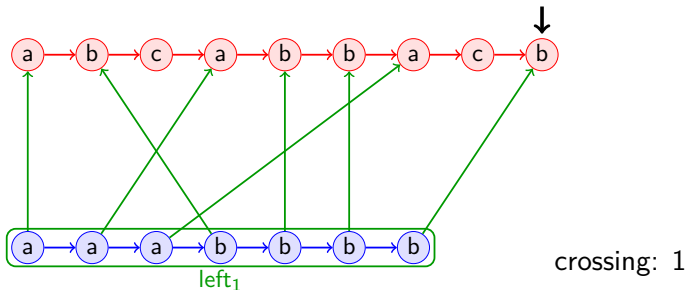
- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:           max of the crossings



( $\equiv$  particular path decomposition of bounded width)

# Crossing

- crossing of an **input position**  
number of maximal factors of the **output**  
that **originate** in the **input prefix** ended by the position
- crossing of an origin graph:      max of the crossings : 2



( $\equiv$  particular path decomposition of bounded width)

# Which sets of origin graphs are generated by transducers?

## Theorem:

A set of origin graphs is realised by an unambiguous SST if and only if it is

- **mso-definable:**

an MSO sentence using  $\rightarrow$ ,  $\rightarrow$ ,  $\rightarrow$  and labelling in  $\Sigma \cup \Gamma$ ;

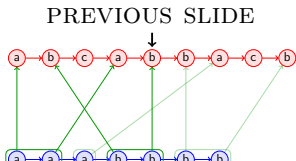
- **functional:**

for each **input word**, there exists at most one origin graph;

- **bounded degree:**

each **input position** is the **origin** of at most **m** **output positions**;

- **crossing bounded:**



# Which sets of origin graphs are generated by transducers?

## Theorem:

A set of origin graphs is realised by an unambiguous  $k$ -registers SST if and only if it is

- **mso-definable:**

an MSO sentence using  $\rightarrow$ ,  $\rightarrow$ ,  $\rightarrow$  and labelling in  $\Sigma \cup \Gamma$ ;

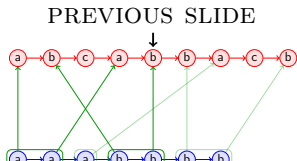
- **functional:**

for each **input word**, there exists at most one origin graph;

- **bounded degree:**

each **input position** is the **origin** of at most  **$m$  output positions**;

- **crossing bounded** by  $k$  :





# Which sets of origin graphs are generated by transducers?

## Theorem:

A set of origin graphs is realised by an  
if and only if it is

$k$ -registers SST

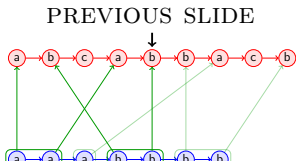
- **mso-definable:**

an MSO sentence using  $\rightarrow$ ,  $\rightarrow$ ,  $\rightarrow$  and labelling in  $\Sigma \cup \Gamma$ ;

- **bounded degree:**

each **input position** is the **origin** of at most  **$m$  output positions**;

- **crossing bounded** by  $k$  :

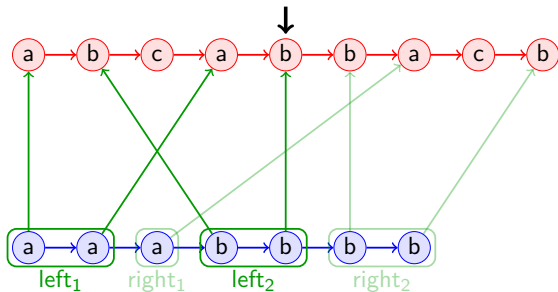


## Sketch of the proof $\implies$

- unambiguous  $\implies$  functional
- NSST  $\implies$  bounded degree

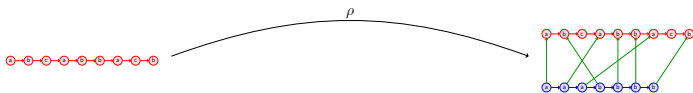
# Sketch of the proof $\implies$

- unambiguous  $\implies$  functional
- NSST  $\implies$  bounded degree
- $k$ -register  $\implies$  crossing bounded by  $k$



# Sketch of the proof $\implies$

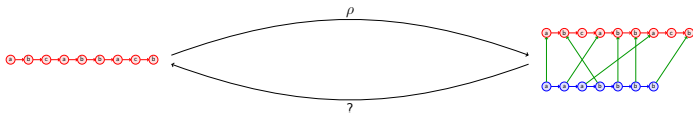
- unambiguous  $\implies$  functional
- NSST  $\implies$  bounded degree
- $k$ -register  $\implies$  crossing bounded by  $k$
- NSST  $\implies$  string-to-origin graph MSO-transduction



# Sketch of the proof $\implies$

- unambiguous  $\implies$  functional
- NSST  $\implies$  bounded degree
- $k$ -register  $\implies$  crossing bounded by  $k$
- NSST  $\implies$  string-to-origin graph MSO-transduction

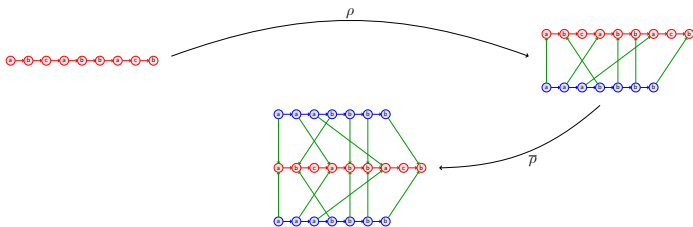
**Proposition:** we can invert this MSO-transduction



# Sketch of the proof $\implies$

- unambiguous  $\implies$  functional
- NSST  $\implies$  bounded degree
- $k$ -register  $\implies$  crossing bounded by  $k$
- NSST  $\implies$  string-to-origin graph MSO-transduction

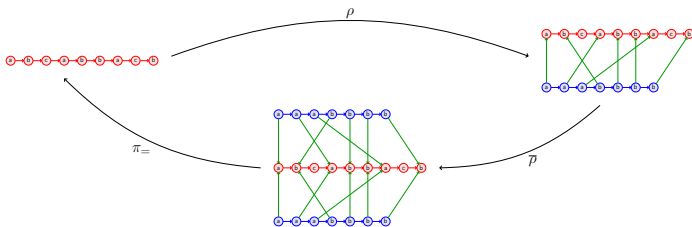
**Proposition:** we can invert this MSO-transduction



# Sketch of the proof $\implies$

- unambiguous  $\implies$  functional
- NSST  $\implies$  bounded degree
- $k$ -register  $\implies$  crossing bounded by  $k$
- NSST  $\implies$  string-to-origin graph MSO-transduction

**Proposition:** we can invert this MSO-transduction  
 $\implies$  MSO-definable

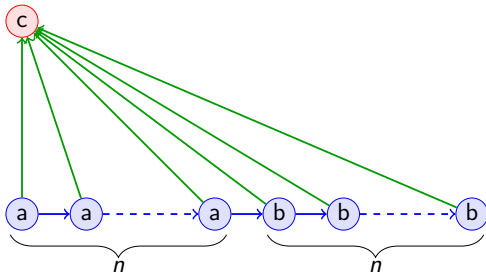


# Sketch of the proof $\implies$

- unambiguous  $\implies$  functional
- NSST  $\implies$  bounded degree
- $k$ -register  $\implies$  crossing bounded by  $k$
- NSST  $\implies$  string-to-origin graph MSO-transduction

**Proposition:** we can invert this MSO-transduction  
 $\implies$  MSO-definable

Note: False when  $\varepsilon$ -transitions are allowed.





## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$

## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$

### Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.

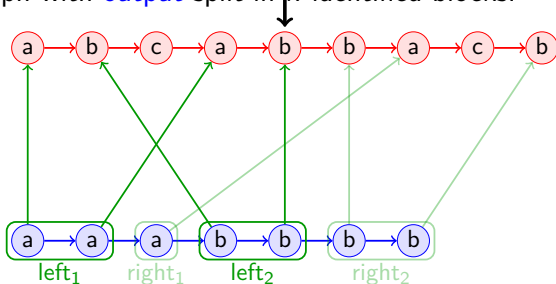
# Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$

## Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.



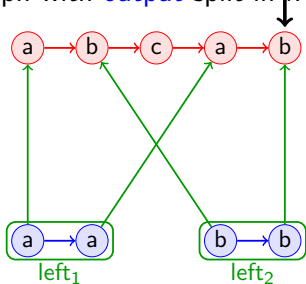
# Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$  with crossing bounded by  $k$

## Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.



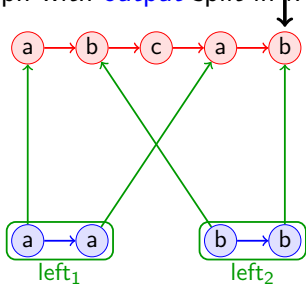
## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs

### Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.



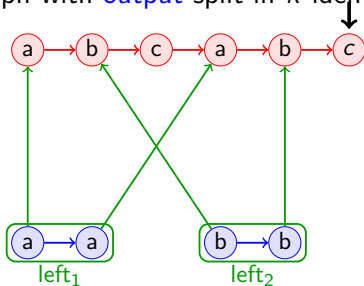
# Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs

## Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.



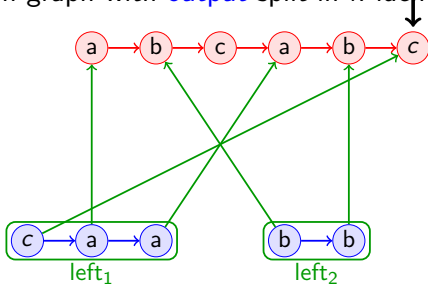
## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$  with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs

### Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.



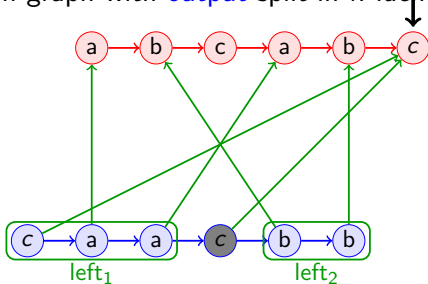
# Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$  with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs

## Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.





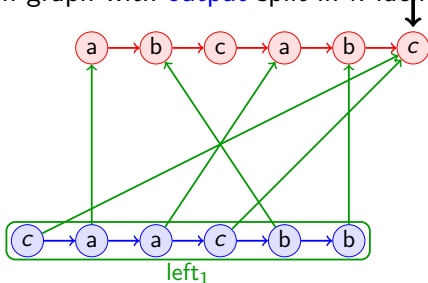
## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs

### Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.



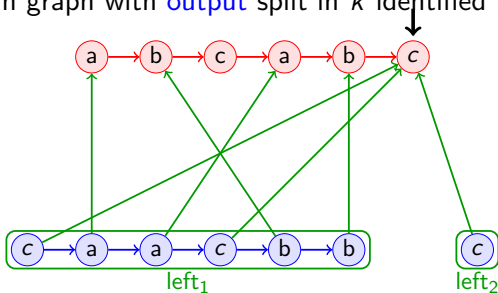
# Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs

## Definition

$k$ -block origin graphs ( $k$ -BLOGs):

An origin graph with **output** split in  $k$  identified blocks.



## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs
- the **folding** of a word  $w$  over  $\Omega_k^*$  is the  $k$ -BLOG  $\alpha_k(w)$   
obtained from the empty graph by applying the operations.  
 $\alpha_k$  can be realised by an MSO-transduction.

## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs
- the folding of a word  $w$  over  $\Omega_k^*$  is the  $k$ -BLOG  $\alpha_k(w)$   
obtained from the empty graph by applying the operations.  
 $\alpha_k$  can be realised by an MSO-transduction.
- there exists a regular language  $L \subseteq \Omega_k^*$  such that

$$g \in G \iff g = \alpha_k(w) \text{ for some } w \in L$$

## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs
- the folding of a word  $w$  over  $\Omega_k^*$  is the  $k$ -BLOG  $\alpha_k(w)$   
obtained from the empty graph by applying the operations.  
 $\alpha_k$  can be realised by an MSO-transduction.
- there exists a regular language  $L \subseteq \Omega_k^*$  such that
$$g \in G \iff g = \alpha_k(w) \text{ for some } w \in L$$
- from an automaton recognising  $L$ ,  
we build a NSST with  $\varepsilon$ -transitions realising  $G$

## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs
- the folding of a word  $w$  over  $\Omega_k^*$  is the  $k$ -BLOG  $\alpha_k(w)$   
obtained from the empty graph by applying the operations.  
 $\alpha_k$  can be realised by an MSO-transduction.
- there exists a regular language  $L \subseteq \Omega_k^*$  such that
$$g \in G \iff g = \alpha_k(w) \text{ for some } w \in L$$
- from an automaton recognising  $L$ ,  
we build a NSST with  $\varepsilon$ -transitions realising  $G$ 
  - if bounded degree  $\implies$  elimination of  $\varepsilon$ -transition

## Sketch of the proof $\Leftarrow$

- Start with an MSO-definable set of origin graphs  $G$   
with crossing bounded by  $k$
- we define a finite set of (partial) operations  $\Omega_k$  on  $k$ -BLOGs
- the folding of a word  $w$  over  $\Omega_k^*$  is the  $k$ -BLOG  $\alpha_k(w)$   
obtained from the empty graph by applying the operations.  
 $\alpha_k$  can be realised by an MSO-transduction.
- there exists a regular language  $L \subseteq \Omega_k^*$  such that

$$g \in G \iff g = \alpha_k(w) \text{ for some } w \in L$$

- from an automaton recognising  $L$ ,  
we build a NSST with  $\varepsilon$ -transitions realising  $G$ 
  - if bounded degree  $\implies$  elimination of  $\varepsilon$ -transition
  - if functional  $\implies$  disambiguation

**Corollary:** The following is decidable:

## **Input**

- Two NSST,  $\mathcal{A}$  and  $\mathcal{B}$ .

## **Question**

- Whether they have the same origin semantics.



**Corollary:** The following is decidable:

## Input

- Two NSST,  $\mathcal{A}$  and  $\mathcal{B}$ .

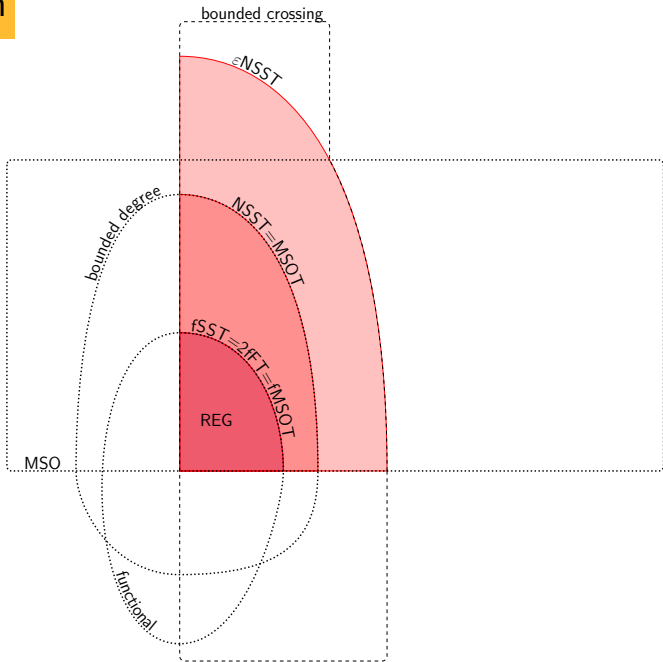
## Question

- Whether they have the same origin semantics.

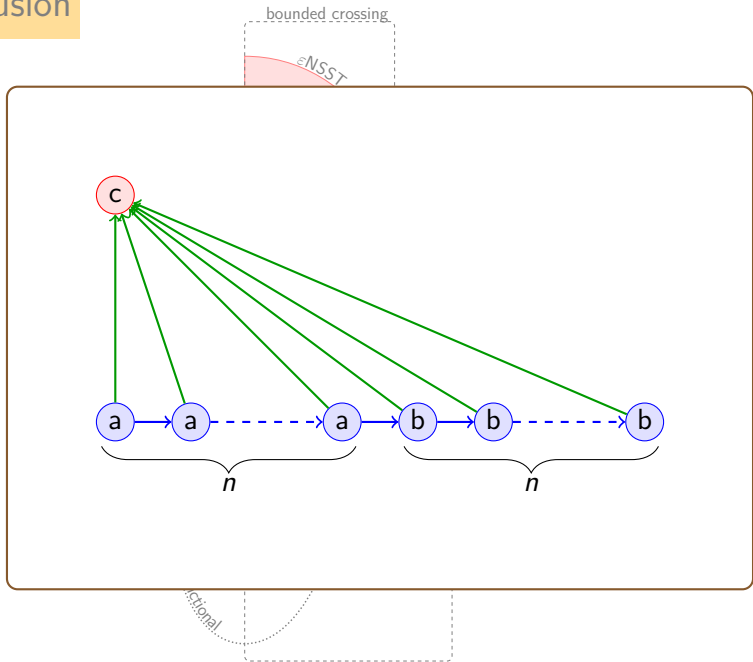
**Proof:** We show that we can check whether  $\mathcal{A} \cap \bar{\mathcal{B}}$  is empty.

- The origin semantics of  $\mathcal{B}$  is MSO-definable by a formula  $\phi$ ,
- We can check whether  $\neg\phi$  is true in some origin graph in the origin semantics of  $\mathcal{A}$ .

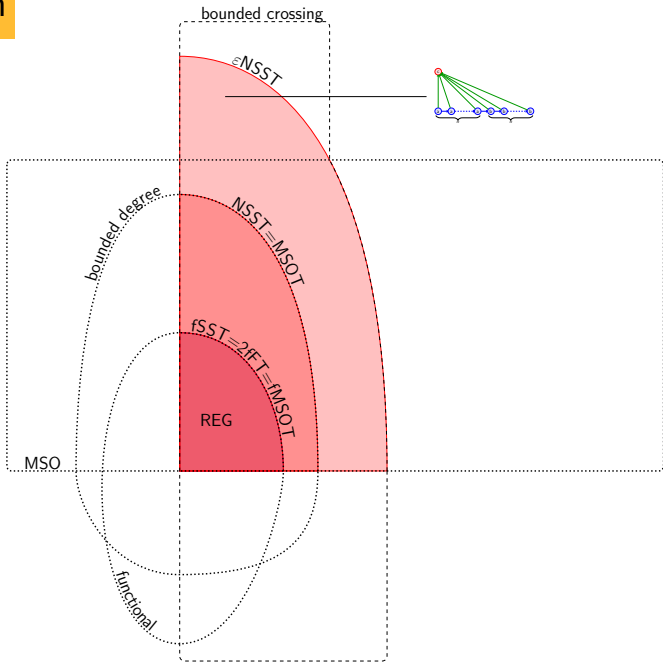
# Conclusion



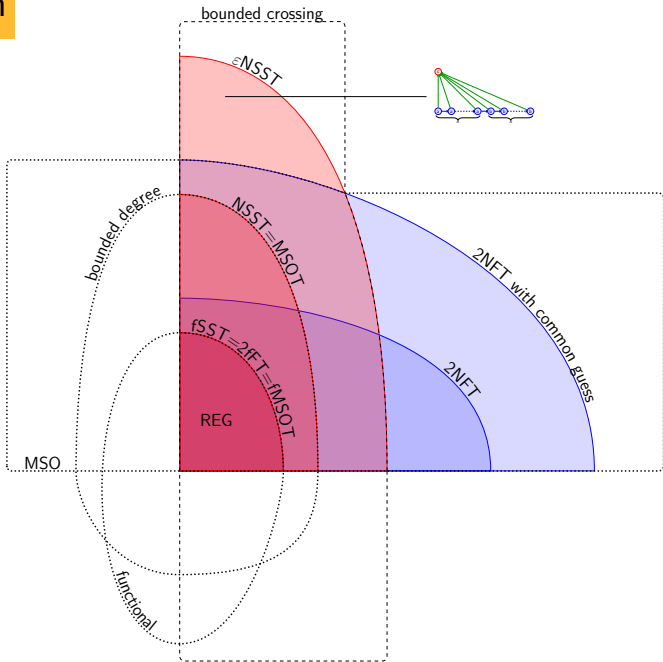
# Conclusion



# Conclusion

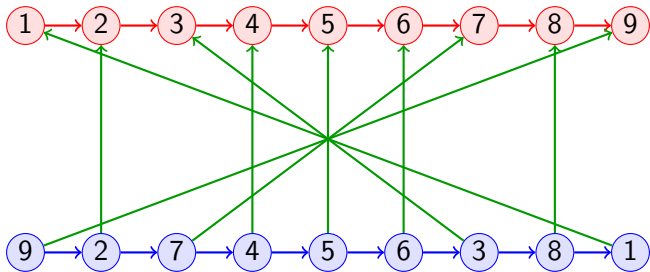


# Conclusion



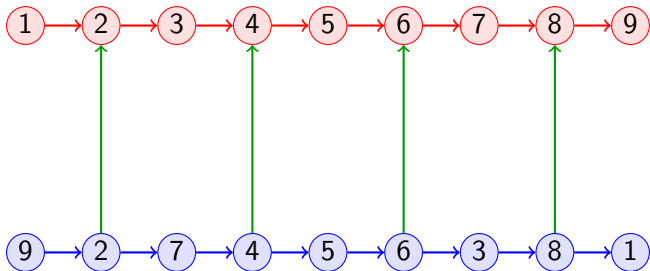
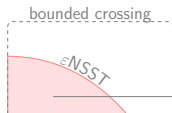
# Conclusion

bounded crossing



functional

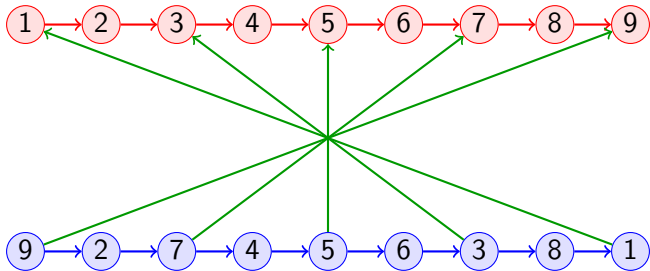
# Conclusion



functional

# Conclusion

bounded crossing

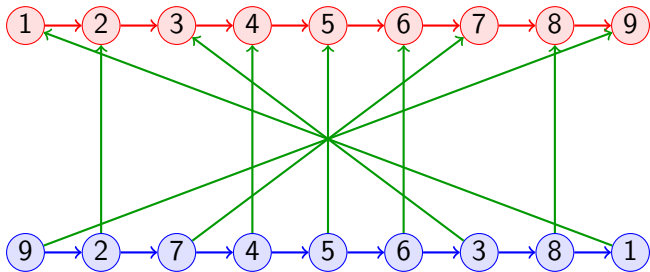


functional



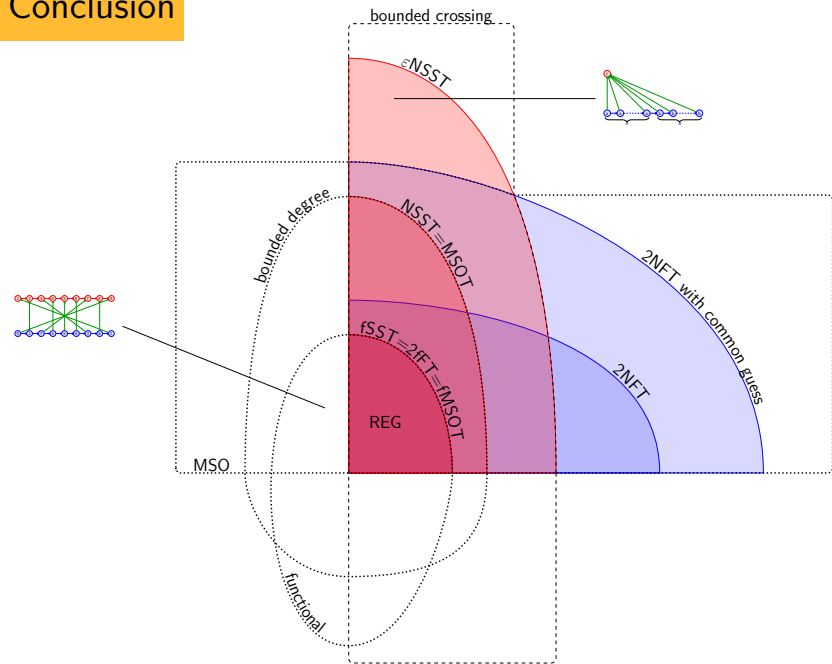
# Conclusion

bounded crossing

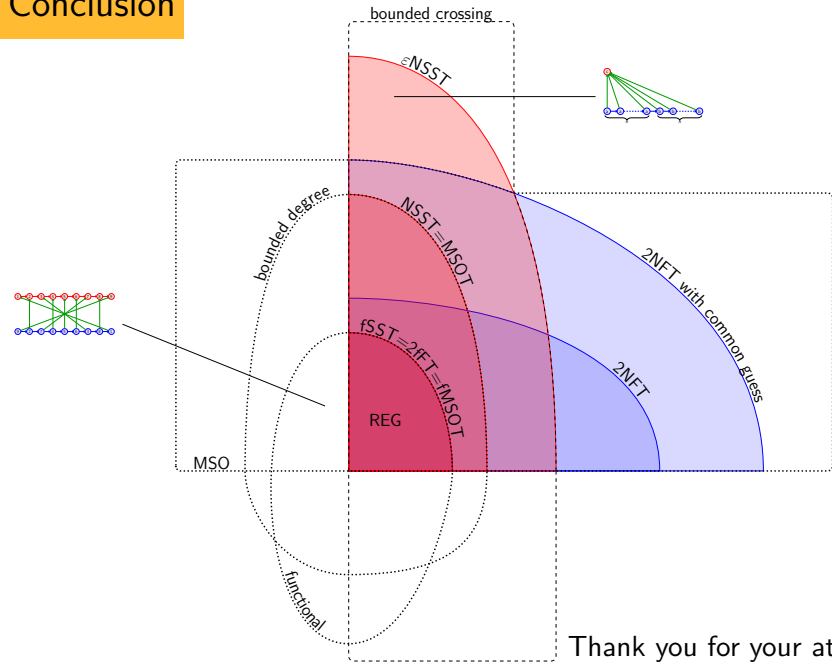


functional

# Conclusion



# Conclusion



Thank you for your attention.