A survey on transducers

decidability, logic and algebra

Pierre-Alain Reynier

LIF, Aix-Marseille University & CNRS

Formal methods to improve software

Software systems are complex and ubiquitous

- critical systems ➜ reliability
- widespread ➜ efficiency, scalability

➜ need for formal methods

Formal methods to improve software

Software systems are complex and ubiquitous

- critical systems ➜ reliability
- widespread ➜ efficiency, scalability

➜ need for formal methods

Automata-based approaches:

- model checking
- controller synthesis
- performance evaluation
- model optimization

Objective: Improve our theoretical understanding of automata models

From Languages to Transductions

Languages	Transductions
$Input \to \{0,1\}$	$Input \to Outputs$
automata	transducers
accept inputs	transform inputs

From Languages to Transductions

Languages	Transductions
$Input \to \{0,1\}$	$Input \to Outputs$
automata	transducers
accept inputs	transform inputs

Applications:

- Word-to-word transducers:
  - language and speech processing
  - model-checking infinite state-space systems
  - reactive systems
  - verification of web sanitizers (tool BEK)
  - string pattern matching
- Nested-word-to-word transducers:
  - XML transformations
  - model for recursive programs

Two-fold objective: theory and applications

Develop the theory of transducers/transductions:

- expressiveness
- closure properties
- decidability

Two-fold objective: theory and applications

Develop the theory of transducers/transductions:

- expressiveness
- closure properties
- decidability

and their applications:

- verification
  - equivalence, type checking ($T(L_{in}) \subseteq L_{out}$)
- model optimization
  - streaming evaluation, minimization of resources

Simplification of models

General Problem

Given a (complex) model of a transformation, does there exist an equivalent simpler model?

➜ Natural question:

- minimization of automata
- make model deterministic
- reduce number of registers
- 2way: reduce number of passes
- ...

Automata/Logic/Algebra connections

Rich theory of regular languages: multiple presentation

Each presentation owns its assets:

- Automata: decision problems
- Logic: closure properties
- Algebra: characterizations of subclasses

➜ try to lift this theory to transductions

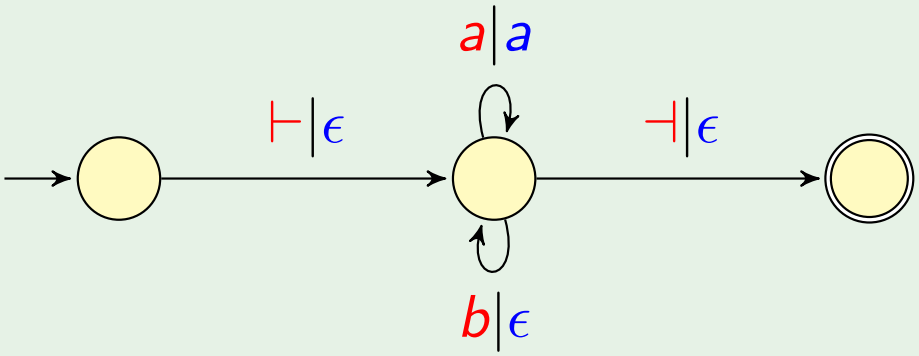
Overview

Overview

(Two-way) finite state transducers

= associate output words with transitions of a finite state automaton

Example (A transducer T)



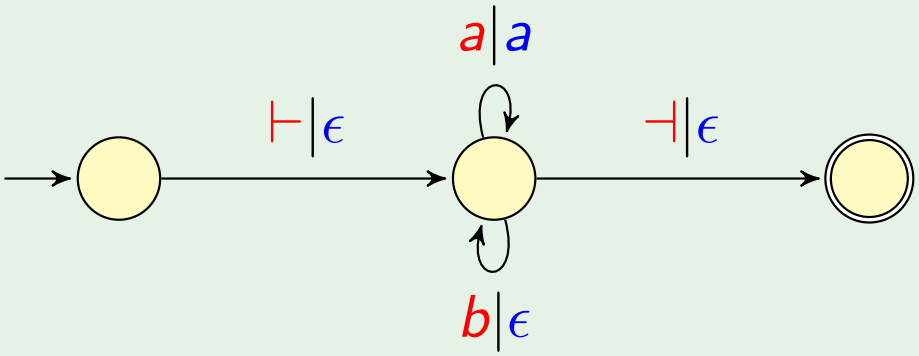
Semantics $[\![T]\!]$: $f : \vdash w \dashv \mapsto a^{\#_a(w)}$, with $w \in \{a, b\}^*$

Non-determinism: semantics is a relation

(Two-way) finite state transducers

= associate output words with transitions of a finite state automaton

Example (A transducer T)



Semantics $[\![T]\!]$: $f : \vdash w \dashv \mapsto a^{\#_a(w)}$, with $w \in \{a, b\}^*$

Non-determinism: semantics is a relation

A transducer is:

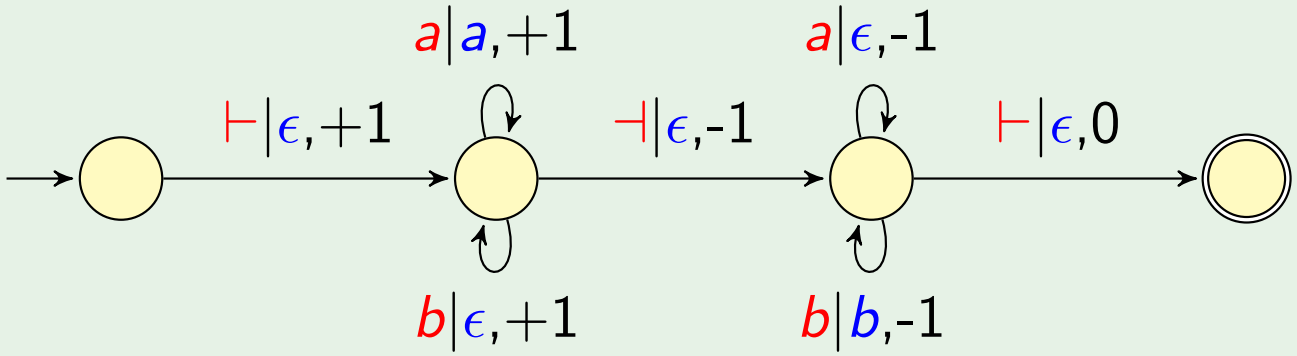
- functional if it realizes a function
- deterministic if the underlying automaton is deterministic

Classes: DFT, fNFT, NFT

(Two-way) finite state transducers

= associate output words with transitions of a finite state automaton

Example (A transducer T)



Semantics $[\![T]\!]$: $f : \vdash w \dashv \mapsto a^{\#_a(w)} b^{\#_b(w)}$, with $w \in \{a, b\}^*$

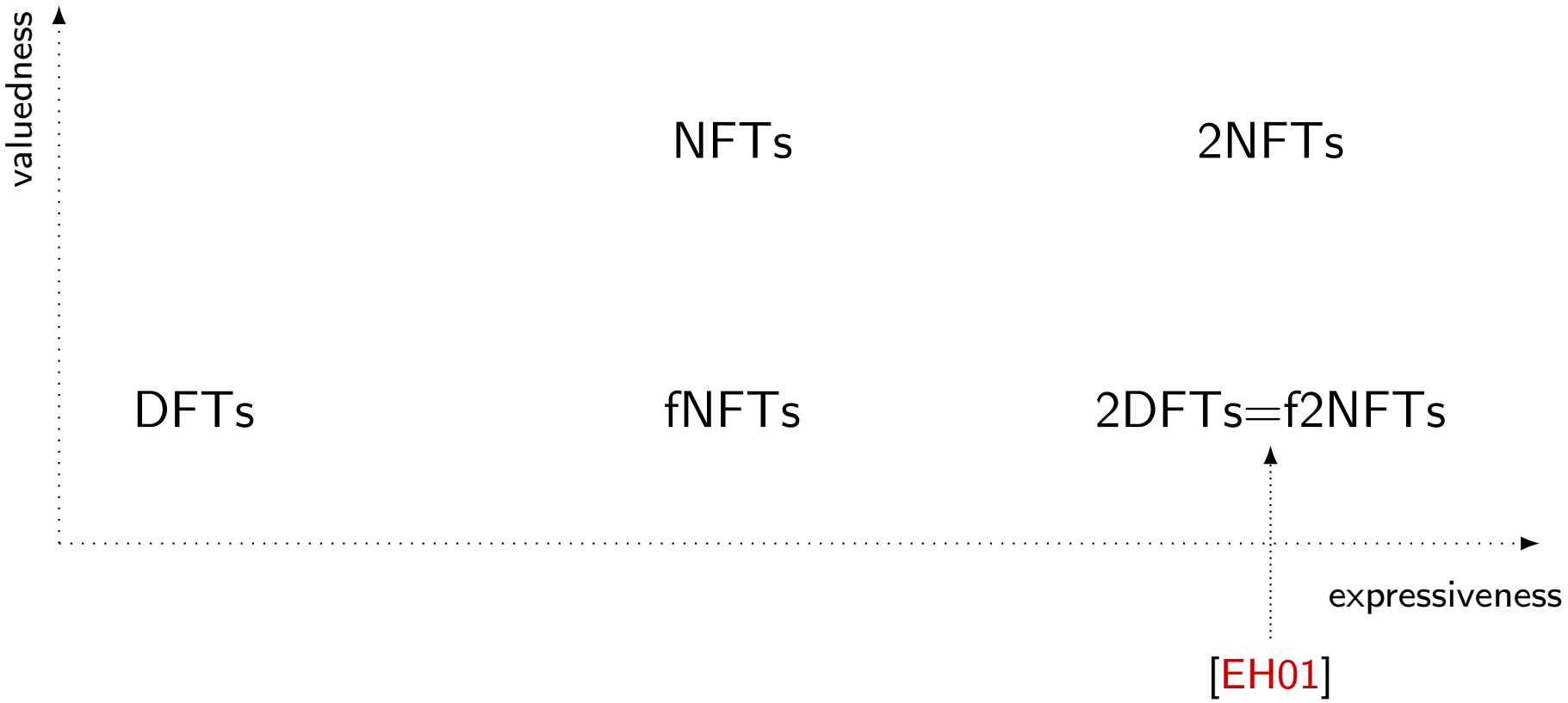
Non-determinism: semantics is a relation

A transducer is:

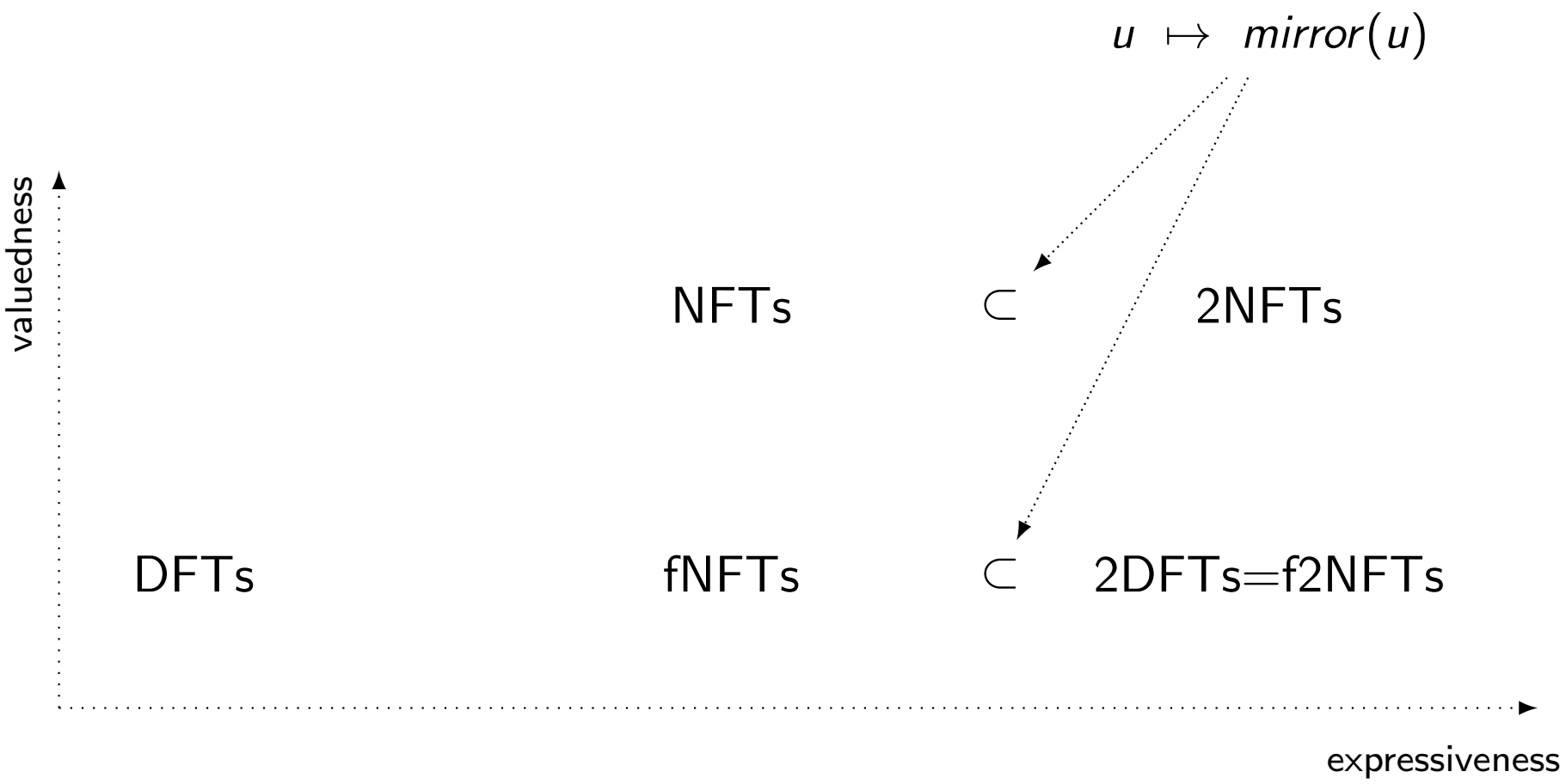
- functional if it realizes a function
- deterministic if the underlying automaton is deterministic

Classes: DFT, fNFT, NFT, 2DFT, f2NFT, 2NFT

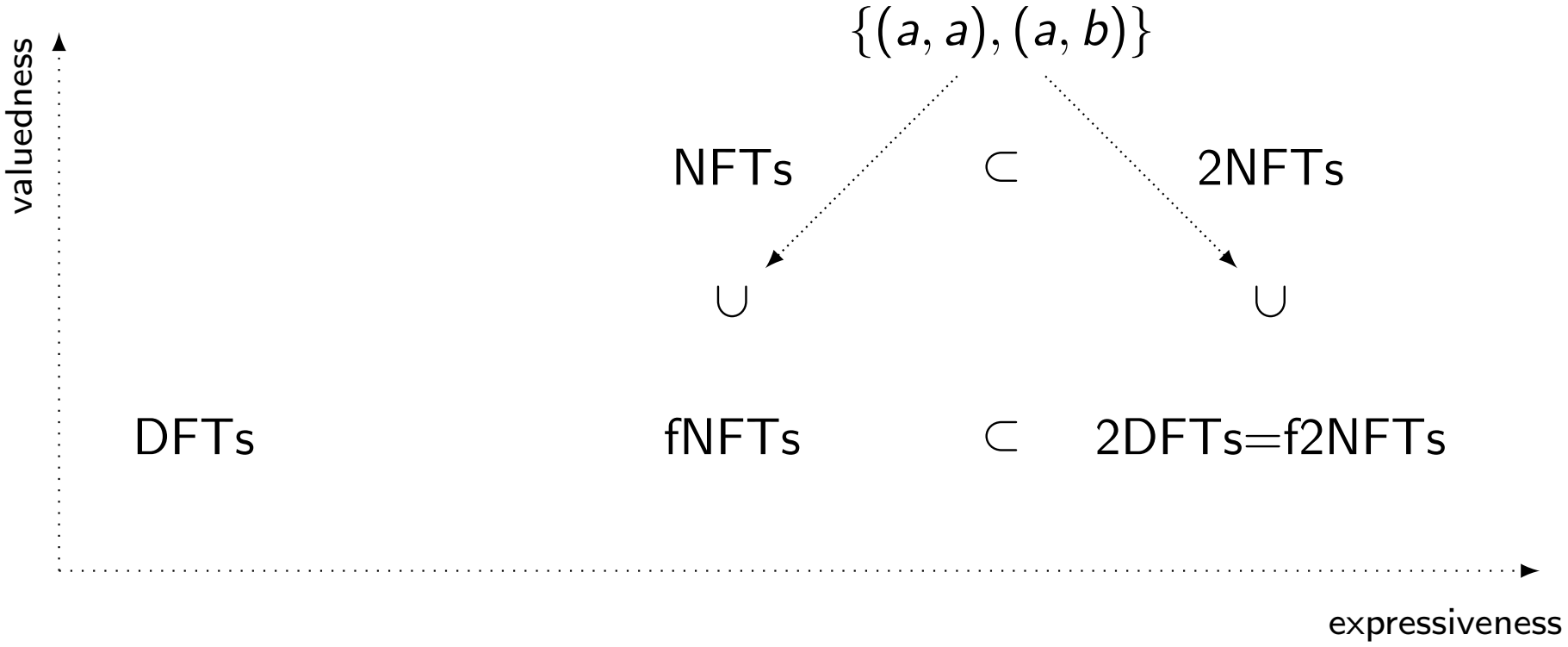
Classes of Transductions



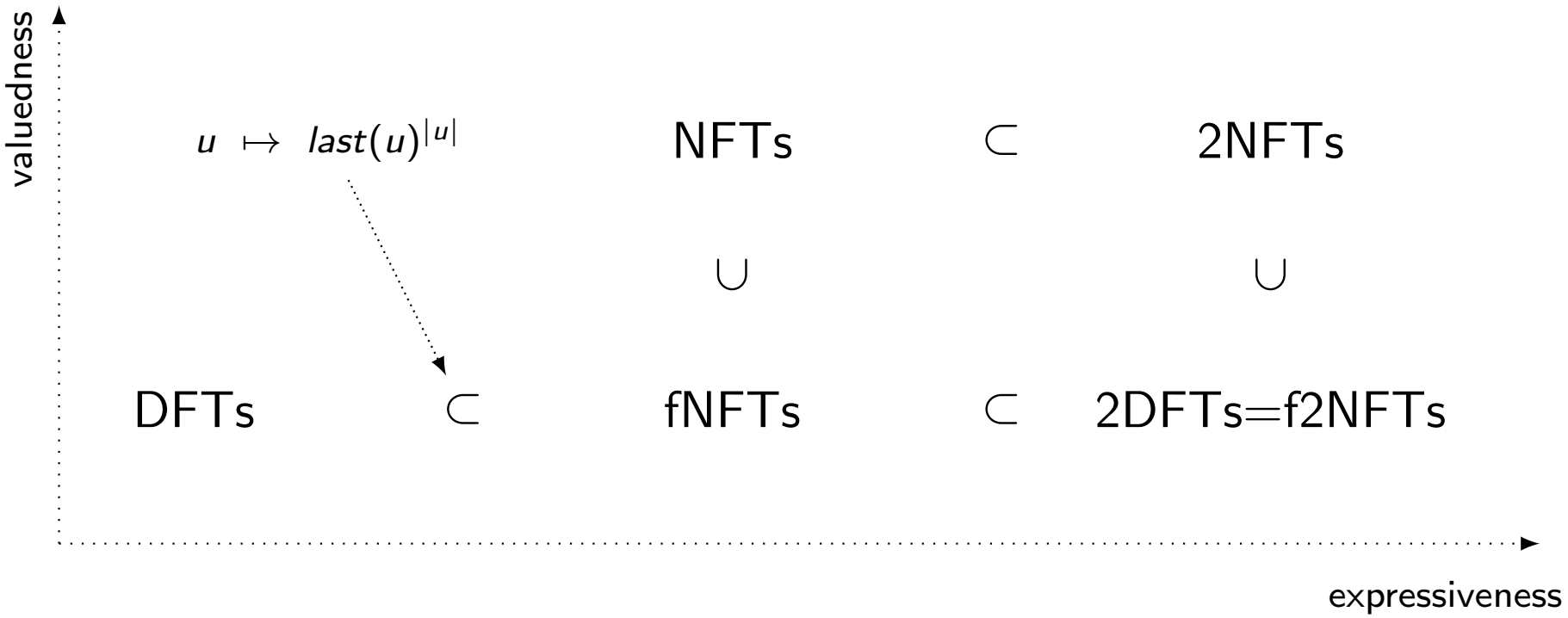
Classes of Transductions



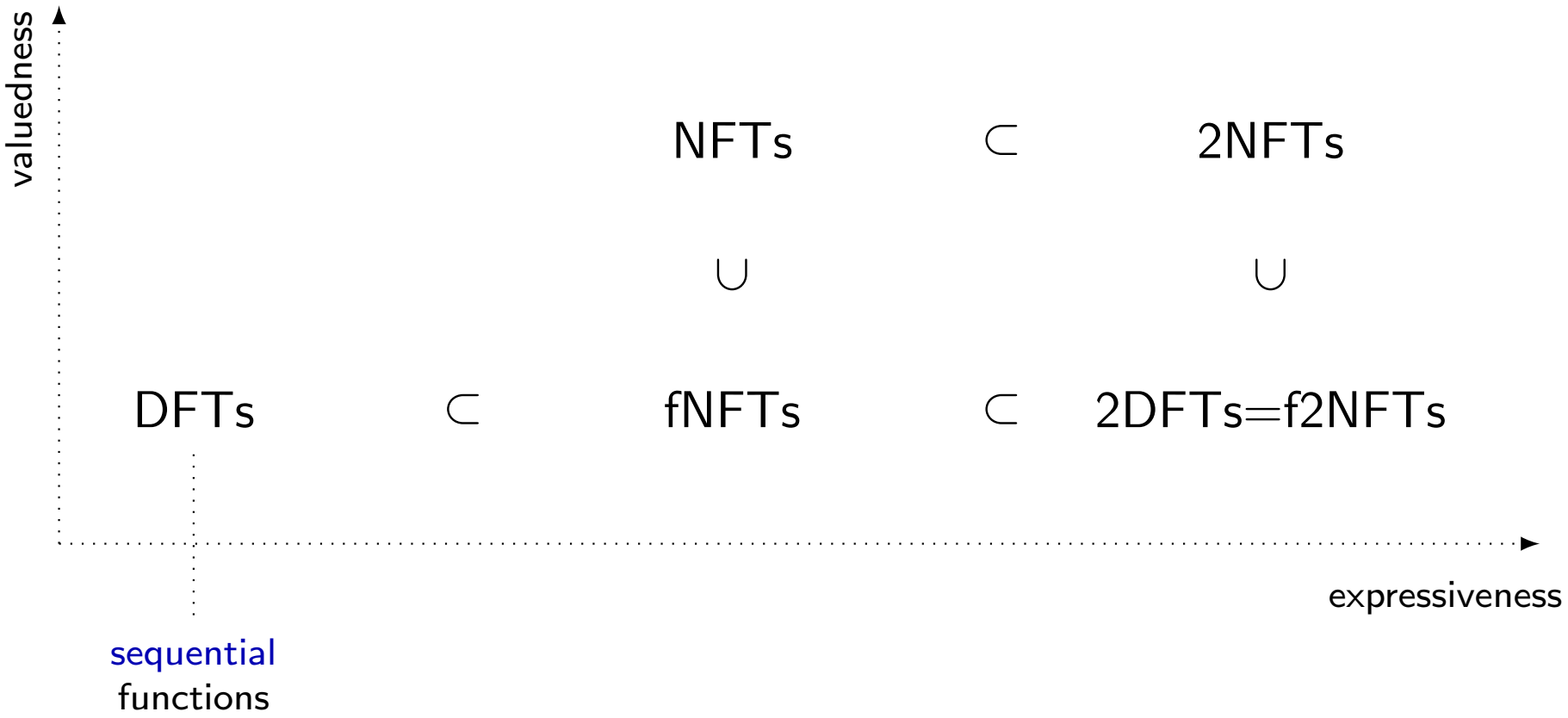
Classes of Transductions



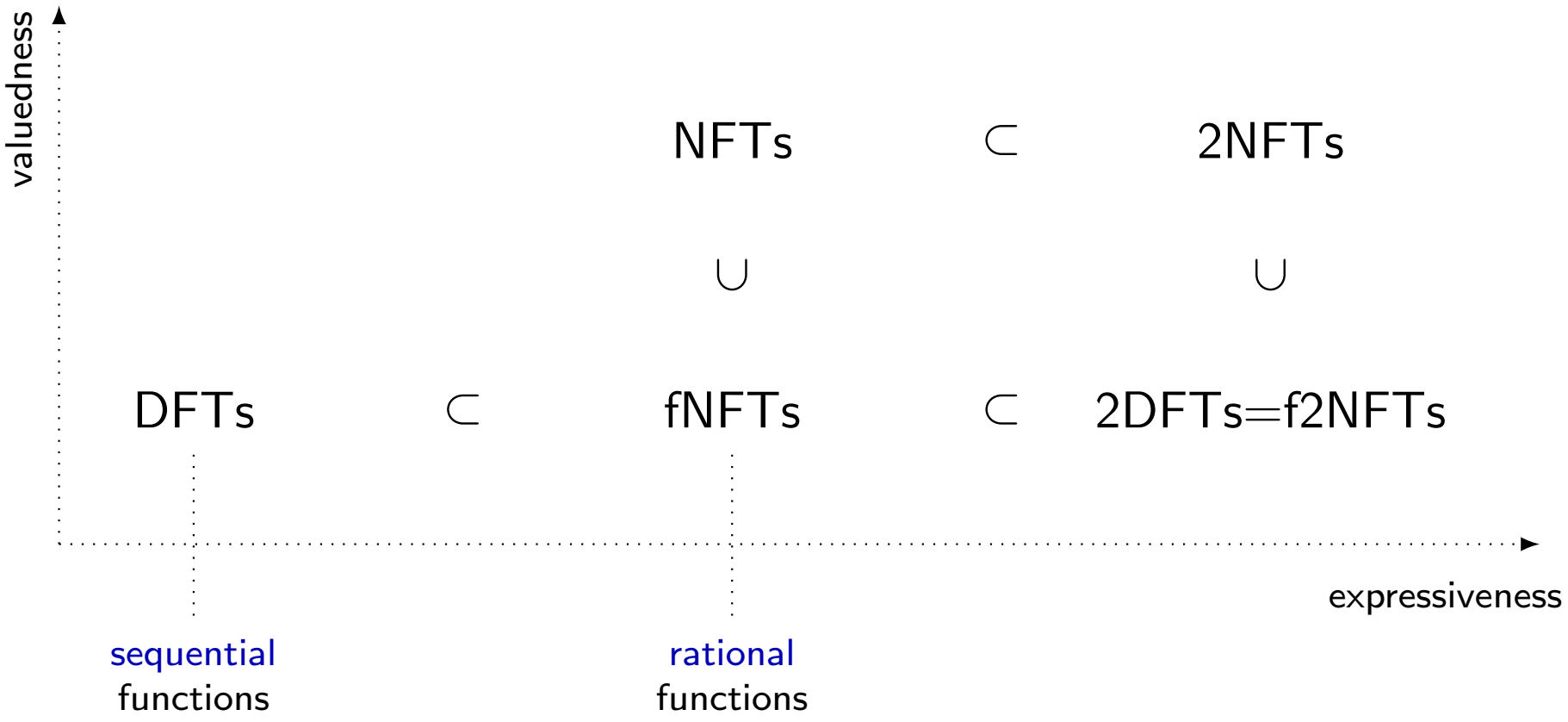
Classes of Transductions



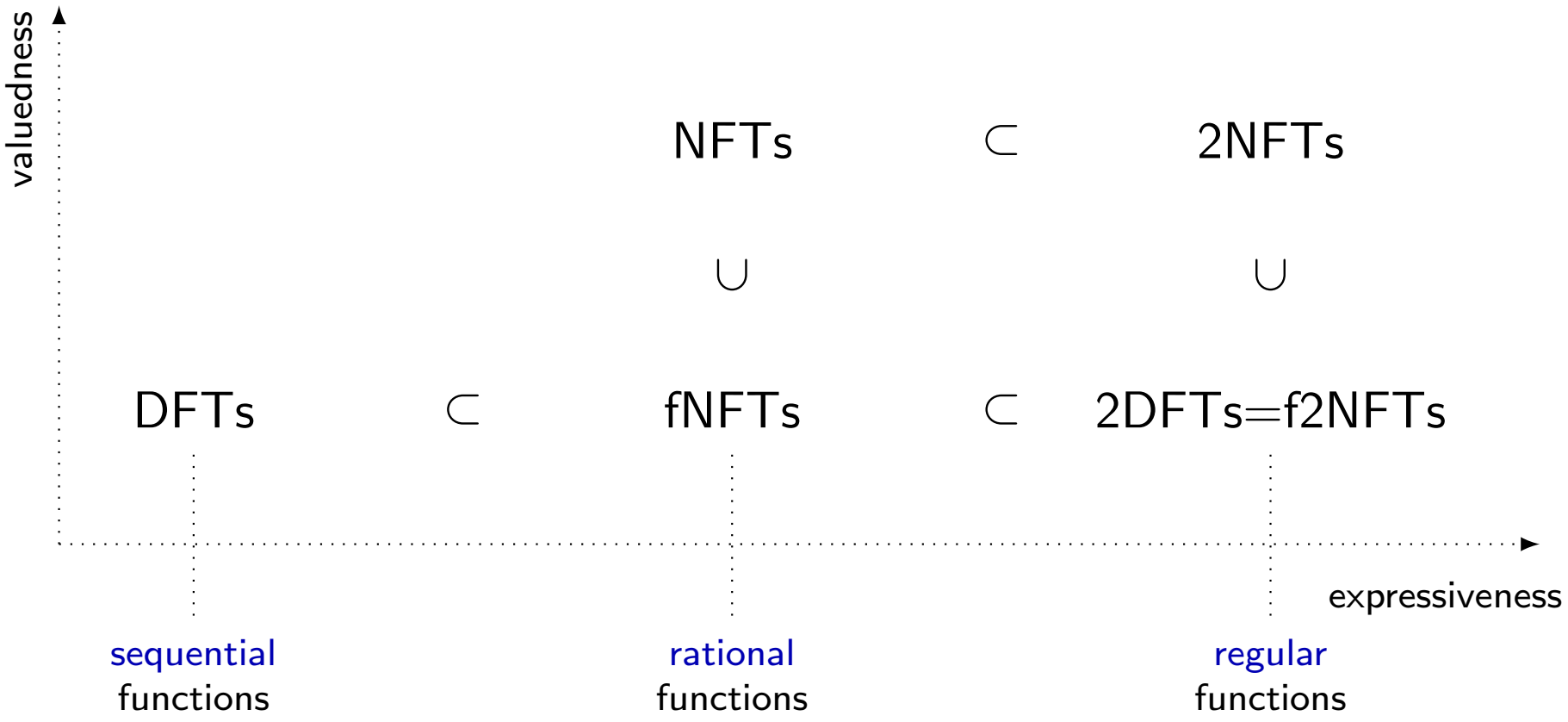
Classes of Transductions



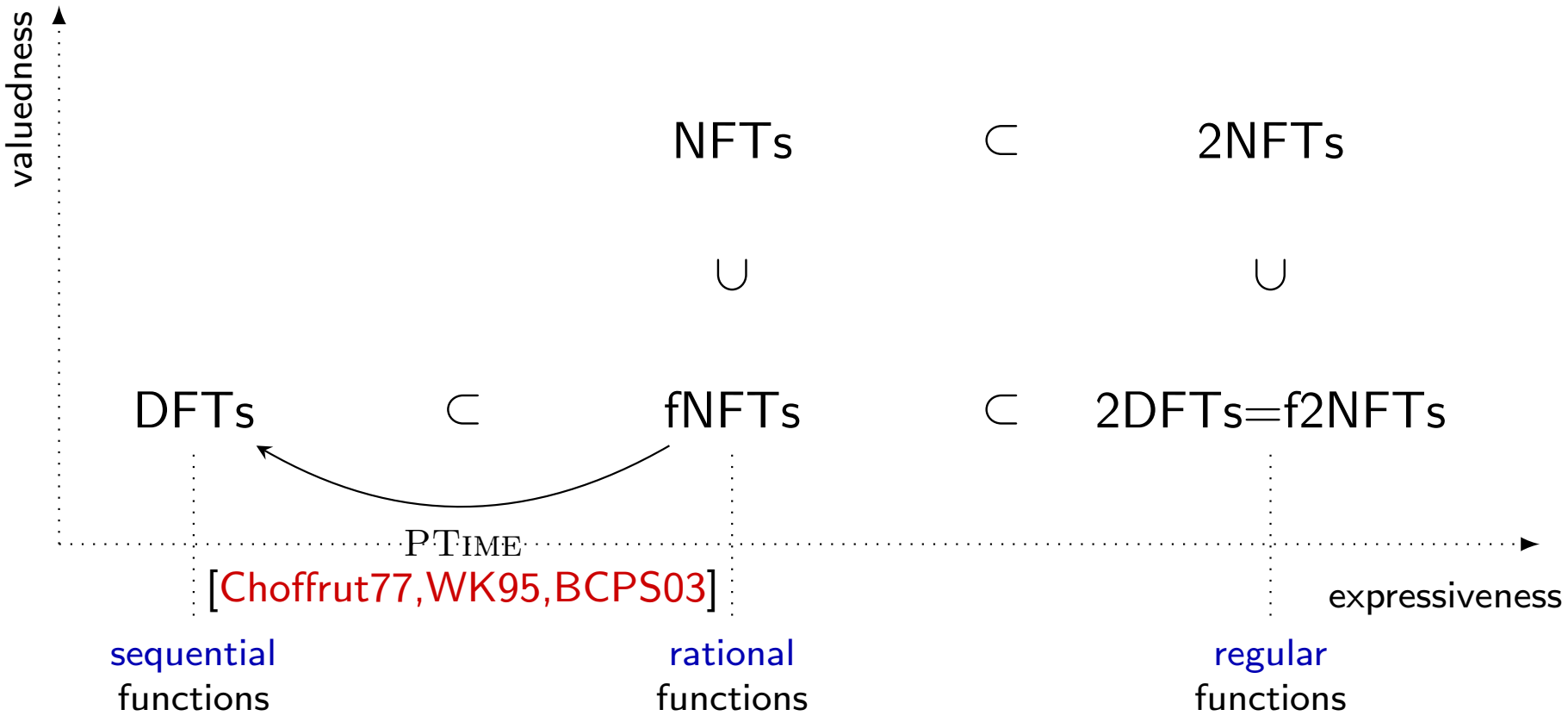
Classes of Transductions



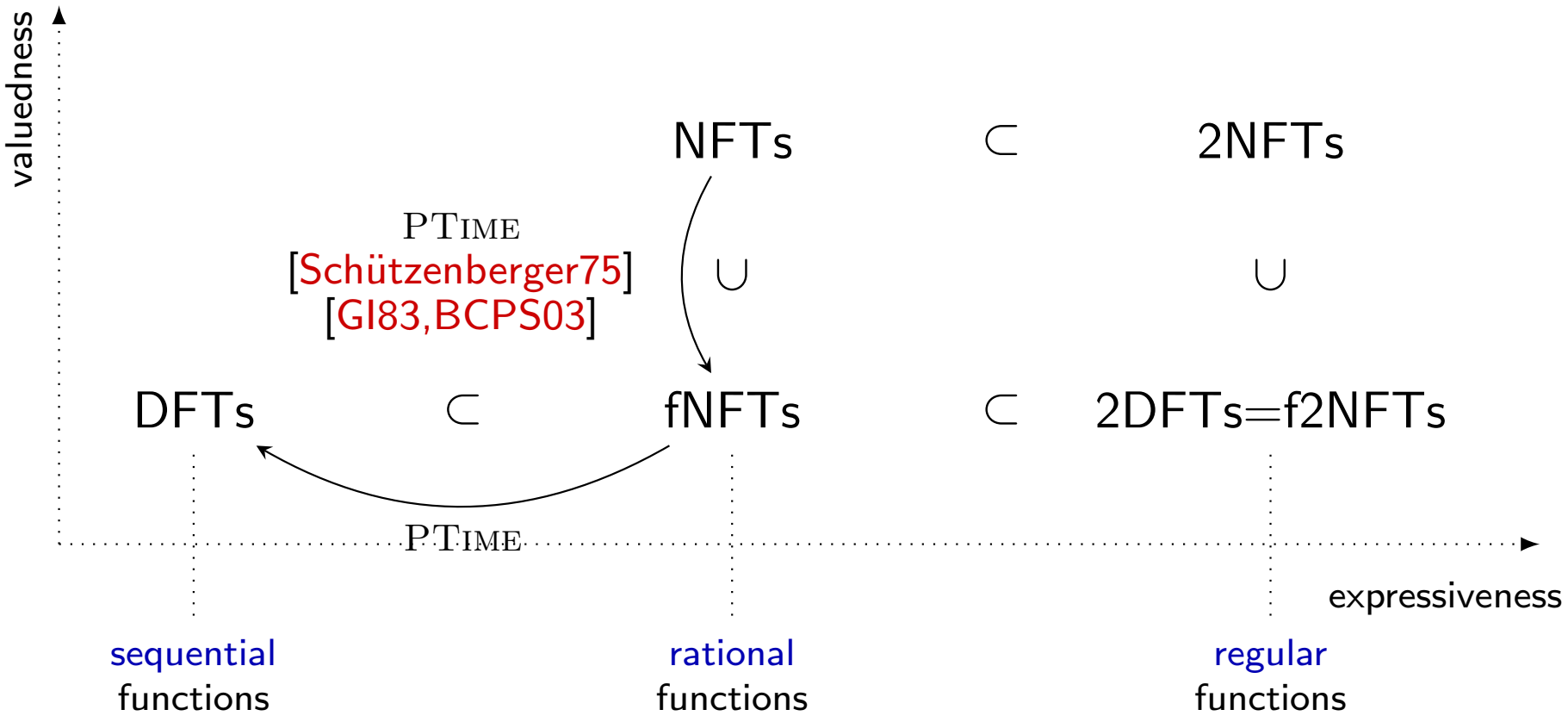
Classes of Transductions



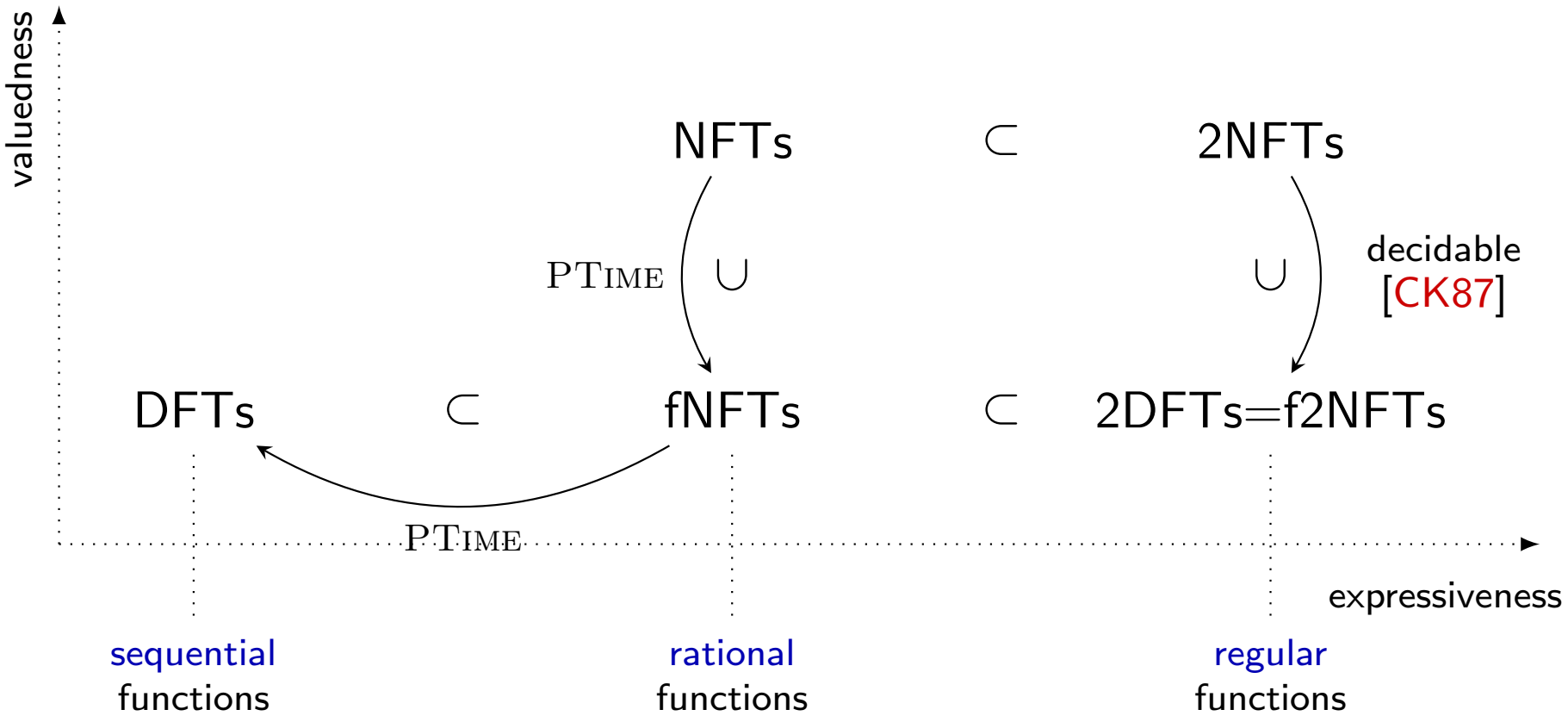
Classes of Transductions



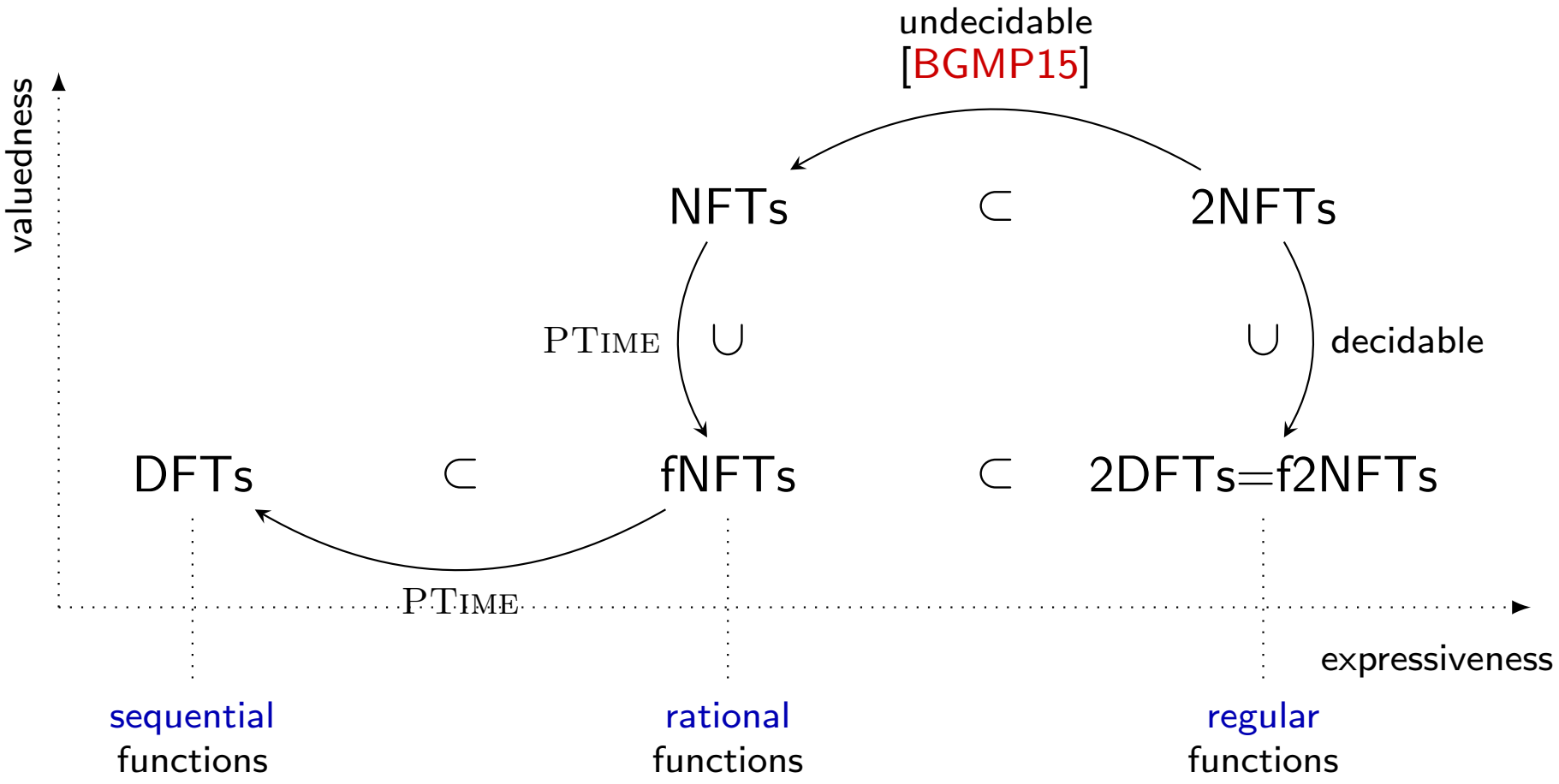
Classes of Transductions



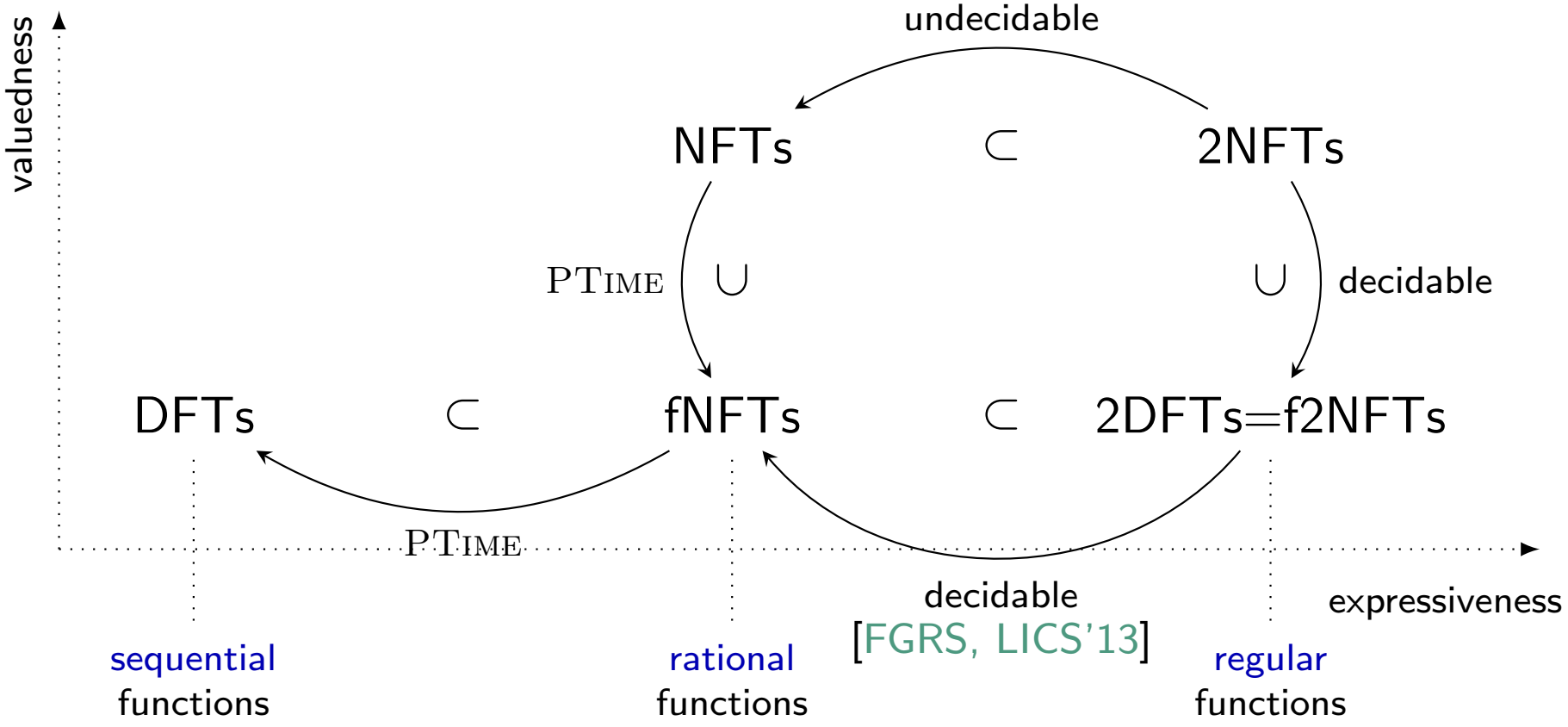
Classes of Transductions



Classes of Transductions



Classes of Transductions



Streaming String Transducers [AC10]

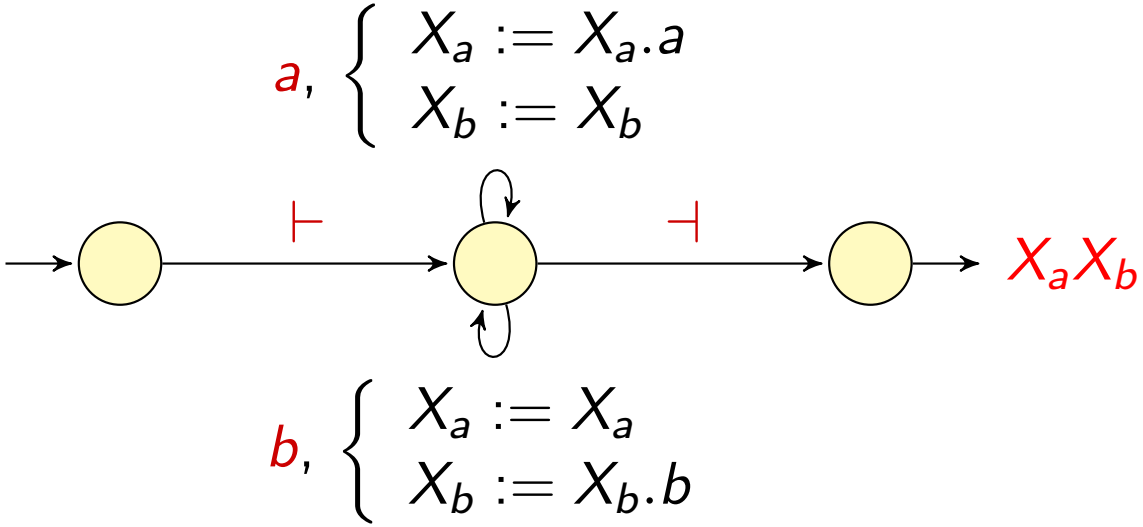
$\vdash w \dashv \;\mapsto\; a^{\#_a(w)} b^{\#_b(w)}$

1-Way DFA + registers

Register updates:

- $X := u \cdot Y \cdot v$
- $X := YZ$

X, Y, Z: registers
u, v: words in Σ^*



Streaming String Transducers [AC10]

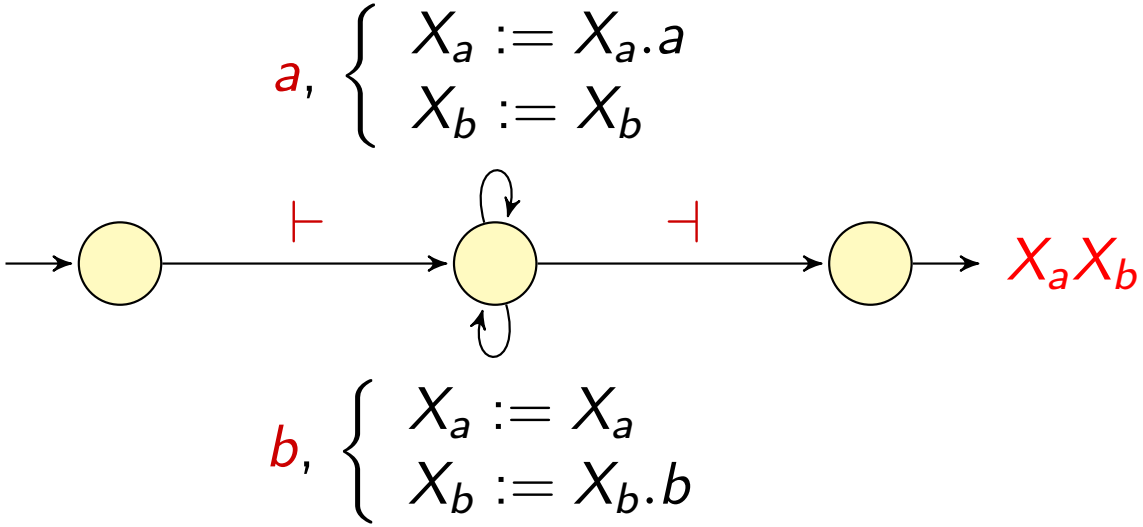
$\vdash w \dashv \;\mapsto\; a^{\#_a(w)} b^{\#_b(w)}$

1-Way DFA + registers

Register updates:

- $X := u \cdot Y \cdot v$
- $X := YZ$

X, Y, Z: registers
u, v: words in Σ^*



Expressiveness results :

- sequential functions: 1-register right appending SST $\quad X := X.a$

Streaming String Transducers [AC10]

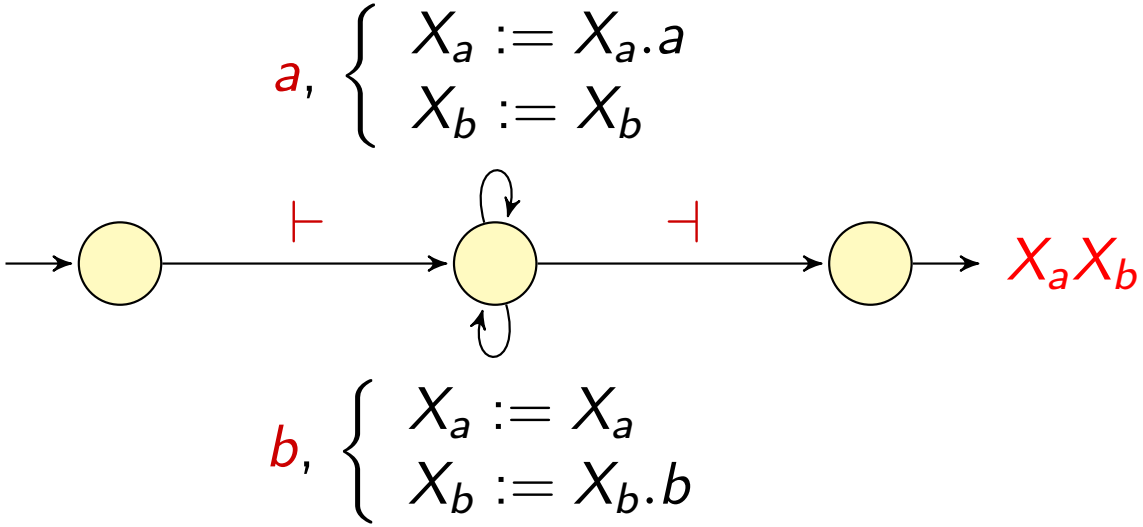
$\vdash w \dashv \ \mapsto a^{\#_a(w)} b^{\#_b(w)}$

1-Way DFA + registers

Register updates:

- $X := u \cdot Y \cdot v$
- $X := YZ$

X, Y, Z: registers
u, v: words in Σ^*



Expressiveness results :

- sequential functions: 1-register right appending SST $X := X.a$
- rational functions: right appending SST $X := Y.a$

Streaming String Transducers [AC10]

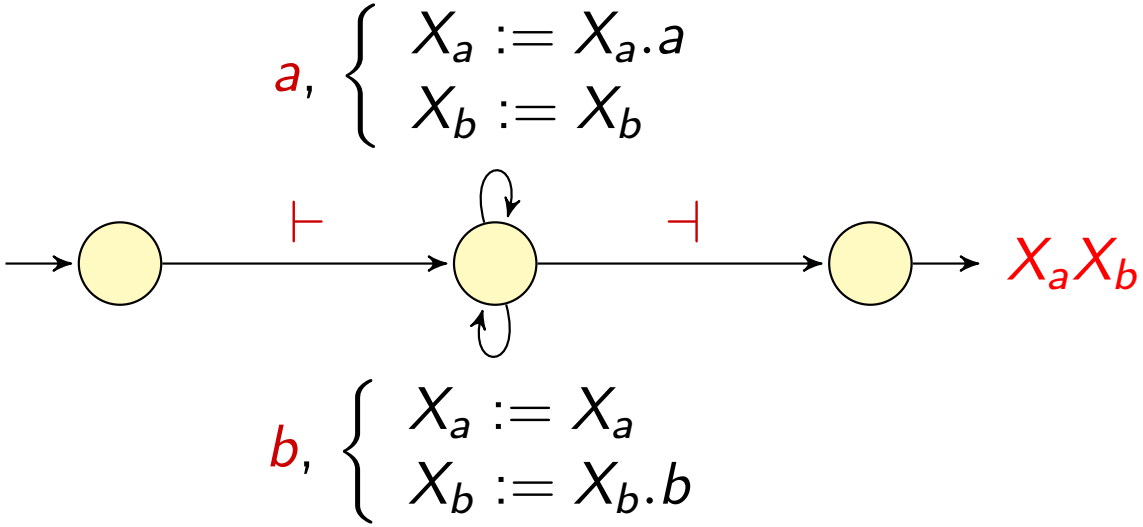
$\vdash w \dashv \ \mapsto a^{\#_a(w)} b^{\#_b(w)}$

1-Way DFA + registers

Register updates:

- $X := u \cdot Y \cdot v$
- $X := YZ$

X, Y, Z: registers
u, v: words in Σ^*



Expressiveness results :

- sequential functions: 1-register right appending SST $X := X.a$
- rational functions: right appending SST $X := Y.a$
- regular functions: copyless SST (X,Y):=(X,X) is forbidden

Overview

From rational to sequential functions [Choffrut77]

Sequentiality Problem

Input: a fNFT
Question: does there exist an equivalent DFT?

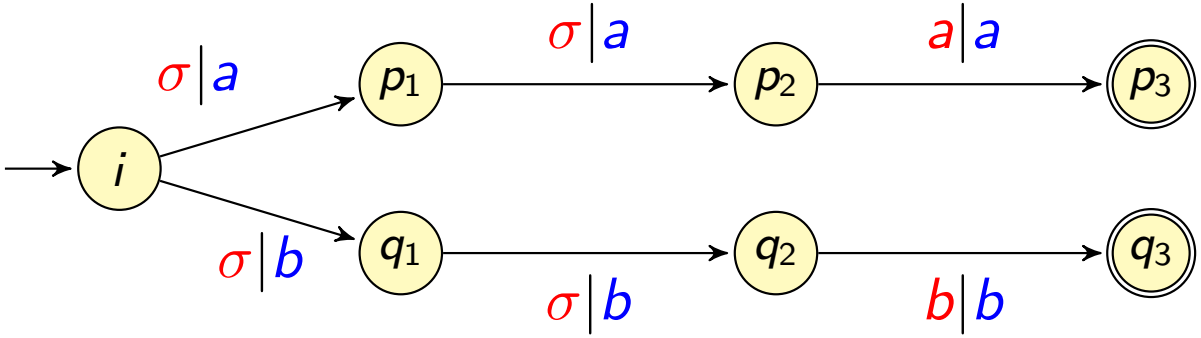
Standard technique:

- subset construction starting from the set of initial states.
- output longest common prefix
- store the unproduced outputs in the state

States of the form $\{(p, a), (q, \varepsilon), (s, bb)\}$

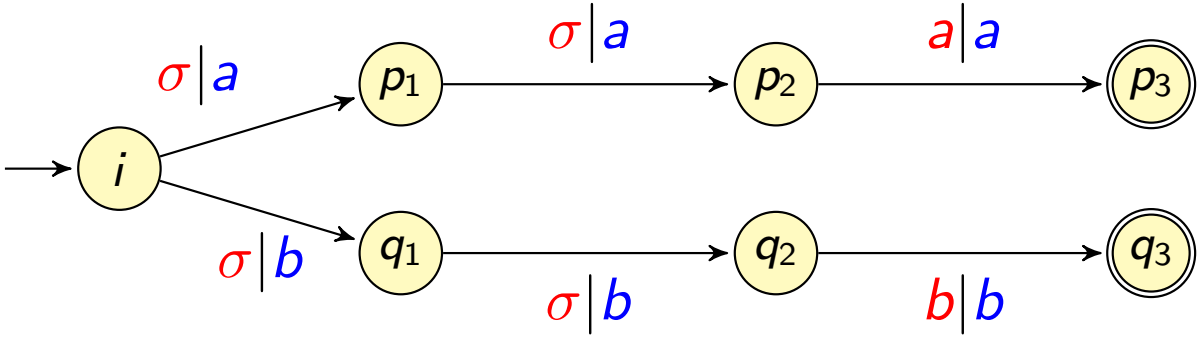
An example

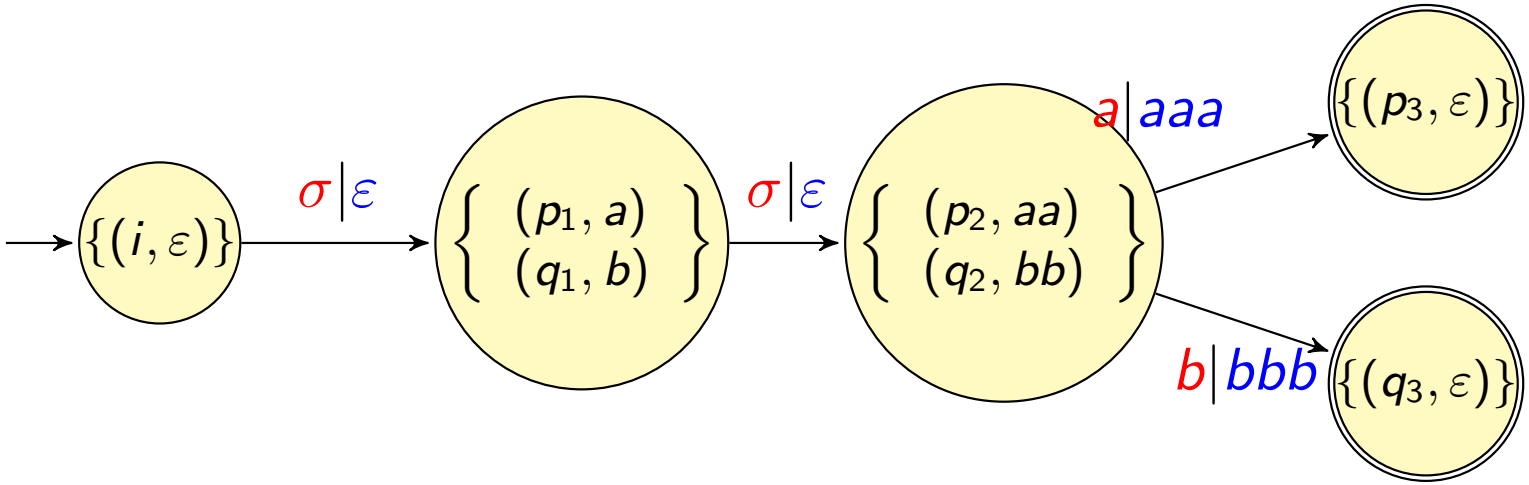
$dom(f) = \Sigma^3$
$f(u) = last(u)^{|u|}$



An example

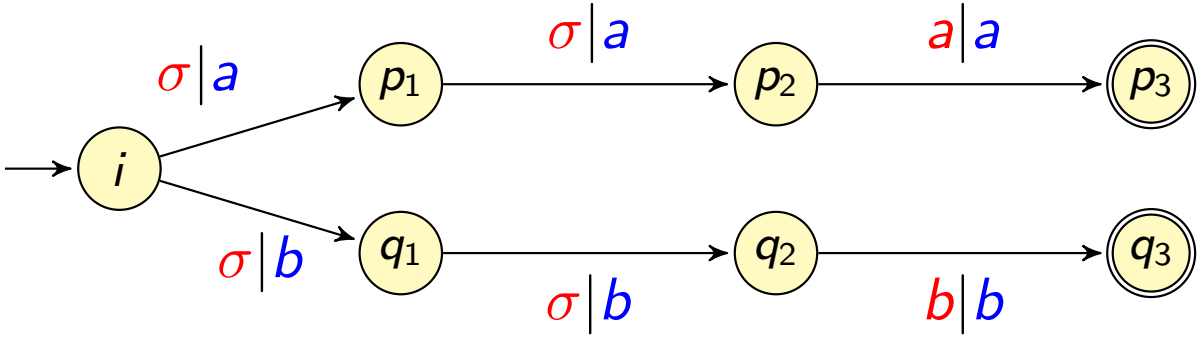
$dom(f) = \Sigma^3$
$f(u) = last(u)^{|u|}$

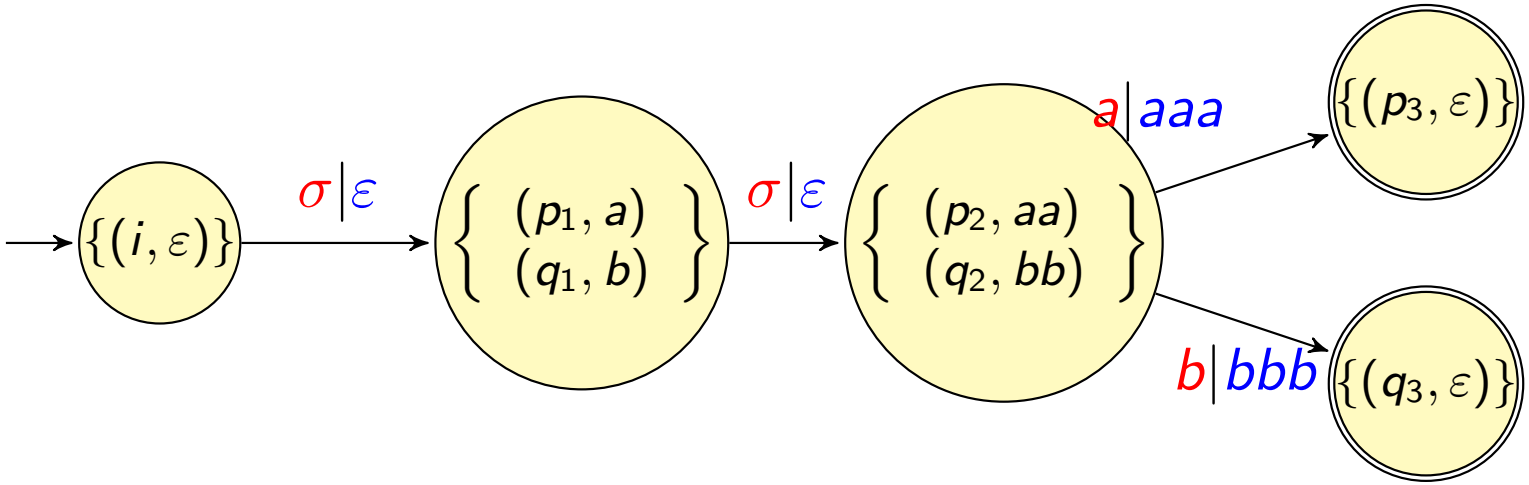




An example

$dom(f) = \Sigma^3$
$f(u) = last(u)^{|u|}$





Goal: characterize termination of subset construction

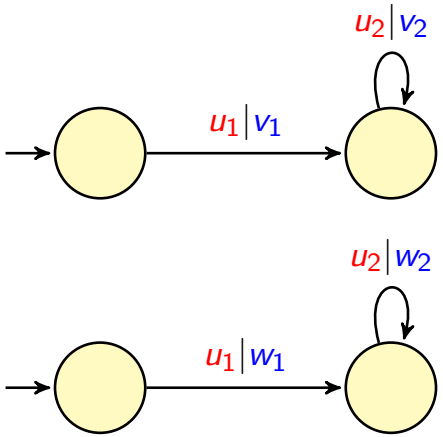
Twinning Property [Choffrut77]

We define:

$$\mathrm{delay}(u, v) = \mathrm{lcp}(u, v)^{-1}.(u, v)$$

Example:
$\mathrm{lcp}(aaa, aab) = aa$
$\mathrm{delay}(aaa, aab) = (a, b)$

For all situations like:



we have $\mathrm{delay}(v_1, w_1) = \mathrm{delay}(v_1 v_2, w_1 w_2)$

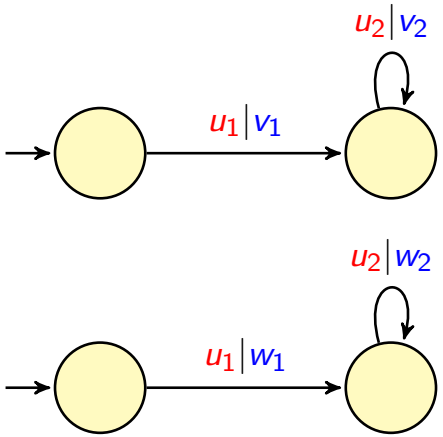
Twinning Property [Choffrut77]

We define:

$$\text{delay}(u, v) = \text{lcp}(u, v)^{-1}.(u, v)$$

Example:
$\text{lcp}(aaa, aab) = aa$
$\text{delay}(aaa, aab) = (a, b)$

For all situations like:



we have $\text{delay}(v_1, w_1) = \text{delay}(v_1 v_2, w_1 w_2)$

Lemma

If a fNFT satisfies the Twinning Property, then the delays computed by the subset construction are bounded.

Corollary

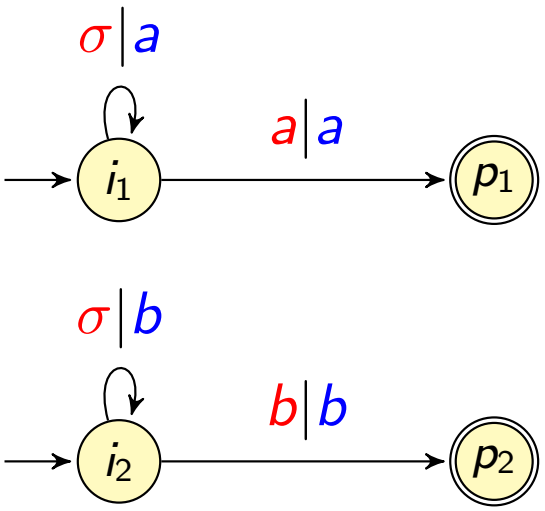
Twinning Property $\iff$ Termination of subset construction.

Theorem ([WK95])

Twinning Property can be decided in PTime.

A counter example

$$dom(f) = \Sigma^*$$
$$f(u) = last(u)^{|u|}$$



After reading an input word u:

- longest common prefix of outputs $= \varepsilon$
- subset construction $= \{(i_1, a^{|u|}),\ (i_2, b^{|u|})\}$

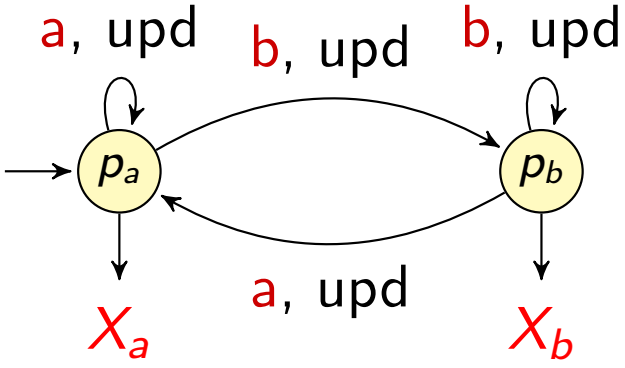
➜ The subset construction does not terminate.

The TP is violated: consider synchronised loops around i_1 and i_2.

Using right-appending streaming string transducers

$$dom(f) = \Sigma^*$$
$$f(u) = last(u)^{|u|}$$

$$\text{upd: } \begin{cases} X_a := X_a.a \\ X_b := X_b.b \end{cases}$$



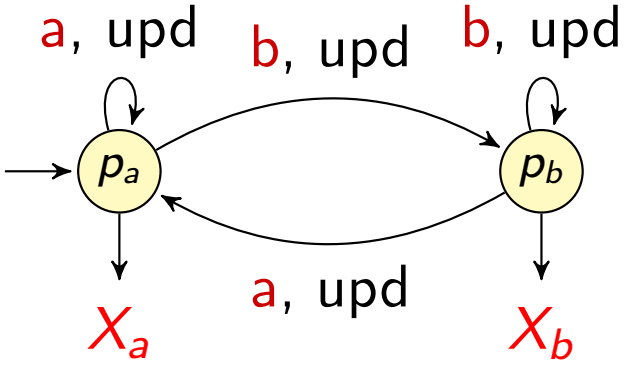
➜ can be realized with 2 registers

Can we do better?

Using right-appending streaming string transducers

$$dom(f) = \Sigma^*$$
$$f(u) = last(u)^{|u|}$$

$$\text{upd: } \begin{cases} X_a := X_a.a \\ X_b := X_b.b \end{cases}$$



➜ can be realized with 2 registers

Can we do better?
No! 1 register is DFT

Register Complexity Problem

Input: A right-appending SST T
Question: Minimal k s.t. there exists a k-raSST T' with $T \equiv T'$

Register complexity using Twinning Property [LICS'16]

Intuition:

2 registers needed if there are 2 runs generating arbitrarily large delays

k registers needed if there are k runs generating pairwise arb. large delays

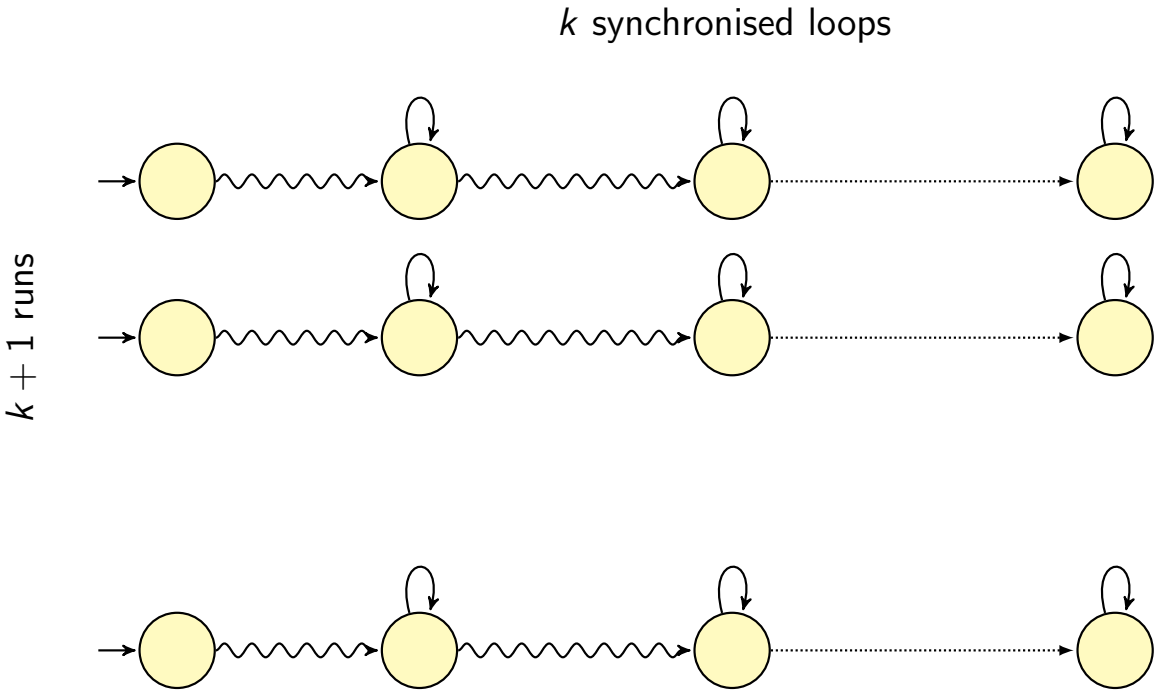
Register complexity using Twinning Property [LICS'16]

Intuition:

2 registers needed if there are 2 runs generating arbitrarily large delays

k registers needed if there are k runs generating pairwise arb. large delays

Contraposition: Twinning Property of order k



➜ there are two runs that remain "close"

Register complexity using Twinning Property [LICS'16]

Lemma

If T satisfies the TP of order k, then from any set of runs on the same input word, one can extract k runs such that every run is "close" from one of these k runs.

Theorem

A fNFT is definable by a k-raSST iff it satisfies the TP of order k.

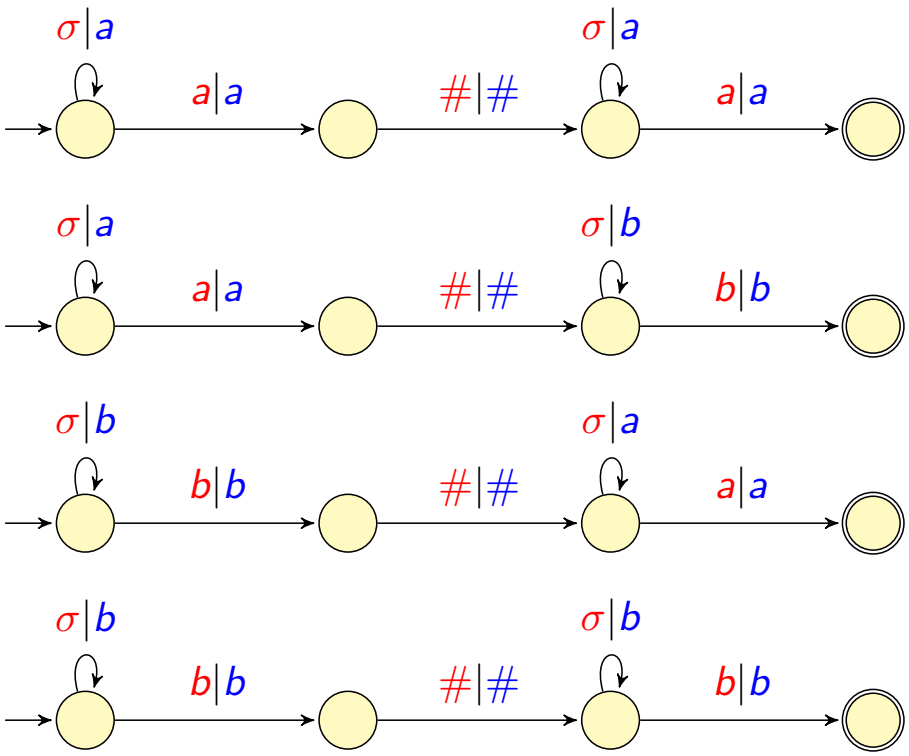
Theorem

Given a fNFT T and k (in unary), deciding whether T satisfies the TP of order k is PSpace-complete.

Register complexity using Twinning Property [LICS'16]

An example: how many registers for the following function?

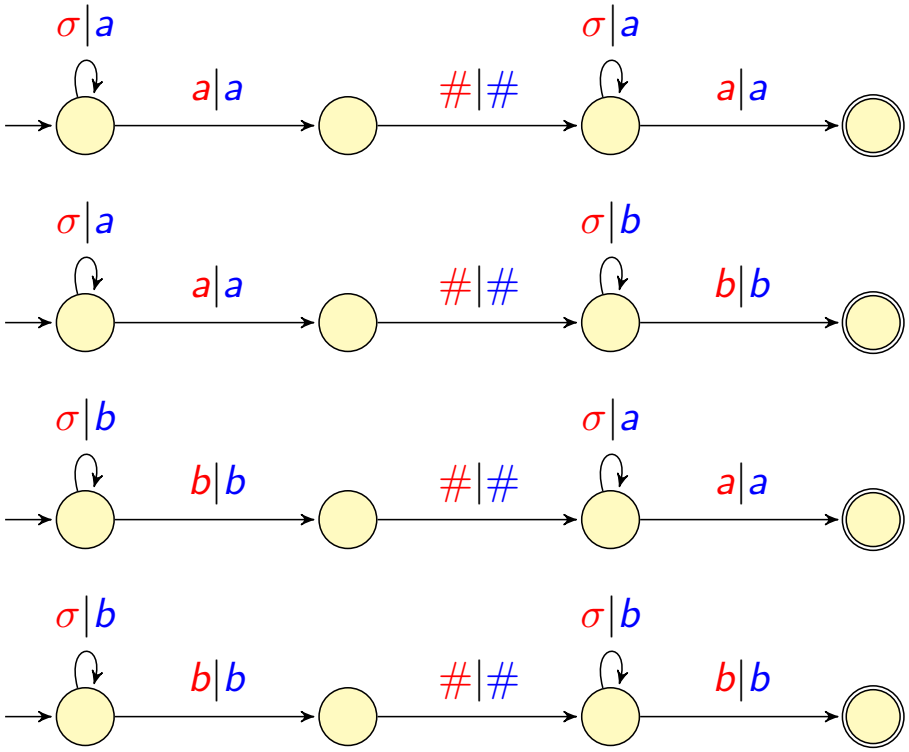
$$f : u_1\#u_2 \mapsto last(u_1)^{|u_1|}\#last(u_2)^{|u_2|}$$



Register complexity using Twinning Property [LICS'16]

An example: how many registers for the following function?

$$f : u_1\#u_2 \mapsto last(u_1)^{|u_1|}\#last(u_2)^{|u_2|}$$

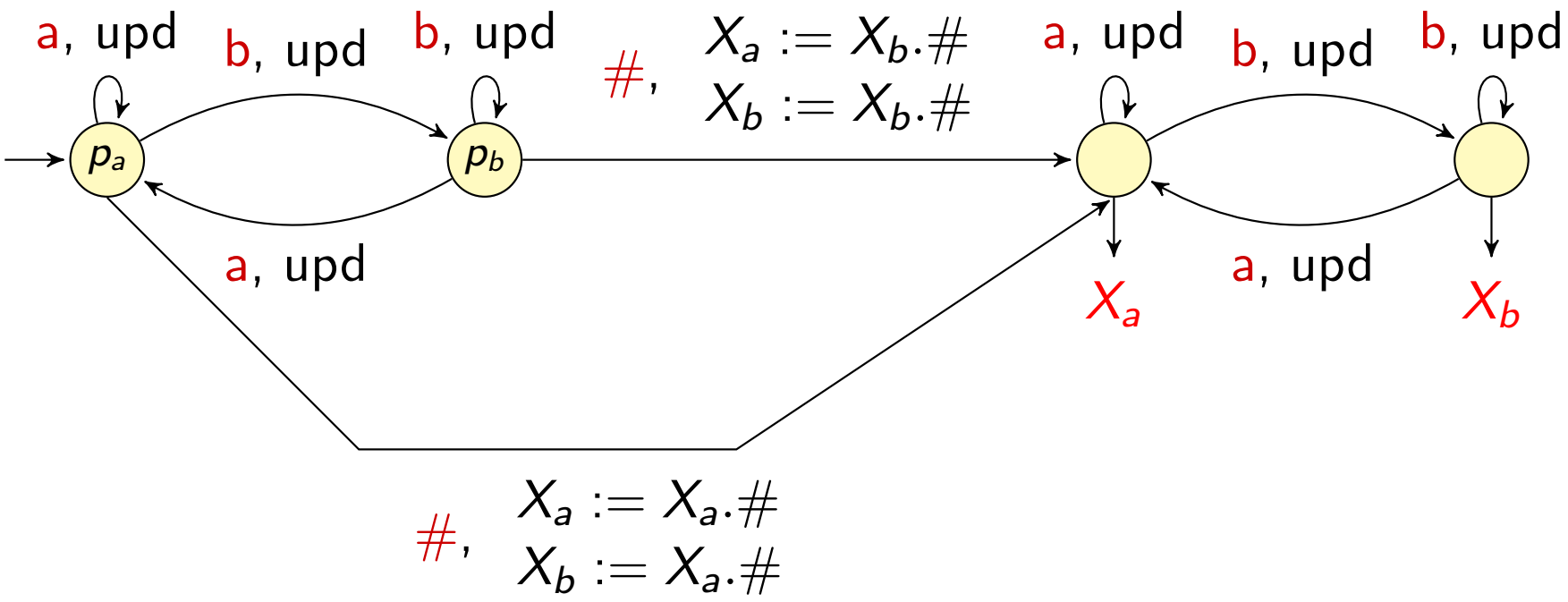


Only 2 registers!

Register complexity using Twinning Property [LICS'16]

An example: how many registers for the following function?

$$f : u_1\#u_2 \mapsto last(u_1)^{|u_1|}\#last(u_2)^{|u_2|}$$



Other patterns: finite valuedness [Weber90]

Definition

A NFT T is finite valued iff $\exists k \mid \forall u \in \Sigma^*, \#[\![T]\!](u) \leq k$.

Theorem

Equivalence of finite valued NFT is decidable.

Other patterns: finite valuedness [Weber90]

Definition

A NFT T is finite valued iff $\exists k \mid \forall u \in \Sigma^*, \#[\![T]\!](u) \leq k$.

Theorem

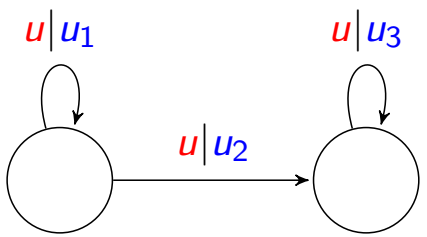
Equivalence of finite valued NFT is decidable.

Theorem

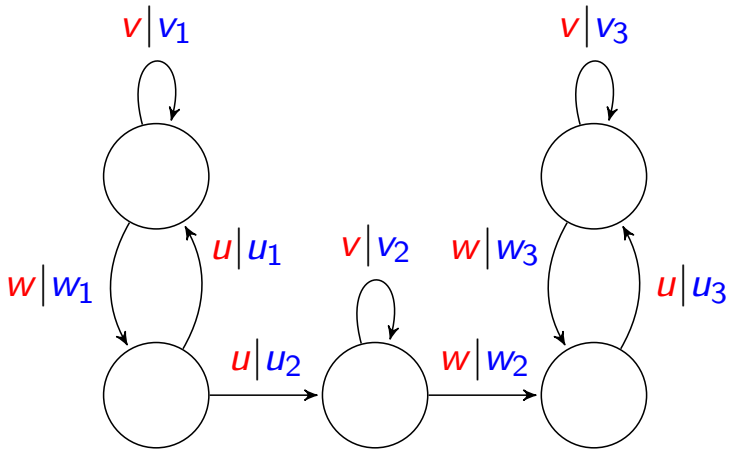
An NFT T is finite-valued iff it satisfies criteria G_0 and G_1.

Theorem

Finite valuedness is decidable in PTime.



we have $u_1 u_2 = u_2 u_3$



we have $|w_1| = |w_2|$

Other patterns: multi-sequential relations [CS86,FJ15]

Definition

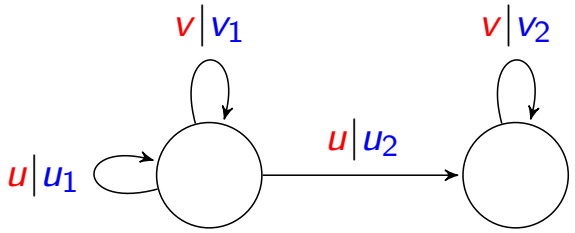
A function/relation R is multi-sequential iff there exists a finite number of DFTs $T_1, \ldots, T_n$ such that $R = \bigcup_i [\![T_i]\!]$

Theorem

An NFT T is multi-sequential iff it satisfies the fork property.

Theorem

Multi sequentiality is decidable in PTime.



we have $\mathrm{delay}(u_1, u_2) = \mathrm{delay}(u_1 v_1, u_2 v_2)$

A perspective about multi-sequential functions/relations

Multi-sequential functions $\equiv$ SST with updates $X := X.a$

Register complexity in this class $\equiv$ Minimization of size of the union

➜ Find the right Twinning Property?

A perspective about multi-sequential functions/relations

Multi-sequential functions $\equiv$ SST with updates $X := X.a$

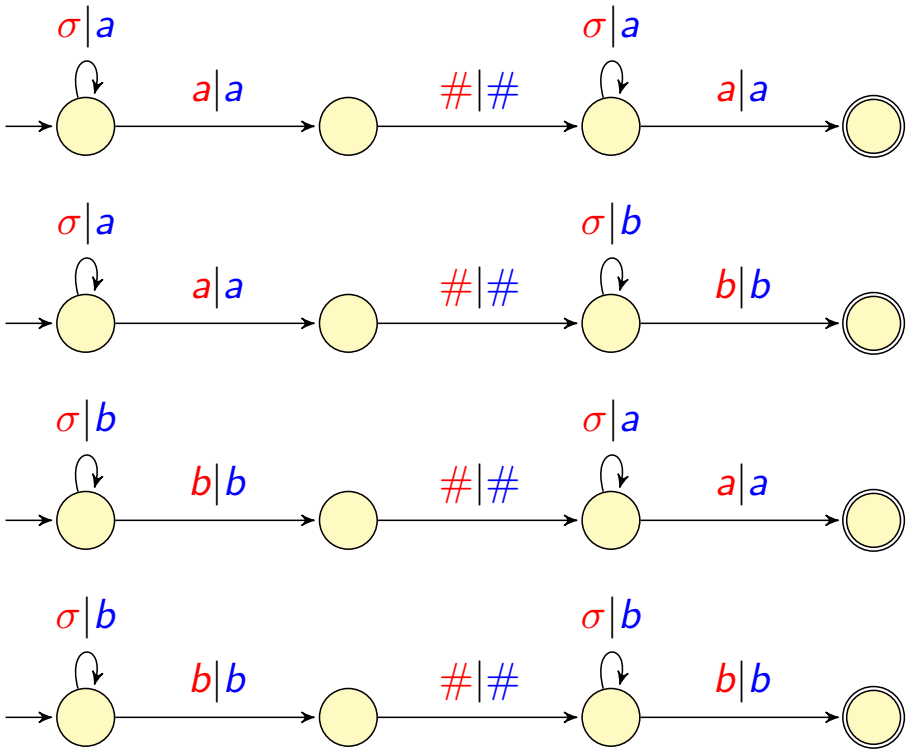
Register complexity in this class $\equiv$ Minimization of size of the union

➜ Find the right Twinning Property?

$$f : u_1 \# u_2 \mapsto$$

$$last(u_1)^{|u_1|} \# last(u_2)^{|u_2|}$$

Minimum=4



Overview

Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists$ *DFA* A *s.t.* $L = L(A)$

Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists$ *DFA* A *s.t.* $L = L(A)$
- $\exists$ *2NFA* A *s.t.* $L = L(A)$

Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists$ *DFA* A *s.t.* $L = L(A)$
- $\exists$ *2NFA* A *s.t.* $L = L(A)$
- $\exists$ *MSO* φ *s.t.* $L = L(\varphi)$

$$MSO \ni \varphi ::= p_a(x) \mid x < y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \wedge \varphi \mid \neg\varphi$$

Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists$ *DFA* A *s.t.* $L = L(A)$
- $\exists$ *2NFA* A *s.t.* $L = L(A)$
- $\exists$ *MSO* φ *s.t.* $L = L(\varphi)$
- *the right congruence* $\approx_L$ *of* L *has finite index*

$$MSO \ni \varphi ::= p_a(x) \mid x < y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \wedge \varphi \mid \neg\varphi$$

$$u \approx_L v \iff \forall w, uw \in L \text{ iff } vw \in L \qquad u^{-1}.L = \{v \mid uv \in L\}$$

Regular languages

Theorem

Let $L \subseteq \Sigma^*$. The following are equivalent:

- $\exists$ *DFA* A *s.t.* $L = L(A)$
- $\exists$ *2NFA* A *s.t.* $L = L(A)$
- $\exists$ *MSO* φ *s.t.* $L = L(\varphi)$
- *the right congruence* $\approx_L$ *of* L *has finite index*
- *the syntactic congruence* $\sim_L$ *has finite index*

$MSO \ni \varphi ::= p_a(x) \mid x < y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \wedge \varphi \mid \neg\varphi$

$u \approx_L v \iff \forall w, uw \in L$ iff $vw \in L$ $\qquad u^{-1}.L = \{v \mid uv \in L\}$

$u \sim_L v \iff \forall w, w', wuw' \in L$ iff $wvw' \in L$

Syntactic monoid: $\Sigma^* / \sim_L$

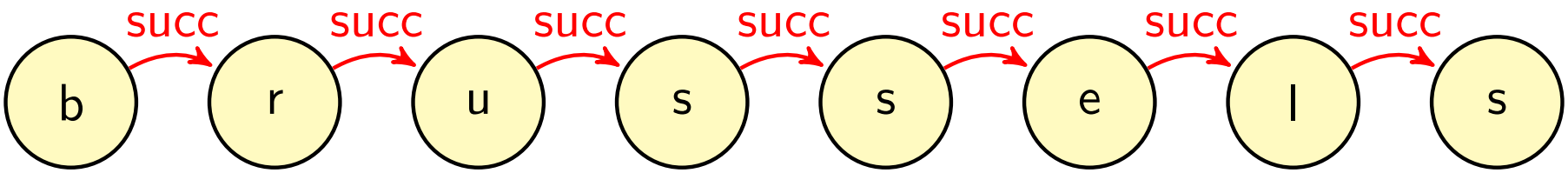
It is isomorphic to the transition monoid of the minimal automaton of L.

MSO Transductions (Courcelle)

- input string seen as the logical structure over $\{succ, (lab_a)_{a \in \Sigma}\}$
- output predicates defined with MSO formulas interpreted over the input structure

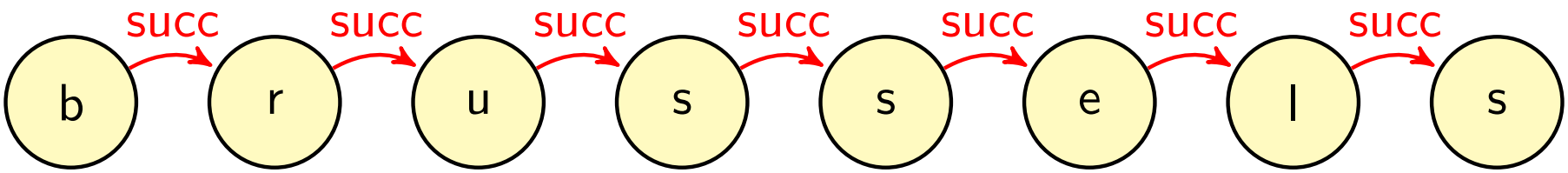
MSO Transductions (Courcelle)

- input string seen as the logical structure over $\{succ, (lab_a)_{a \in \Sigma}\}$
- output predicates defined with MSO formulas interpreted over the input structure



MSO Transductions (Courcelle)

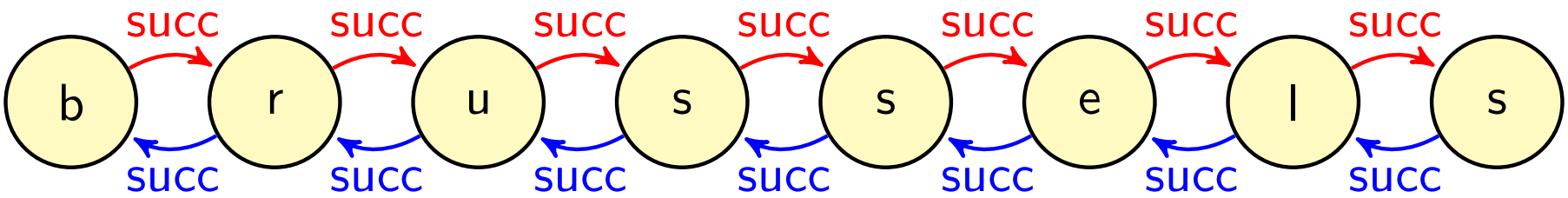
- input string seen as the logical structure over $\{succ, (lab_a)_{a \in \Sigma}\}$
- output predicates defined with MSO formulas interpreted over the input structure



$$\begin{aligned} \phi_{succ}(x, y) &\equiv succ(y, x) \\ \phi_{lab_a}(x) &\equiv lab_a(x) \end{aligned}$$

MSO Transductions (Courcelle)

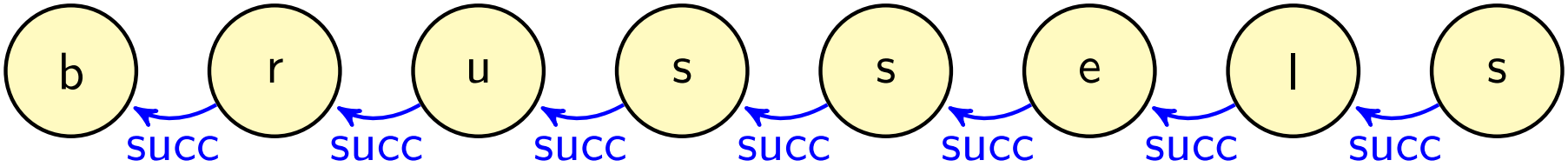
- input string seen as the logical structure over $\{succ, (lab_a)_{a \in \Sigma}\}$
- output predicates defined with MSO formulas interpreted over the input structure



$$\begin{aligned} \phi_{succ}(x, y) &\equiv succ(y, x) \\ \phi_{lab_a}(x) &\equiv lab_a(x) \end{aligned}$$

MSO Transductions (Courcelle)

- input string seen as the logical structure over $\{succ, (lab_a)_{a \in \Sigma}\}$
- output predicates defined with MSO formulas interpreted over the input structure



$$\phi_{succ}(x, y) \equiv succ(y, x)$$
$$\phi_{lab_a}(x) \equiv lab_a(x)$$

A Buchi theorem for transductions

Theorem ([Engelfriet,Hoogeboom,01])

2DFT and MSOT are (effectively) expressively equivalent.

A Buchi theorem for transductions

Theorem ([Engelfriet,Hoogeboom,01])

2DFT and MSOT are (effectively) expressively equivalent.

Emptiness

- Given an MSOT ϕ, does $[\![\phi]\!] = \varnothing$ hold ?
  1. Transform ϕ into a 2DFT T
  2. Check whether $dom(T) = \varnothing$

A Buchi theorem for transductions

Theorem ([Engelfriet,Hoogeboom,01])

2DFT and MSOT are (effectively) expressively equivalent.

Emptiness

- Given an MSOT ϕ, does $[\![\phi]\!] = \varnothing$ hold ?
  1. Transform ϕ into a 2DFT T
  2. Check whether $dom(T) = \varnothing$

Equivalence

- Given two MSOT ϕ_1, ϕ_2, does $[\![\phi_1]\!] = [\![\phi_2]\!]$ hold ?
  1. Transform ϕ_1, ϕ_2 into 2DFT T_1, T_2
  2. Check equivalence of T_1 and T_2

Order-preserving MSOT

- no backward edges in the MSO graph.
- e.g. reverse is not order-preserving
- syntactic presentation: guard formula $\phi_{succ}(x, y)$ by $x \leq y$.

Order-preserving MSOT

- no backward edges in the MSO graph.
- e.g. reverse is not order-preserving
- syntactic presentation: guard formula $\phi_{succ}(x, y)$ by $x \leq y$.

Theorem ([Bojanczyk14,Filiot15])

Rational functions coincide with order-preserving MSO transductions.

As a corollary of 1-Way definability of 2-Way transducers, we have:

Theorem

It is decidable, given an MSOT ϕ, whether ϕ is equivalent to some order-preserving MSOT.

Sequential functions [Choffrut 03]

Key: consider earliest transducers (produce output asap)

What can be produced after reading input u?

$$\hat{f}(u) = \mathrm{lcp}(\{f(uv) \mid uv \in dom(f)\})$$

Sequential functions [Choffrut 03]

Key: consider earliest transducers (produce output asap)

What can be produced after reading input u?

$$\hat{f}(u) = \mathrm{lcp}(\{f(uv) \mid uv \in dom(f)\})$$

Define $u \approx_f v \iff \begin{cases} u^{-1}.dom(f) = v^{-1}.dom(f) \\ \forall w, \hat{f}(u)^{-1}.f(uw) = \hat{f}(v)^{-1}.f(vw) \end{cases}$

➜ same behaviors after u and v

Theorem

$f : \Sigma^* \to \Sigma^*$ is sequential iff $\approx_f$ has finite index.

Sequential functions [Choffrut 03]

Key: consider earliest transducers (produce output asap)

What can be produced after reading input u?

$$\hat{f}(u) = \mathrm{lcp}(\{f(uv) \mid uv \in dom(f)\})$$

Define $u \approx_f v \iff \begin{cases} u^{-1}.dom(f) = v^{-1}.dom(f) \\ \forall w, \hat{f}(u)^{-1}.f(uw) = \hat{f}(v)^{-1}.f(vw) \end{cases}$

➜ same behaviors after u and v

Theorem

$f : \Sigma^* \to \Sigma^*$ is sequential iff $\approx_f$ has finite index.

Classes of $\approx_f$ can be used to define a minimal DFT:

$$[u]_{\approx_f} \xrightarrow{a|v} [ua]_{\approx_f} \text{ where } v = \hat{f}(u)^{-1}.\hat{f}(ua)$$

Rational functions [Reutenauer, Schutzenberger 01]

Function $f : u \to last(u)^{|u|}$ is not sequential

➜ needs some information about the suffix of the input word (look-ahead)

- may be infinite: $f : u_1\#u_2\#u_3 \ldots \mapsto last(u_1)^{|u_1|}\#last(u_2)^{|u_2|}\# \ldots$
- but not arbitrary: mirror image

Rational functions [Reutenauer, Schutzenberger 01]

Function $f : u \to last(u)^{|u|}$ is not sequential

➜ needs some information about the suffix of the input word (look-ahead)

- may be infinite: $f : u_1\#u_2\#u_3 \ldots \mapsto last(u_1)^{|u_1|}\#last(u_2)^{|u_2|}\# \ldots$
- but not arbitrary: mirror image

Original left congruence: $u \leftharpoonup_f v$ iff

- $dom(f).u^{-1} = dom(f).v^{-1}$
- Functions $w \mapsto f(wu)$ and $w \mapsto f(wv)$ are "adjacent"

Adjacent: $sup_w |\text{delay}(f(wu), f(wv))| < +\infty$

Rational functions [Reutenauer, Schutzenberger 01]

Function $f : u \to last(u)^{|u|}$ is not sequential

➜ needs some information about the suffix of the input word (look-ahead)

- may be infinite: $f : u_1\#u_2\#u_3 \ldots \mapsto last(u_1)^{|u_1|}\#last(u_2)^{|u_2|}\# \ldots$
- but not arbitrary: mirror image

Original left congruence: $u \leftarrow_f v$ iff

- $dom(f).u^{-1} = dom(f).v^{-1}$
- Functions $w \mapsto f(wu)$ and $w \mapsto f(wv)$ are "adjacent"

Theorem

$f : \Sigma^* \to \Sigma^*$ is rational iff $\leftarrow_f$ has finite index.

Rational functions [Reutenauer, Schutzenberger 01]

Function $f : u \to last(u)^{|u|}$ is not sequential

➜ needs some information about the suffix of the input word (look-ahead)

- may be infinite: $f : u_1\#u_2\#u_3 \ldots \mapsto last(u_1)^{|u_1|}\#last(u_2)^{|u_2|}\# \ldots$
- but not arbitrary: mirror image

Original left congruence: $u \leftarrow_f v$ iff

- $dom(f).u^{-1} = dom(f).v^{-1}$
- Functions $w \mapsto f(wu)$ and $w \mapsto f(wv)$ are "adjacent"

Theorem

$f : \Sigma^* \to \Sigma^*$ is rational iff $\leftarrow_f$ has finite index.

In addition, definition of a canonical transducer.

Rational functions: an example

Function $f : u \to last(u)^{|u|}$ with $\Sigma = \{a, b\}$

Classes of $\leftharpoonup_f$: $[\varepsilon] = \{\varepsilon\}$ $\quad [a] = \Sigma^* a$ $\quad [b] = \Sigma^* b$

Indeed, we have $xa \leftharpoonup_f ya$:

$$f(wxa) = a^{|w|+|x|+1} \quad \text{and} \quad f(wya) = a^{|w|+|y|+1}$$

➜ delay between $f(wxa)$ and $f(wya)$ does not depend on w

Rational functions: an example

Function $f : u \to last(u)^{|u|}$ with $\Sigma = \{a, b\}$

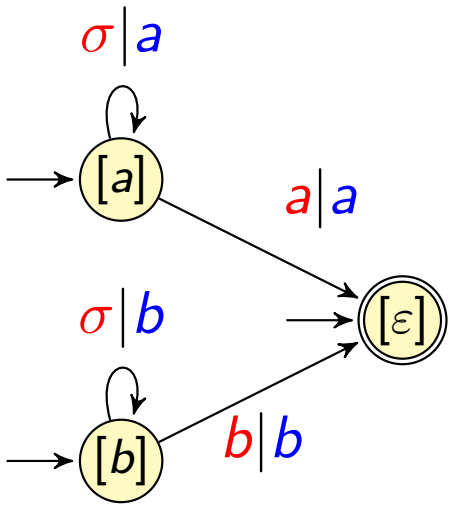
Classes of $\leftharpoonup_f$: $[\varepsilon] = \{\varepsilon\}$ $\quad [a] = \Sigma^* a$ $\quad [b] = \Sigma^* b$

Indeed, we have $xa \leftharpoonup_f ya$:

$$f(wxa) = a^{|w|+|x|+1} \quad \text{and} \quad f(wya) = a^{|w|+|y|+1}$$

➜ delay between $f(wxa)$ and $f(wya)$ does not depend on w

The fNFT guesses at each step the class (for $\leftharpoonup_f$) of the suffix



Regular functions

No algebraic presentation exists (yet).

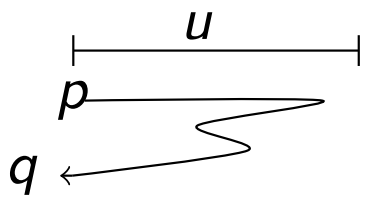
But one can define the transition monoid of a transducer.

Regular functions

No algebraic presentation exists (yet).

But one can define the transition monoid of a transducer.

For a two-way transducer T, we define left-to-left runs of T: $Runs_{LL}(u)$=pairs (p, q) such that:



Similarly for left-to-right, right-to-left...

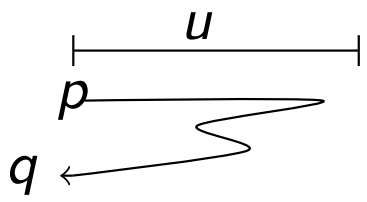
Congruence of the transition monoid:
Define $u \sim_T v \iff \forall x, y \in \{L, R\}, Runs_{xy}(u) = Runs_{xy}(v)$

Regular functions

No algebraic presentation exists (yet).

But one can define the transition monoid of a transducer.

For a two-way transducer T, we define left-to-left runs of T: $Runs_{LL}(u)$=pairs (p, q) such that:



Similarly for left-to-right, right-to-left...

Congruence of the transition monoid:
Define $u \sim_T v \iff \forall x, y \in \{L, R\}, Runs_{xy}(u) = Runs_{xy}(v)$

Does not depend on the outputs!

Can also be defined for SST (taking into account the register updates)

First-Order definable transformations

Theorem (First-Order definable languages)

Given a language $L \subseteq \Sigma^*$, the following are equivalent:

- *L is definable in $FO(<, \{p_a, a \in \Sigma\})$*
- *L is accepted by an automaton whose transition monoid is aperiodic*
- *L is star-free*
- *L is definable in LTL (linear temporal logic)*

Aperiodic monoid M: $\exists n \mid \forall m \in M, n^n = m^{n+1}$

First-Order definable transformations

Theorem ([FGL16])

Let T be a fNFT. Then $[\![T]\!]$ is order-pres. FOT definable iff its canonical transducer is aperiodic.

Corollary

Order-preserving FOT is decidable among rational functions.

First-Order definable transformations

Theorem ([FGL16])

Let T be a fNFT. Then $[\![T]\!]$ is order-pres. FOT definable iff its canonical transducer is aperiodic.

Corollary

Order-preserving FOT is decidable among rational functions.

Theorem ([FKT14,CD15,DJR16])

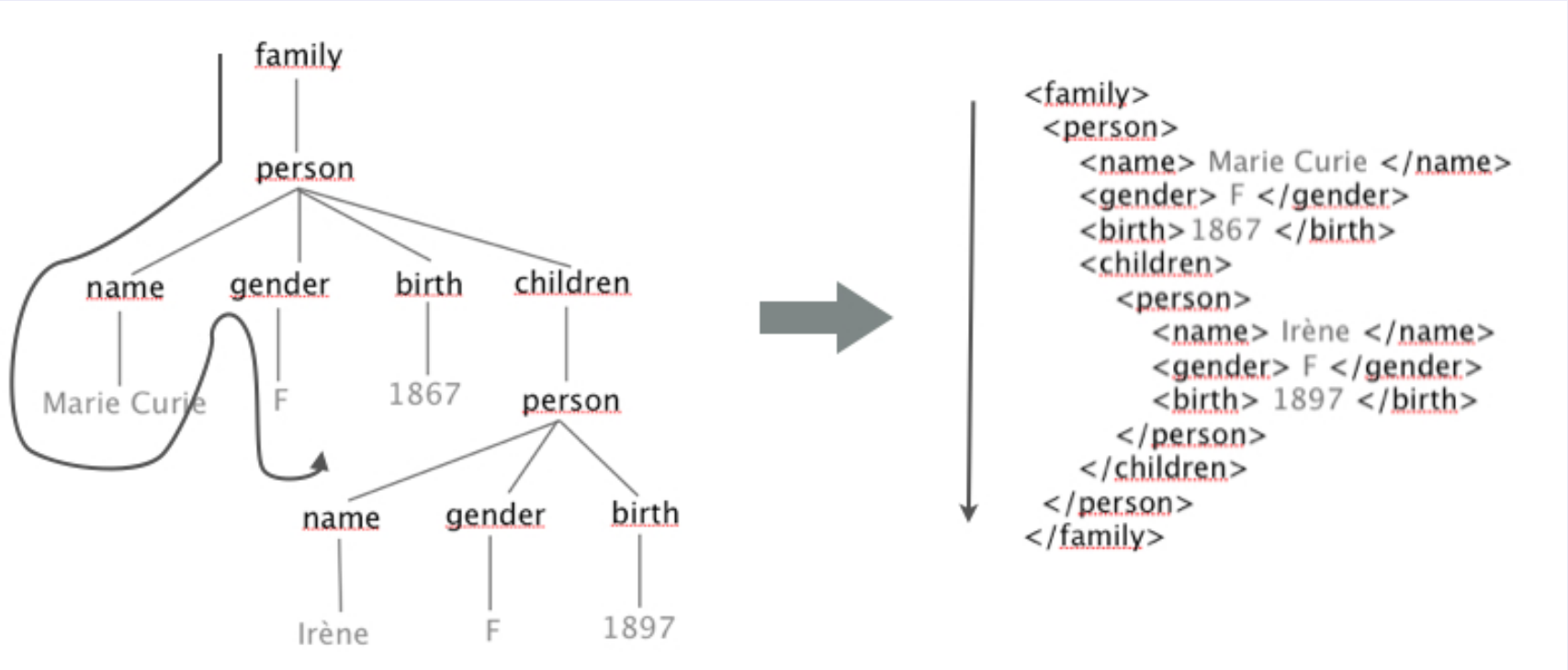
Let $f : \Sigma^* \to \Sigma^*$. The following are equivalent:

- *f is FOT-definable*
- *f is definable by an aperiodic 2DFT*
- *f is definable by an aperiodic copyless SST*

Overview

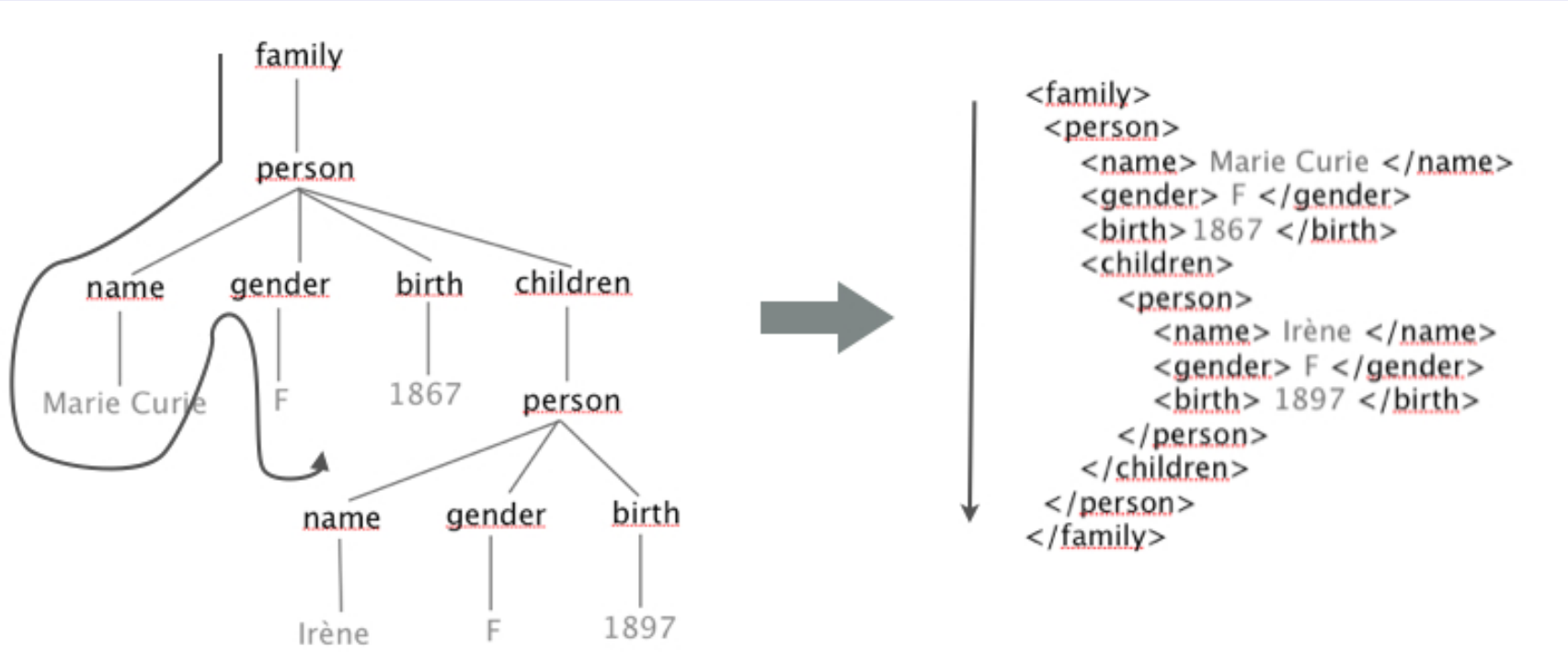
Motivation: Streaming XML Transformations

XML document ≡ tree linearization



Motivation: Streaming XML Transformations

XML document ≡ tree linearization



XML documents are words with a nesting structure:

Structured alphabet: call symbols, return symbols, internal symbols

Visibly Pushdown Automata (VPAs) [AM04]

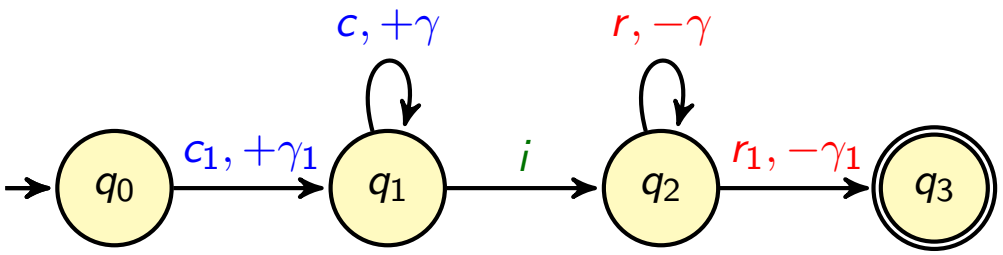
VPAs = Pushdown Automata on *structured* alphabet $\Sigma = \Sigma_c \uplus \Sigma_r \uplus \Sigma_i$:

- push **one** stack symbol on call symbols Σ_c
- pop **one** stack symbol on return symbols Σ_r
- don't touch the stack on internal symbols Σ_i

Visibly Pushdown Automata (VPAs) [AM04]

VPAs = Pushdown Automata on *structured* alphabet $\Sigma = \Sigma_c \uplus \Sigma_r \uplus \Sigma_i$:

- push **one** stack symbol on call symbols Σ_c
- pop **one** stack symbol on return symbols Σ_r
- don't touch the stack on internal symbols Σ_i

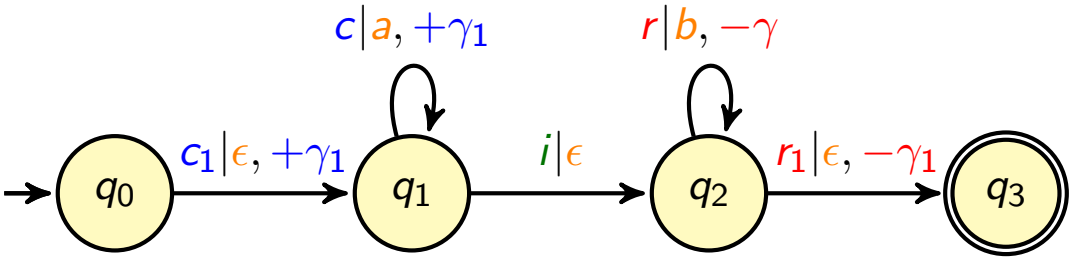


- closure properties
- decidability properties
- determinizable

$$L(A) = \{c_1 c^n \ i \ r^n \ r_1 \ \mid \ n \geq 0\}$$

Visibly Pushdown Transducers (VPTs) [RS08],[MFCS'10]

= associate output words in Δ^* with transitions of a VPA

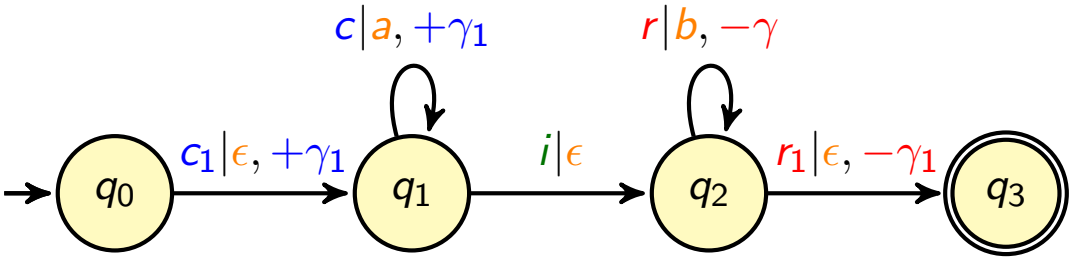


$$R(T) = \{(c_1 c^n \ i \ r^n \ r_1, a^n b^n) \ | \ n \geq 0\}$$

- nested word-to-word
- ranges are CFL
- functional VPT
- deterministic VPT

Visibly Pushdown Transducers (VPTs) [RS08],[MFCS'10]

= associate output words in Δ^* with transitions of a VPA



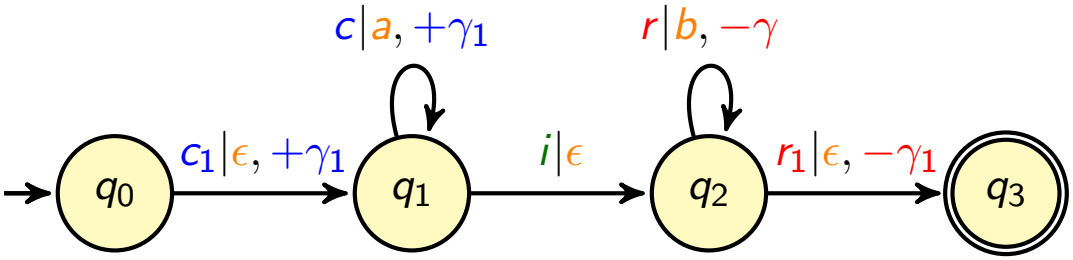
$$R(T) = \{(c_1 c^n \ i \ r^n \ r_1, a^n b^n) \ | \ n \geq 0\}$$

- nested word-to-word
- ranges are CFL
- functional VPT
- deterministic VPT

- Positive results: functionality, k-valuedness, equivalence
- Negative results: composition, type-checking
- Open problem: determinizability, finite valuedness

Visibly Pushdown Transducers (VPTs) [RS08],[MFCS'10]

= associate output words in Δ^* with transitions of a VPA



$$R(T) = \{(c_1 c^n \ i \ r^n \ r_1, a^n b^n) \ | \ n \geq 0\}$$

- nested word-to-word
- ranges are CFL
- functional VPT
- deterministic VPT

- Positive results: functionality, k-valuedness, equivalence
- Negative results: composition, type-checking
- Open problem: determinizability, finite valuedness

- Well-nested output words: composition and type-checking [DLT'14]

Streamability Problem for VPT [FSTTCS'11]

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT T.

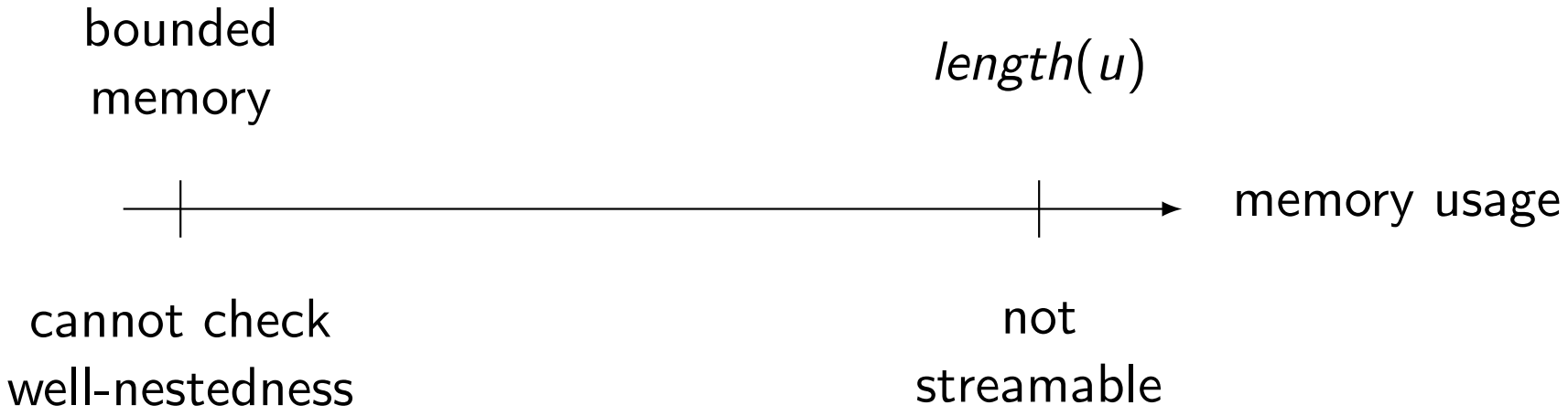
How much memory is needed to compute $T(u)$ from an input stream u?

Streamability Problem for VPT [FSTTCS'11]

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT T.

How much memory is needed to compute $T(u)$ from an input stream u?

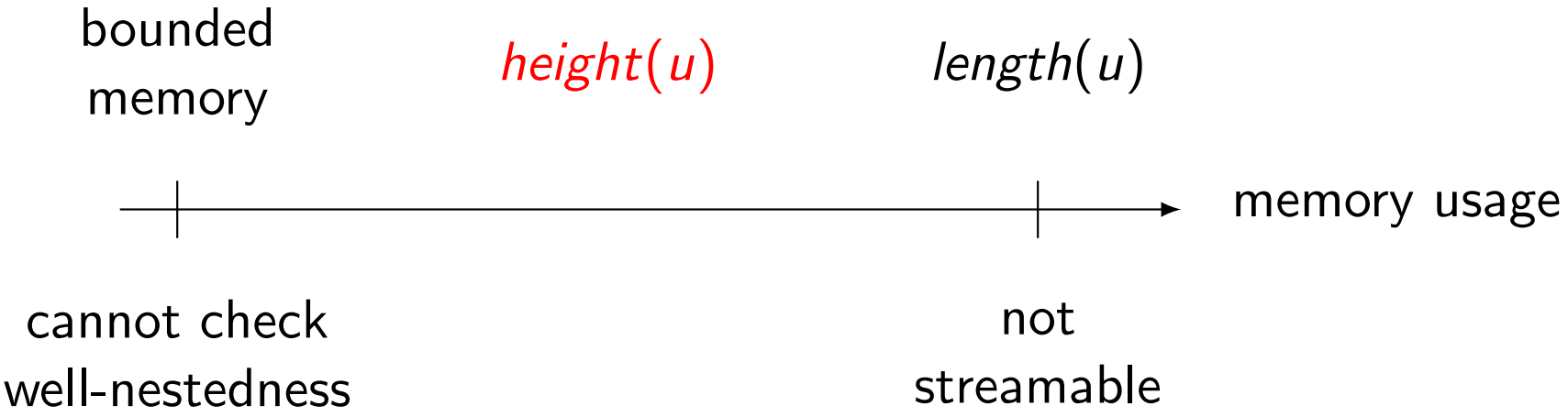


Streamability Problem for VPT [FSTTCS'11]

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT T.

How much memory is needed to compute $T(u)$ from an input stream u?



Streamability Problem for VPT [FSTTCS'11]

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT T.

How much memory is needed to compute $T(u)$ from an input stream u?

Height Bounded Memory Problem

Input: a transformation T defined by a fVPT

Output: can T be realized with height-bounded memory?

$$\exists f : \mathbb{N} \to \mathbb{N} \cdot \forall u \in dom(T)$$

$T(u)$ can be computed with $f(height(u))$-bounded memory?

Streamability Problem for VPT [FSTTCS'11]

Streaming evaluation: avoid the storage of the whole input

Fix a functional VPT T.

How much memory is needed to compute $T(u)$ from an input stream u?

Height Bounded Memory Problem

Input: a transformation T defined by a fVPT

Output: can T be realized with height-bounded memory?

$$\exists f : \mathbb{N} \to \mathbb{N} \cdot \forall u \in dom(T)$$

$T(u)$ can be computed with $f(height(u))$-bounded memory?

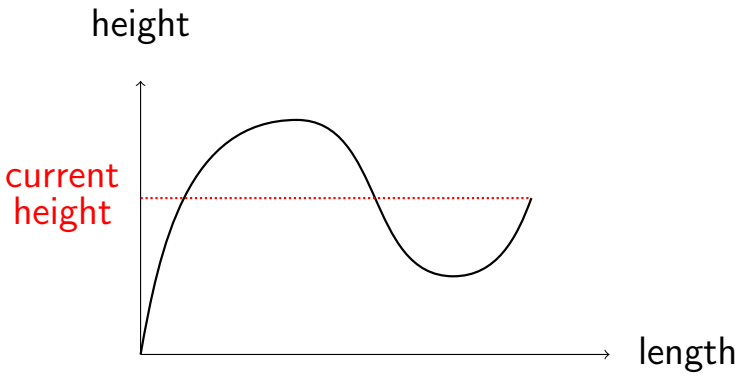
Using a twinning property, we prove:

Decidable in Co-NPTime

But: memory may depend exponentially in $height(u)$

Online Bounded Memory [FSTTCS'11]

Current height:

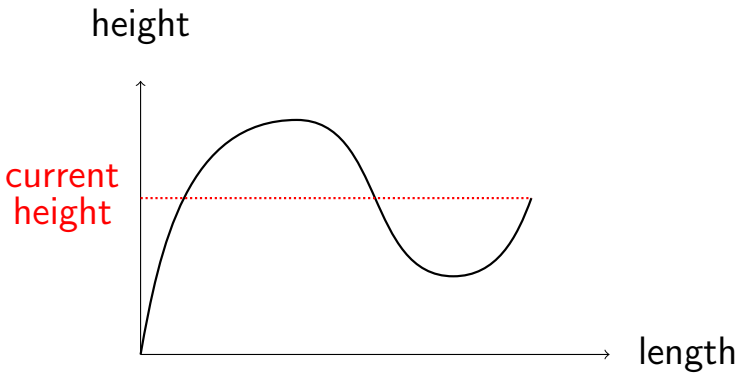


Online Bounded Memory Problem

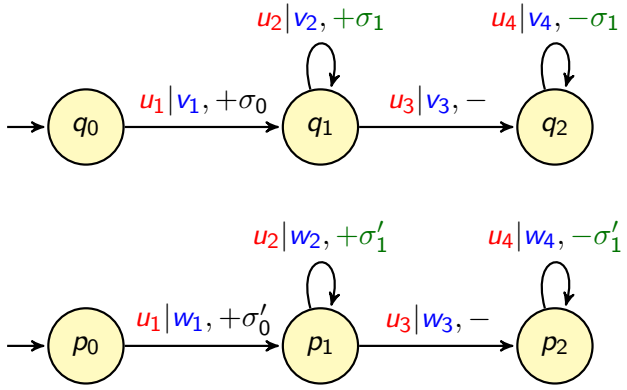
Input: a transformation T defined by a fVPT
Output: can T be realized with current-height-bounded memory?

Online Bounded Memory [FSTTCS'11]

Current height:



Matched Twinning Property



$\text{delay}(v_1, w_1) = \text{delay}(v_1 v_2 v_3 v_4, w_1 w_2 w_3 w_4)$

Online Bounded Memory Problem

Input: a transformation T defined by a fVPT

Output: can T be realized with current-height-bounded memory?

Characterization : Matched Twinning Property

➜ ensures quadratic dependence in the current height

➜ Decidable in Co-NPTime

Logical characterization for VPTs [LICS'16]

Theorem

MSO definable nested word-to-word transformations are expressively equivalent to 2DVPT$_{su}$.

- Two-way VPTs (stack operation exchanged when reading backwards)
- su : single-use

Key result: 2DVPT^{LA} $\equiv$2DVPT (LA:Look Around)

Logical characterization for VPTs [LICS'16]

Theorem

MSO definable nested word-to-word transformations are expressively equivalent to 2DVPT$_{su}$.

- Two-way VPTs (stack operation exchanged when reading backwards)
- su : single-use

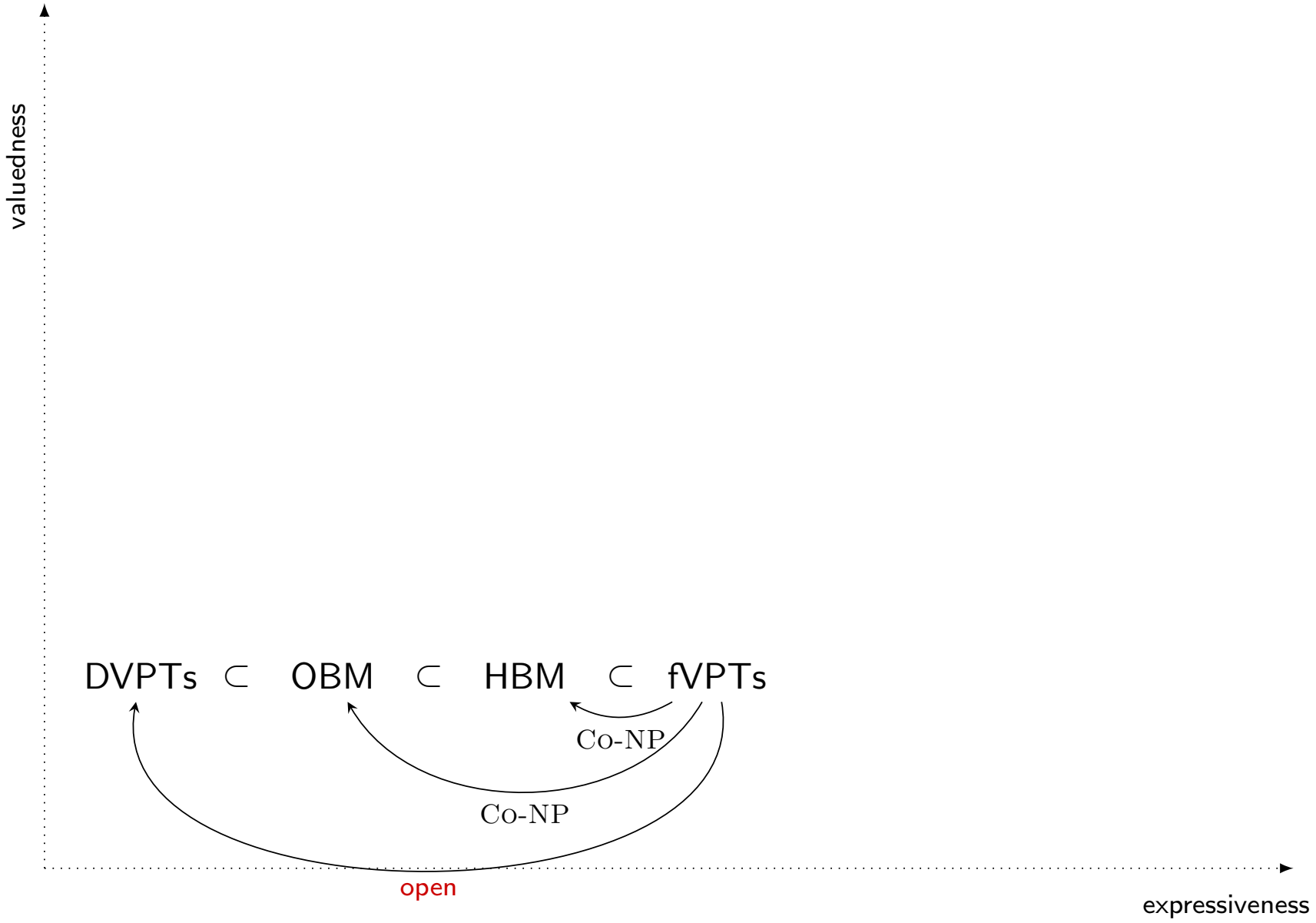
Key result: 2DVPT^{LA} $\equiv$2DVPT (LA:Look Around)

Bonus:

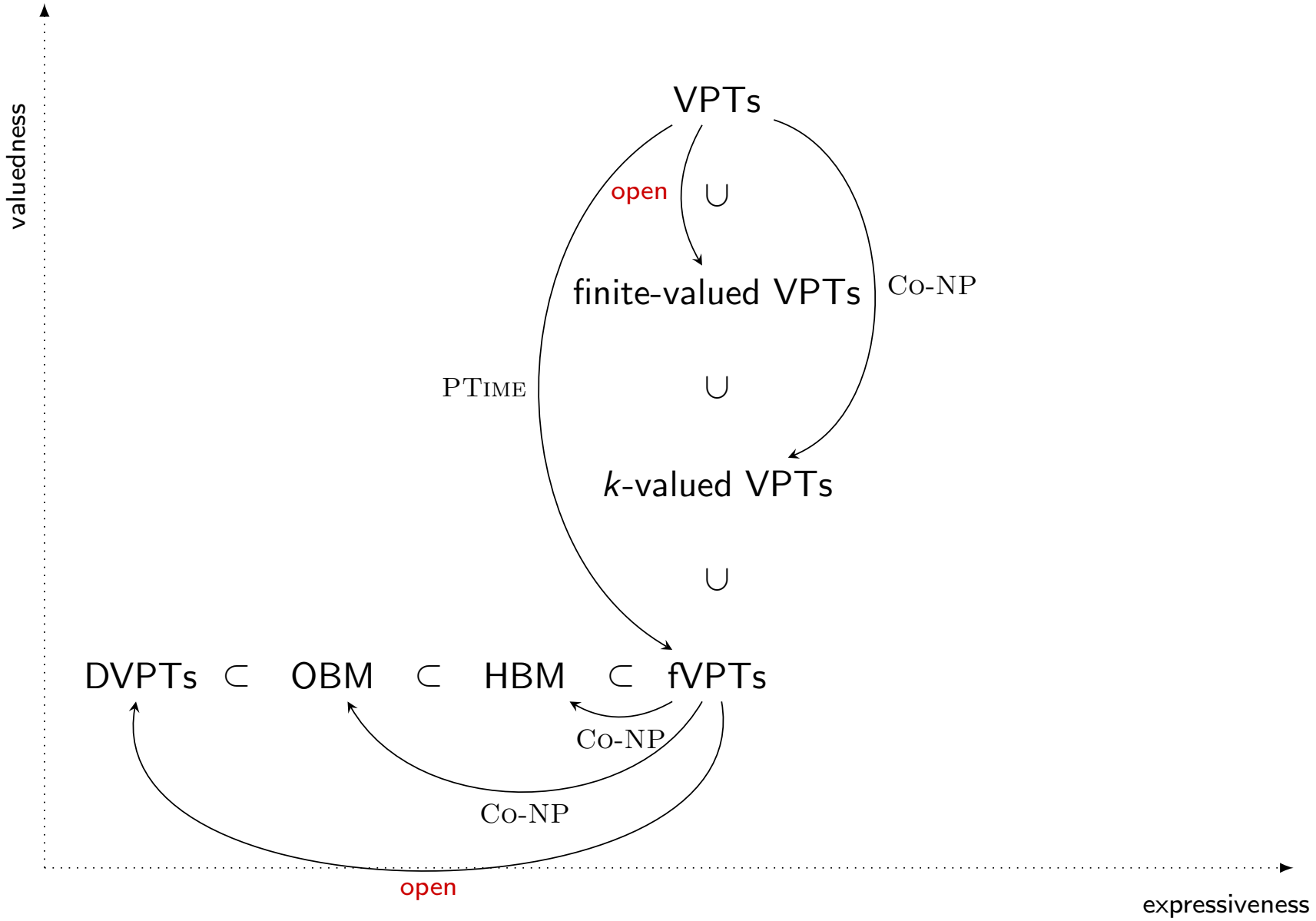
Theorem

Order-preserving MSO definable nested word-to-word transformations are expressively equivalent to functional VPTs.

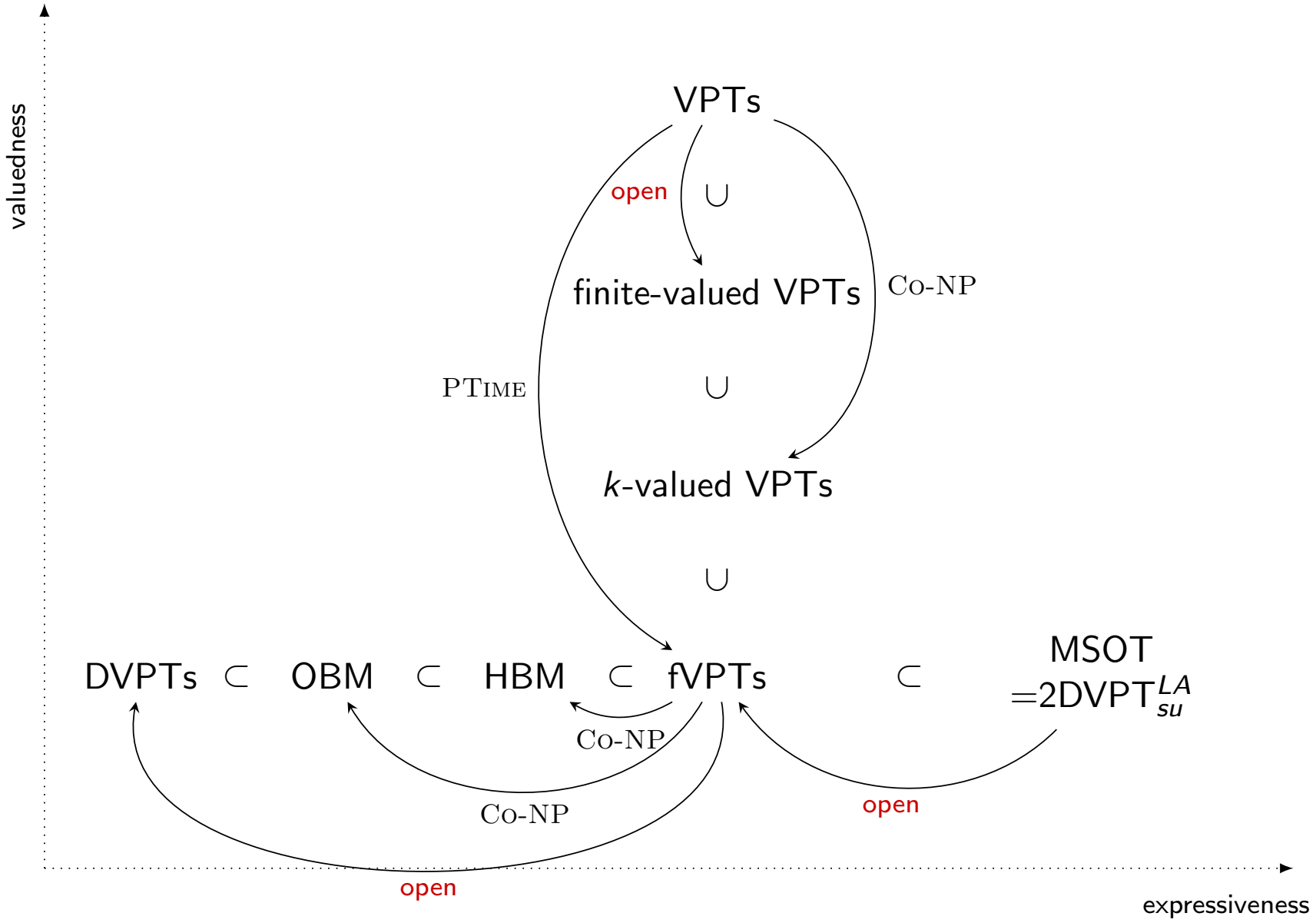
Summary for nested words transductions



Summary for nested words transductions



Summary for nested words transductions



Overview

Summary

- Decidability results based on patterns
  - Shift from NFT to raSST
  - Allows to derive efficient decision procedures
- Logic and Algebra connections
  - Minimization for sequential functions
  - Decidability of FOT for rational functions
  - Characterization only for regular functions
- Nested words
  - Most of the decidability results can be lifted
  - Logic can be lifted too
  - Algebra is missing

I did not present...

Some of these results are valid not only for transducers.

Recent results on model simplification:

- minimize the number of passes of a 2DFT
- minimize the number of registers of non-det. SST

Important results about relations:

- k-valuedness is decidable
- k-valued relations can be decomposed into functions

Recent trend about an alternative semantics (see next talk)

Perspectives

Shift from rational to regular functions

- register minimization
- algebraic presentation, canonical object

Specification languages for transformations

Alternative semantics to break undecidability/high complexity

Perspectives

Shift from rational to regular functions

- register minimization
- algebraic presentation, canonical object

Specification languages for transformations

Alternative semantics to break undecidability/high complexity

Thanks!