Word transducers: from 2-way to 1-way

Gabriele Puppis

joint works with Felix Baschenis
Olivier Gauwin
Anca Muscholl

Transform objects, here: words

transduction = mapping (or relation) from words to words

santiago $\longrightarrow$	sntg	erase vowels
santiago $\longrightarrow$	ogaitnas	reverse
santiago $\longrightarrow$	santiagosantiago	duplicate
santiago $\longrightarrow$	antiagos	rotate

1DFT = 1-way deterministic finite transducers

santiago ⟶ sntg erase vowels

1DFT = 1-way deterministic finite transducers

santiago → sntg erase vowels

2DFT = 2-way deterministic finite transducers

santiago → ogaitnas reverse

santiago → santiagosantiago duplicate

1DFT = 1-way deterministic finite transducers

1NFT = … non-deterministic …

santiago	→	sntg	erase vowels

2DFT = 2-way deterministic finite transducers

2NFT = … non-deterministic …

santiago	→	ogaitnas	reverse
santiago	→	santiagosantiago	duplicate

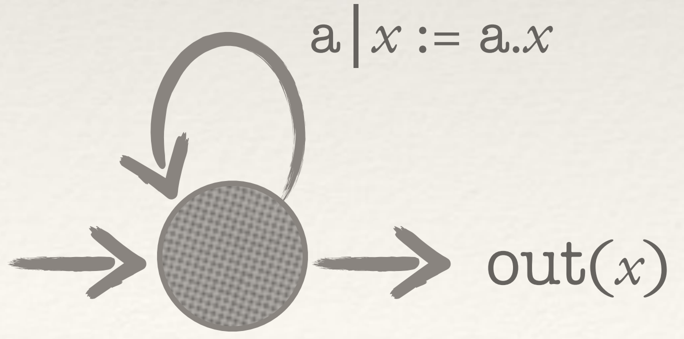
SST = streaming string transducers [Alur, Cerny '10]

- deterministic or non-deterministic
- 1-way
- write-only registers to store partial outputs

SST = streaming string transducers [Alur, Cerny '10]

- deterministic or non-deterministic
- 1-way
- write-only registers to store partial outputs

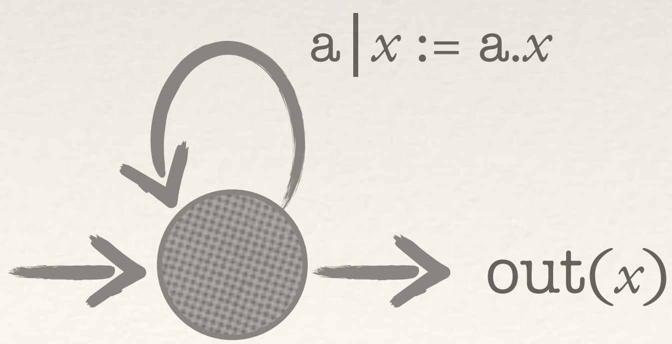
santiago ⟶ ogaitnas



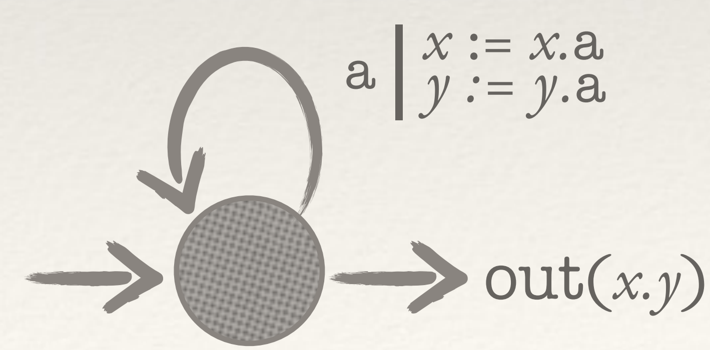
SST = streaming string transducers [Alur, Cerny '10]

- deterministic or non-deterministic
- 1-way
- write-only registers to store partial outputs

santiago ⟶ ogaitnas



santiago ⟶ santiagosantiago



MSOT = monadic second-order transductions [Courcelle '95]

logically define the output inside copies of the input:

- domain: unary formula selecting positions in each copy
- order: binary formula defining an order on the domain
- letters: unary formulas partitioning the domain

MSOT = monadic second-order transductions [Courcelle '95]

logically define the output inside copies of the input:

- domain: unary formula selecting positions in each copy
- order: binary formula defining an order on the domain
- letters: unary formulas partitioning the domain

santiago ⟶ santiagosantiago duplicate

$\varphi_<(x,y)$ = "x, y in the same copy and $x < y$
or x in the first copy and y in the second copy"

$$\text{1DFT} \quad \underset{w \mapsto ww}{\subsetneq} \quad \text{2DFT} \quad = \quad \text{DSST} \quad = \quad \text{MSOT}$$

Automata = logic ?

$$\begin{array}{ccccccc}
 & & w \mapsto ww & & & & \\
\text{1DFT} & \subsetneq & \text{2DFT} & = & \text{DSST} & = & \text{MSOT} \\
\rotatebox{-90}{$\subsetneq$} & & \rotatebox{-90}{$\subsetneq$} & & \rotatebox{-90}{$\subsetneq$} & & \rotatebox{-90}{$\subsetneq$} \\
\text{1NFT} & & \text{2NFT} & & \text{NSST} & & \text{NMSOT} \\
w \mapsto \Sigma^{|w|} & & & & & &
\end{array}$$

Automata = logic ?

$$
\begin{tikzcd}
1\mathrm{DFT} \arrow[r, phantom, "\subsetneq"] \arrow[d, phantom, sloped, "\subsetneq"] & \overset{w \mapsto ww}{2\mathrm{DFT}} \arrow[r, phantom, "="] \arrow[d, phantom, sloped, "\subsetneq"] & \mathrm{DSST} \arrow[r, phantom, "="] \arrow[d, phantom, sloped, "\subsetneq"] & \mathrm{MSOT} \arrow[d, phantom, sloped, "\subsetneq"] \\
\underset{w \mapsto \Sigma^{|w|}}{1\mathrm{NFT}} \arrow[r, phantom, "\subsetneq"] & \underset{w \mapsto w^*}{2\mathrm{NFT}} \arrow[r, phantom, "\neq"] & \underset{uv \mapsto vu}{\mathrm{NSST}} \arrow[r, phantom, "="] & \mathrm{NMSOT}
\end{tikzcd}
$$

Automata = logic ?

$$
\begin{array}{ccccccc}
 & & w \mapsto ww & & & & \\
\text{1DFT} & \subsetneq & \text{2DFT} & = & \text{DSST} & = & \text{MSOT} \\
\rotatebox[origin=c]{-90}{$\subsetneq$} & & \rotatebox[origin=c]{-90}{$\subsetneq$} & & \rotatebox[origin=c]{-90}{$\subsetneq$} \;\; [\,\| \;\; \text{if functional}] & & \rotatebox[origin=c]{-90}{$\subsetneq$} \\
\text{1NFT} & \subsetneq & \text{2NFT} & \neq & \text{NSST} & = & \text{NMSOT} \\
w \mapsto \Sigma^{|w|} & & w \mapsto w^* & & uv \mapsto vu & &
\end{array}
$$

$[\,= \;\; \text{if functional}]$

First part

2NFT vs 1NFT

- characterisation of 1-way definability
- undecidability in the non-functional case

Second part

Minimising resources

- sweeps of 2NFT vs registers of NSST
- characterisation of k-sweep definability

Problem:

given a 2NFT, is it 1-way definable (equivalent to some 1NFT) ?

Problem:

given a 2NFT, is it 1-way definable (equivalent to some 1NFT) ?

The above problem is decidable, with non-elementary complexity.

[Filiot, Gauwin, Reynier, Servais '13]

Our result:

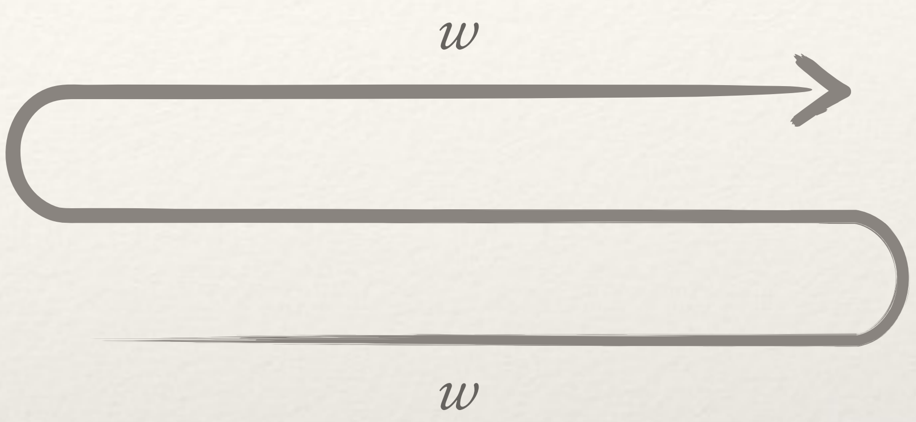
Given a functional 2NFT★ T,

- we can construct a 1NFT $T' \subsetneq T$ (2ExpTime)
- T is 1-way definable iff $T' = T$
- we can decide the latter (ExpSpace)

★ *sweeping* for simplicity

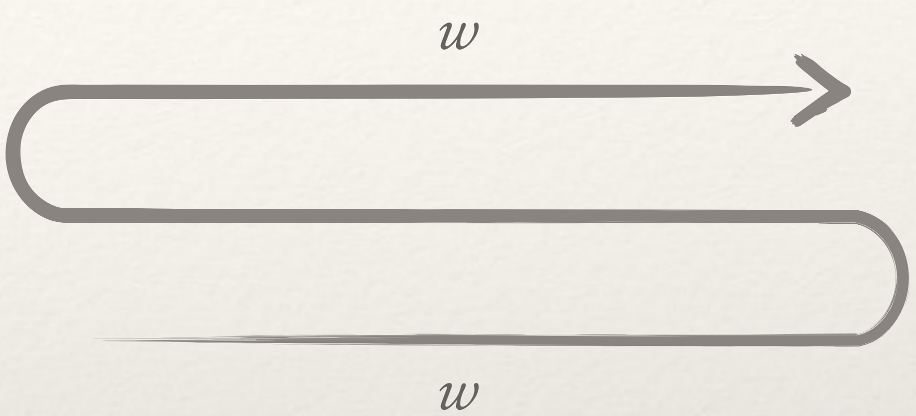
Fix a regular language R.

$$T(w) = \begin{cases} w.w & \text{if } w \in R \\ \perp & \text{otherwise} \end{cases}$$



Fix a regular language R.

$$T(w) = \begin{cases} w.w & \text{if } w \in R \\ \perp & \text{otherwise} \end{cases}$$

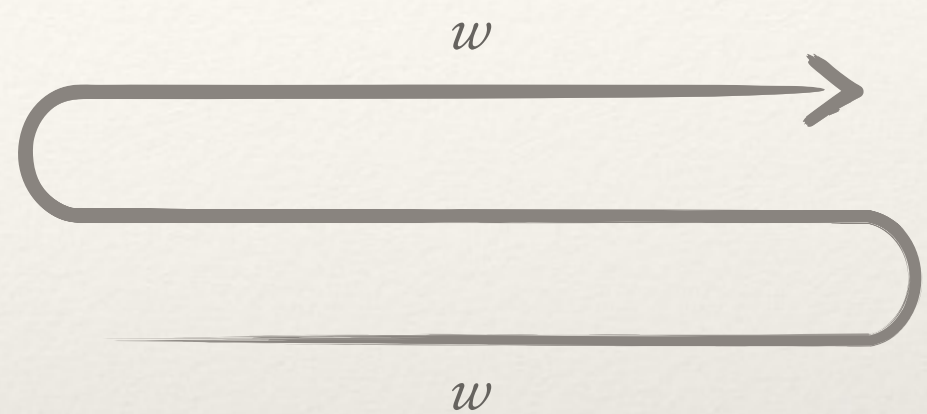


- $R = \Sigma^* \longrightarrow$ T is not 1-way definable

Example

Fix a regular language R.

$$T(w) = \begin{cases} w.w & \text{if } w \in R \\ \perp & \text{otherwise} \end{cases}$$



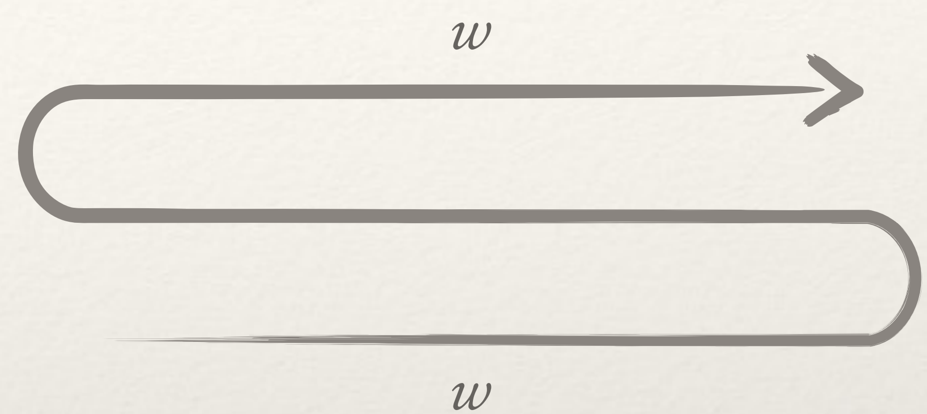
- $R = \Sigma^*$ $\longrightarrow$ T is not 1-way definable
- $R = [0]\Sigma[1]\Sigma\ldots\Sigma[2^n\text{-}1]\Sigma$ $\longrightarrow$ T has size n

  equivalent 1-way T' has size $\geq 2^{2^n}$

Example

Fix a regular language R.

$$T(w) = \begin{cases} w.w & \text{if } w \in R \\ \perp & \text{otherwise} \end{cases}$$

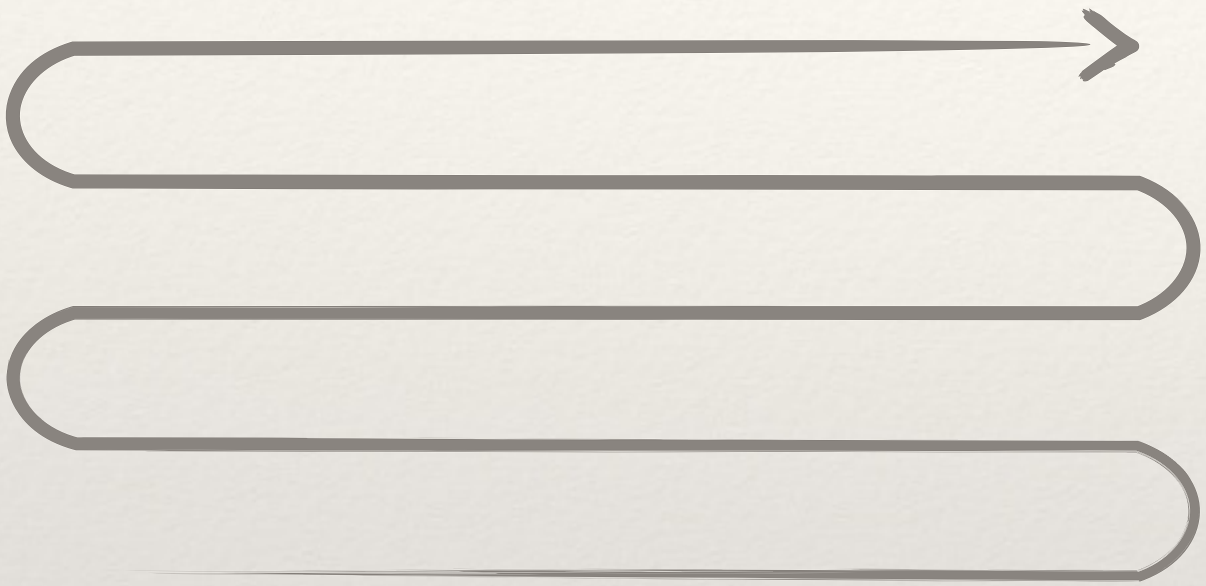


- $R = \Sigma^*$ $\longrightarrow$ T is not 1-way definable
- $R = [0]\Sigma[1]\Sigma\ldots\Sigma[2^n\text{-}1]\Sigma$ $\longrightarrow$ T has size n

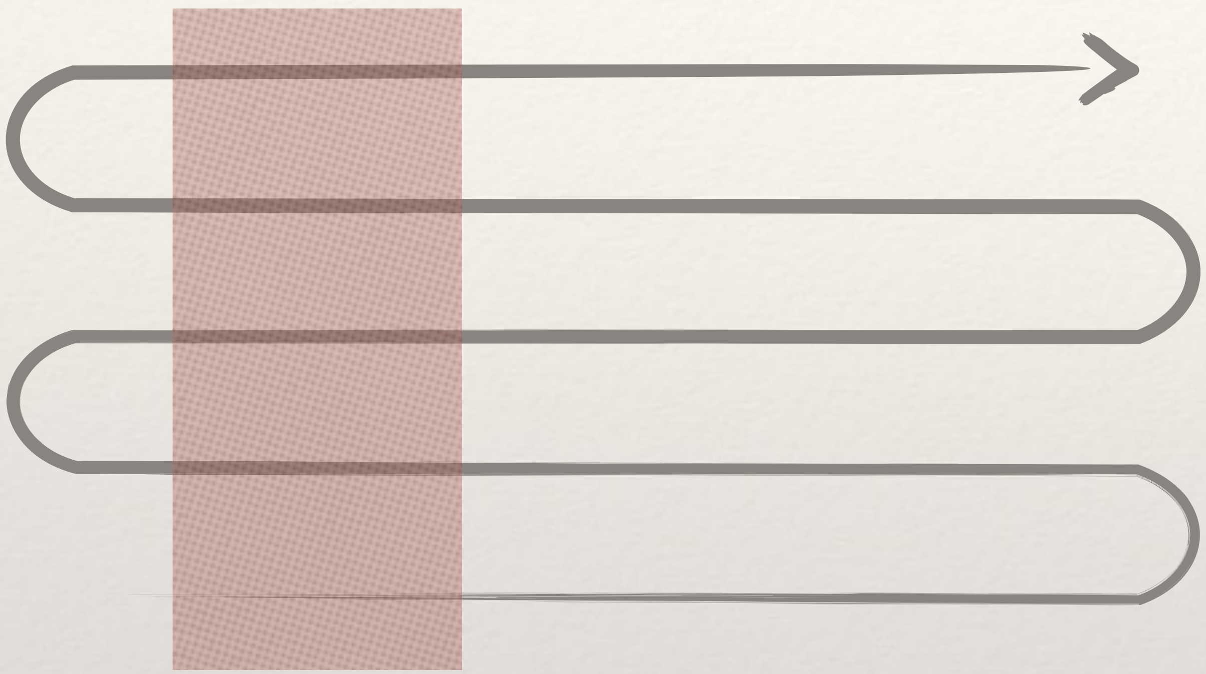
  equivalent 1-way T' has size $\geq 2^{2^n}$
- $R = \{\, \mathtt{abc} \,\}^*$ $\longrightarrow$ T is 1-way definable

  (output "abc" twice every 3 input letters)

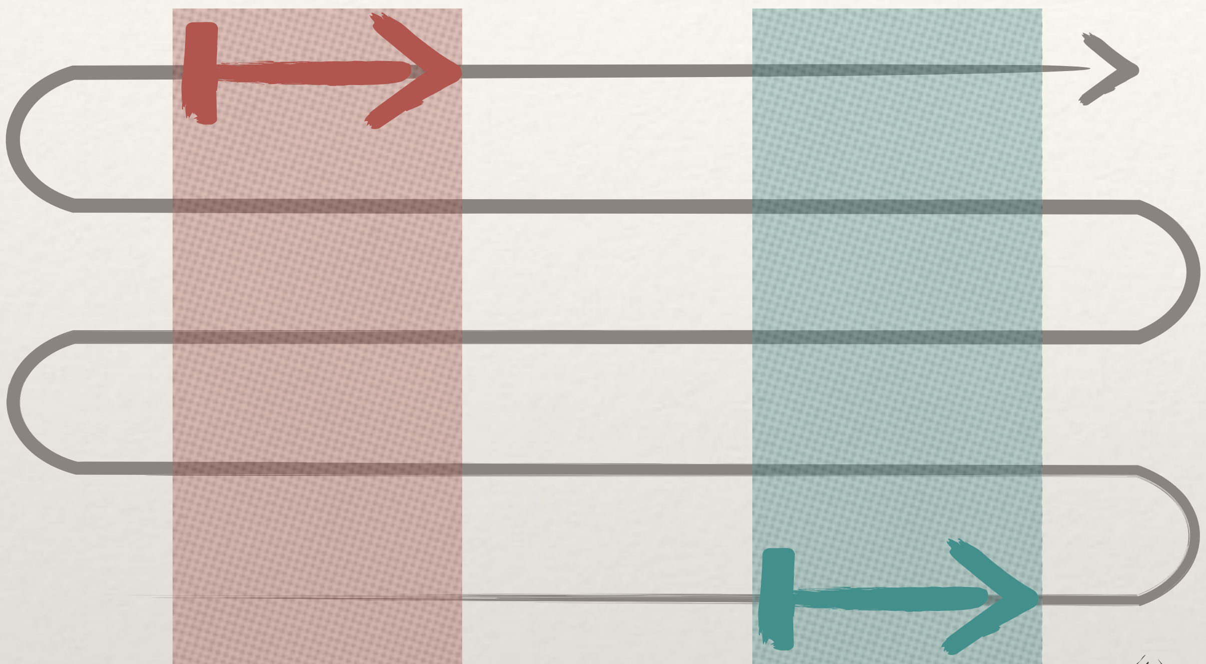
2NFT



2NFT



2NFT

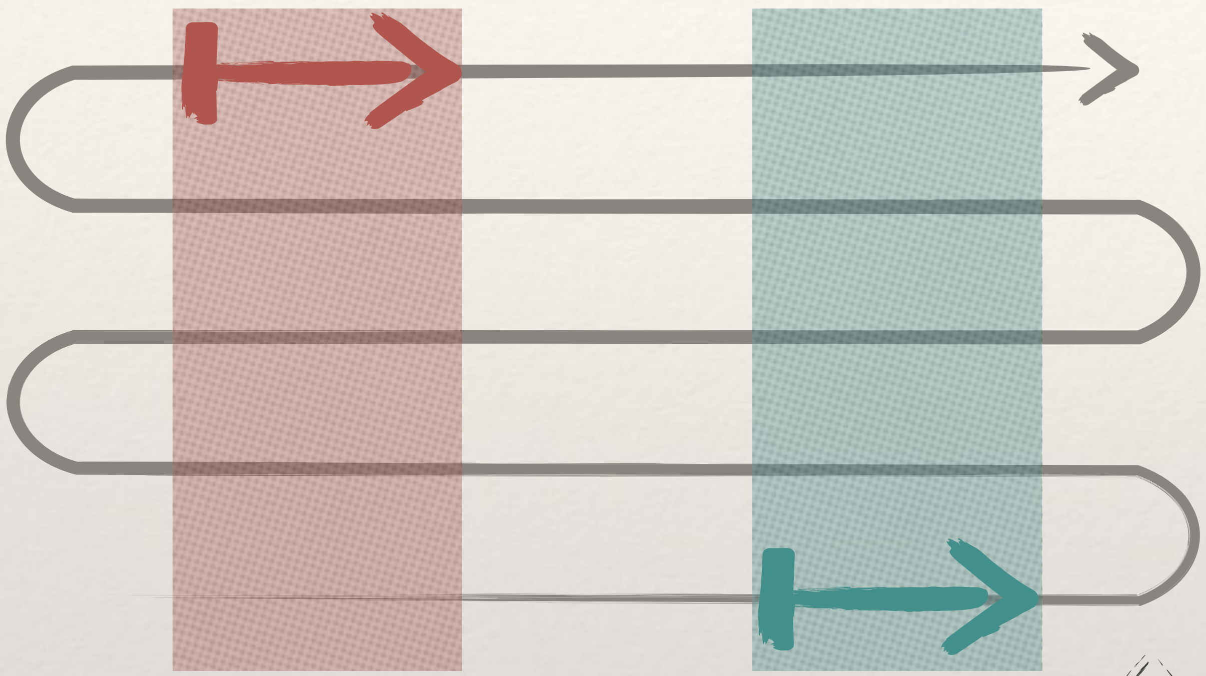


Inversion

- two loops
- non-empty outputs produced by the intercepted factors
- output order ≠ input order

2NFT

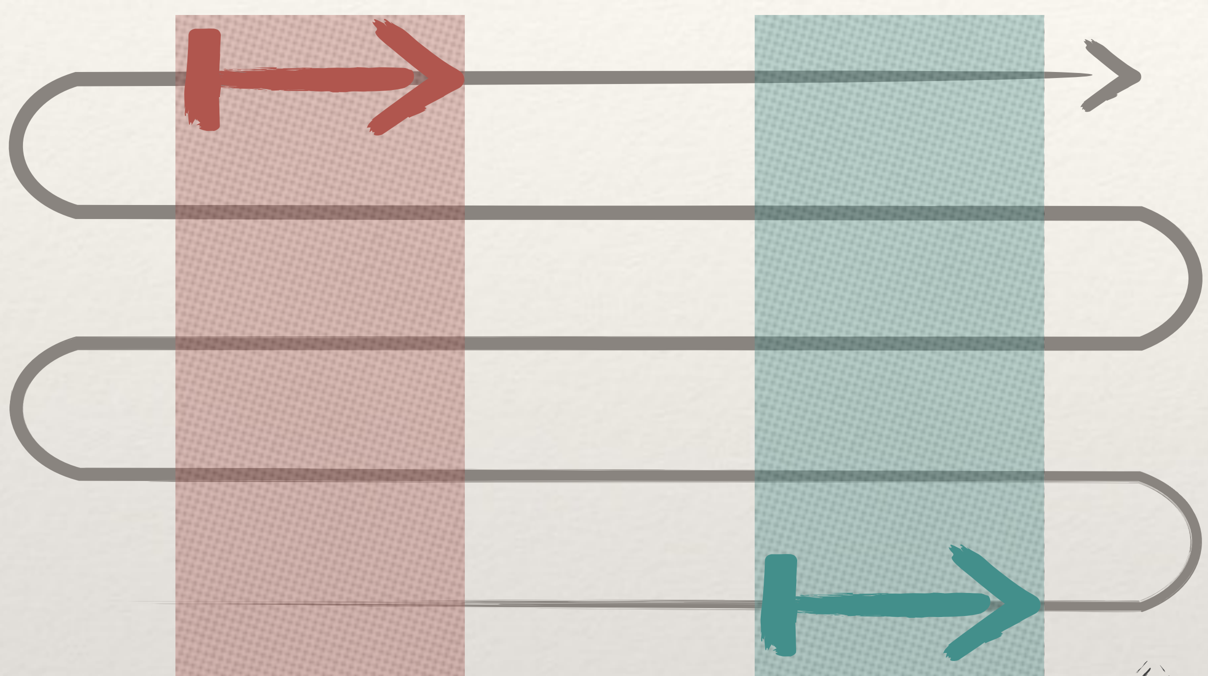
equivalent 1NFT



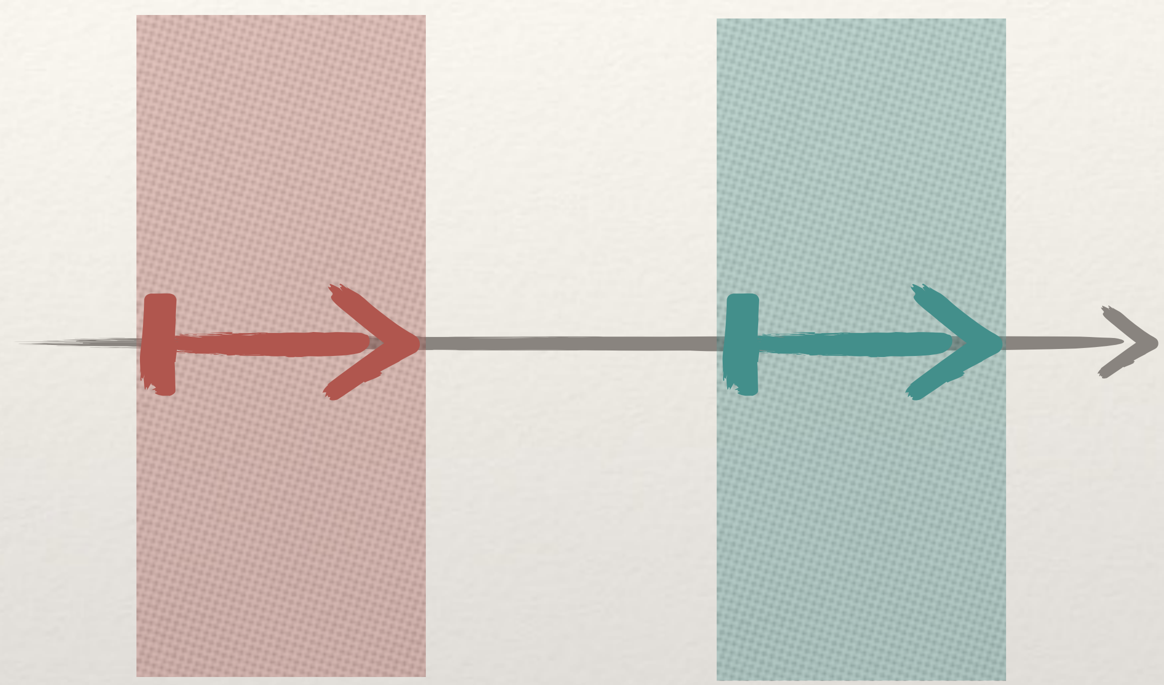
Inversion

- two loops
- non-empty outputs produced by the intercepted factors
- output order ≠ input order

2NFT



equivalent 1NFT

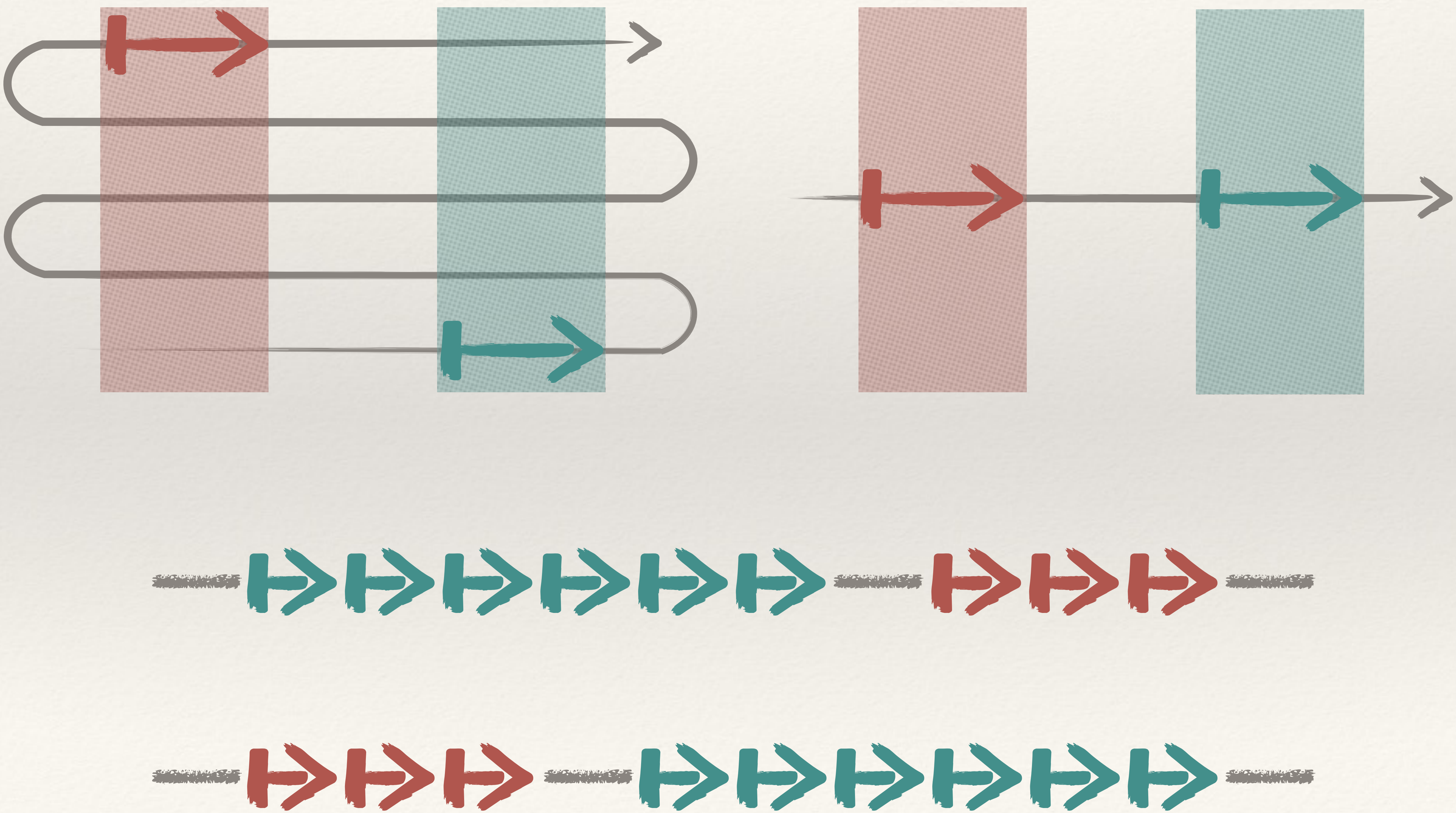


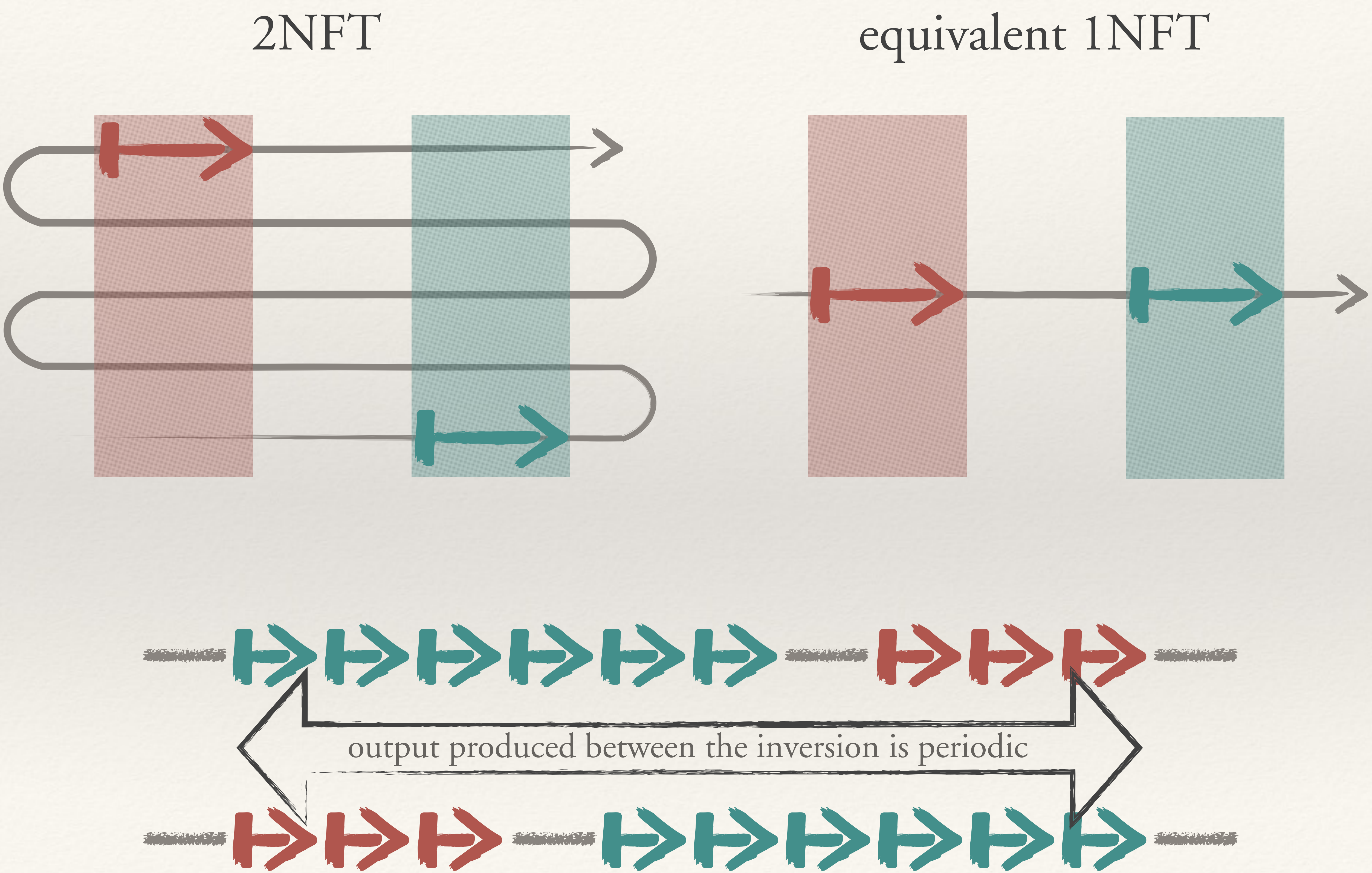
Inversion

- two loops
- non-empty outputs produced by the intercepted factors
- output order ≠ input order

2NFT

equivalent 1NFT

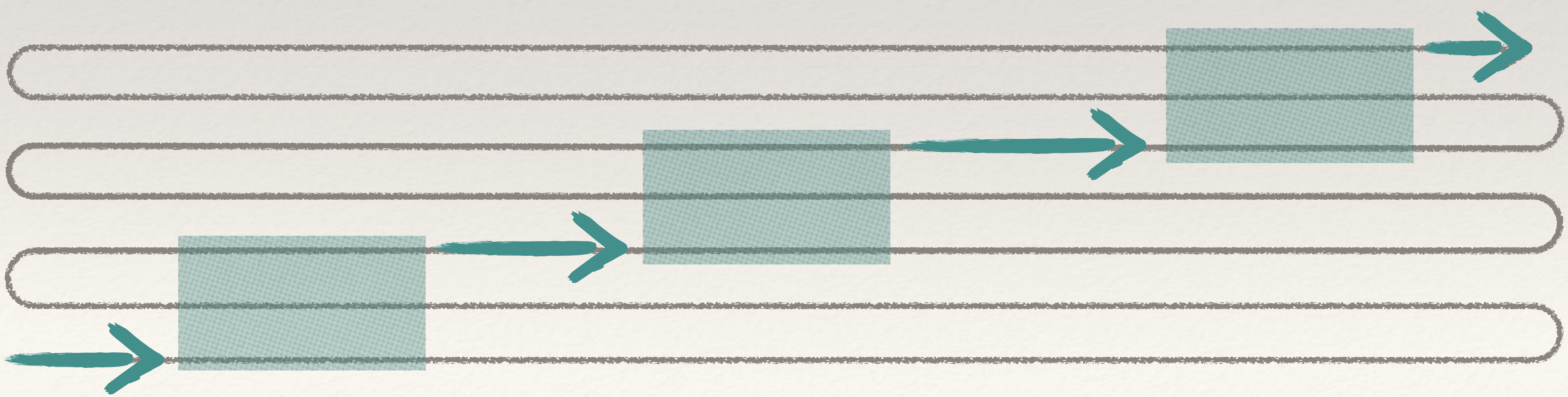



2NFT
equivalent 1NFT
output produced between the inversion is periodic

- T is 1-way definable
- every inversion produces an output of bounded period

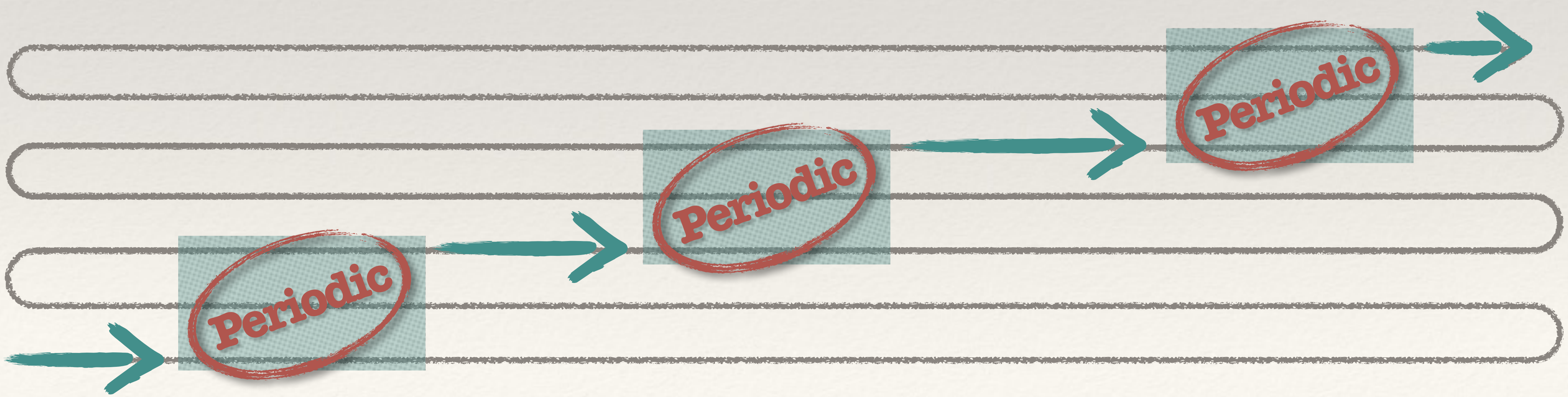
T.F.A.E.:

- T is 1-way definable
- every inversion produces an output of bounded period
- every run admits a stair-like decomposition



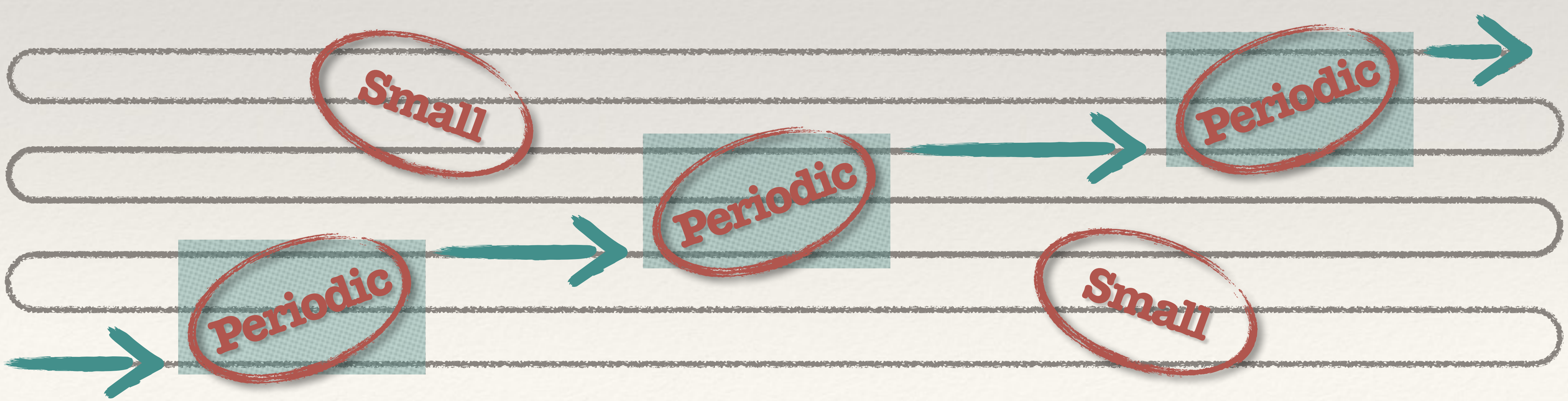
T.F.A.E.:

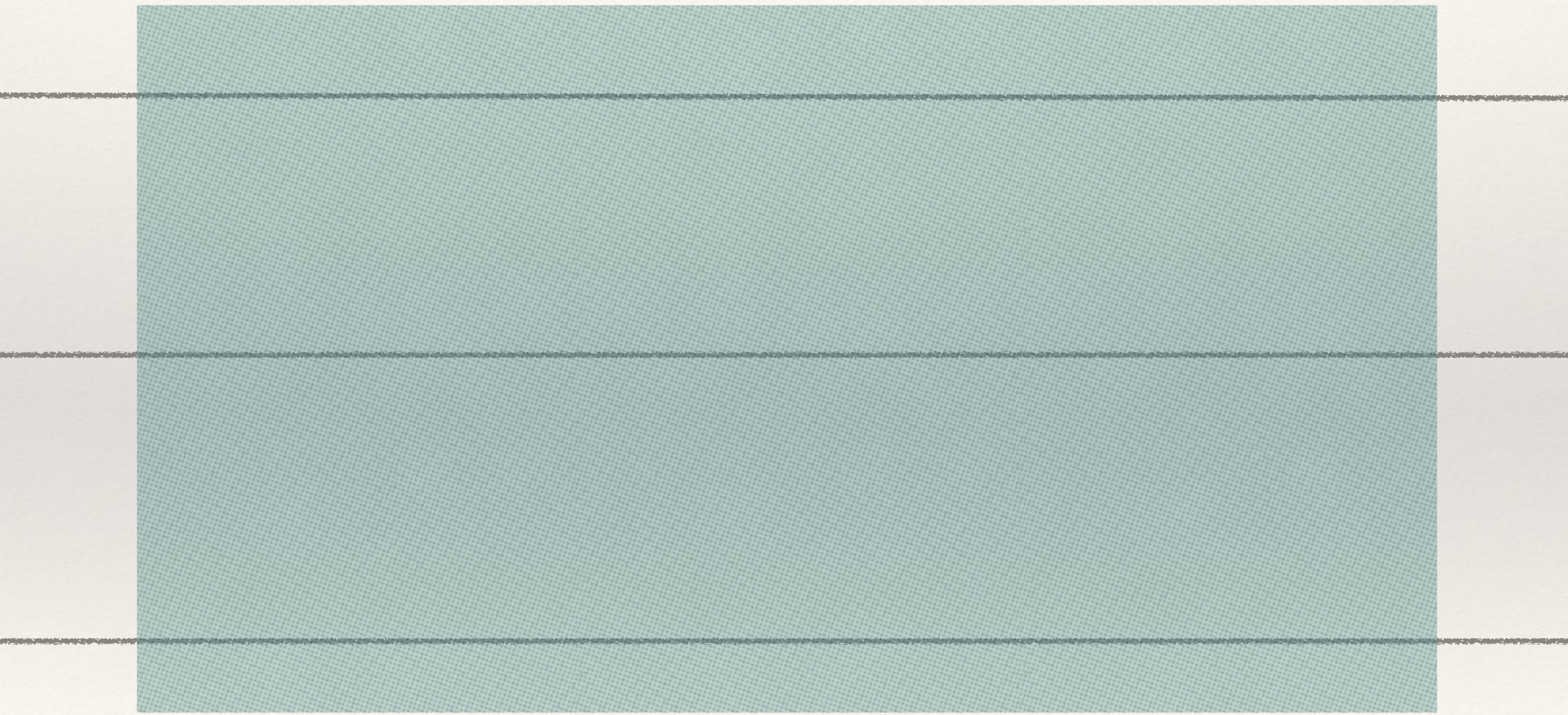
- T is 1-way definable
- every inversion produces an output of bounded period
- every run admits a stair-like decomposition



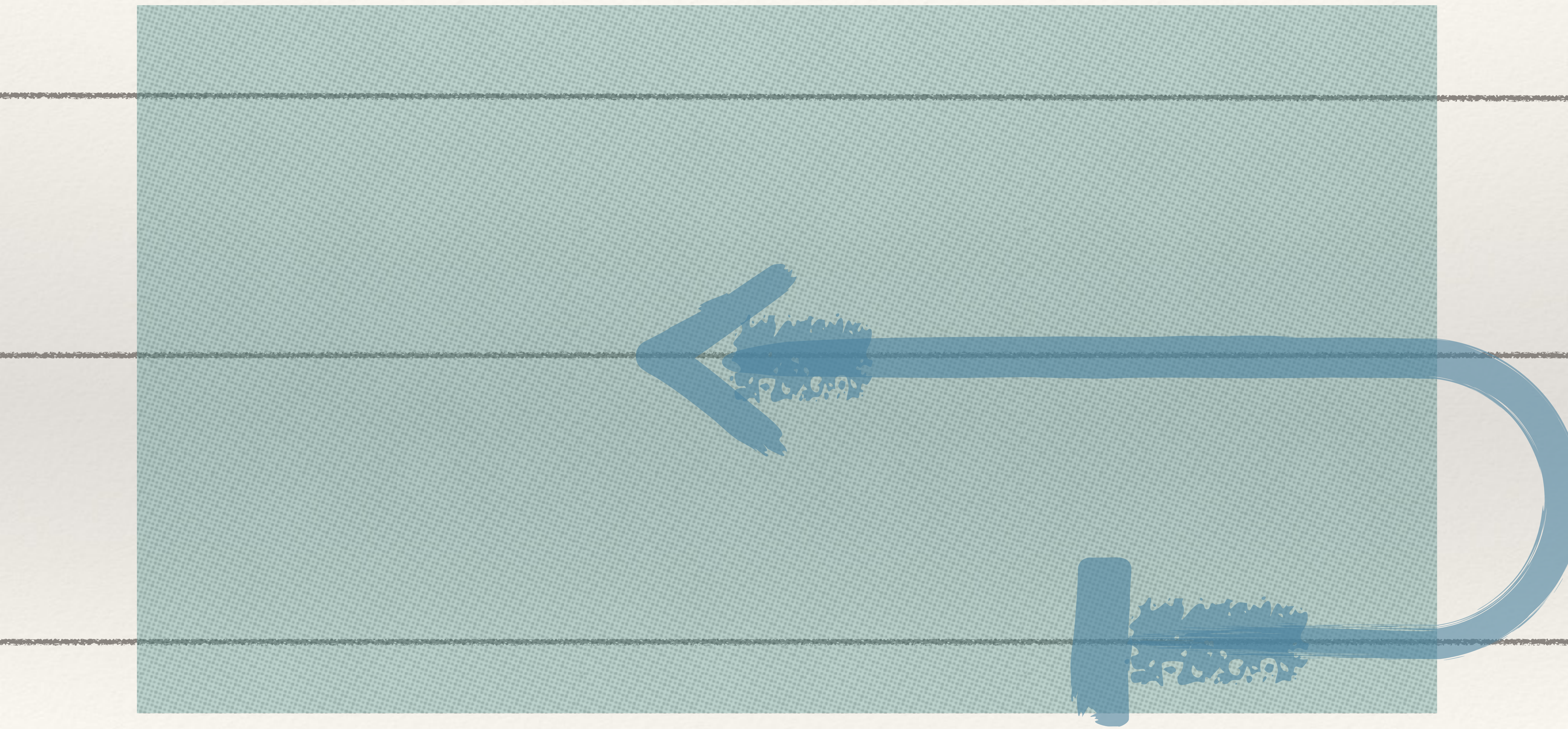
T.F.A.E.:

- T is 1-way definable
- every inversion produces an output of bounded period
- every run admits a stair-like decomposition

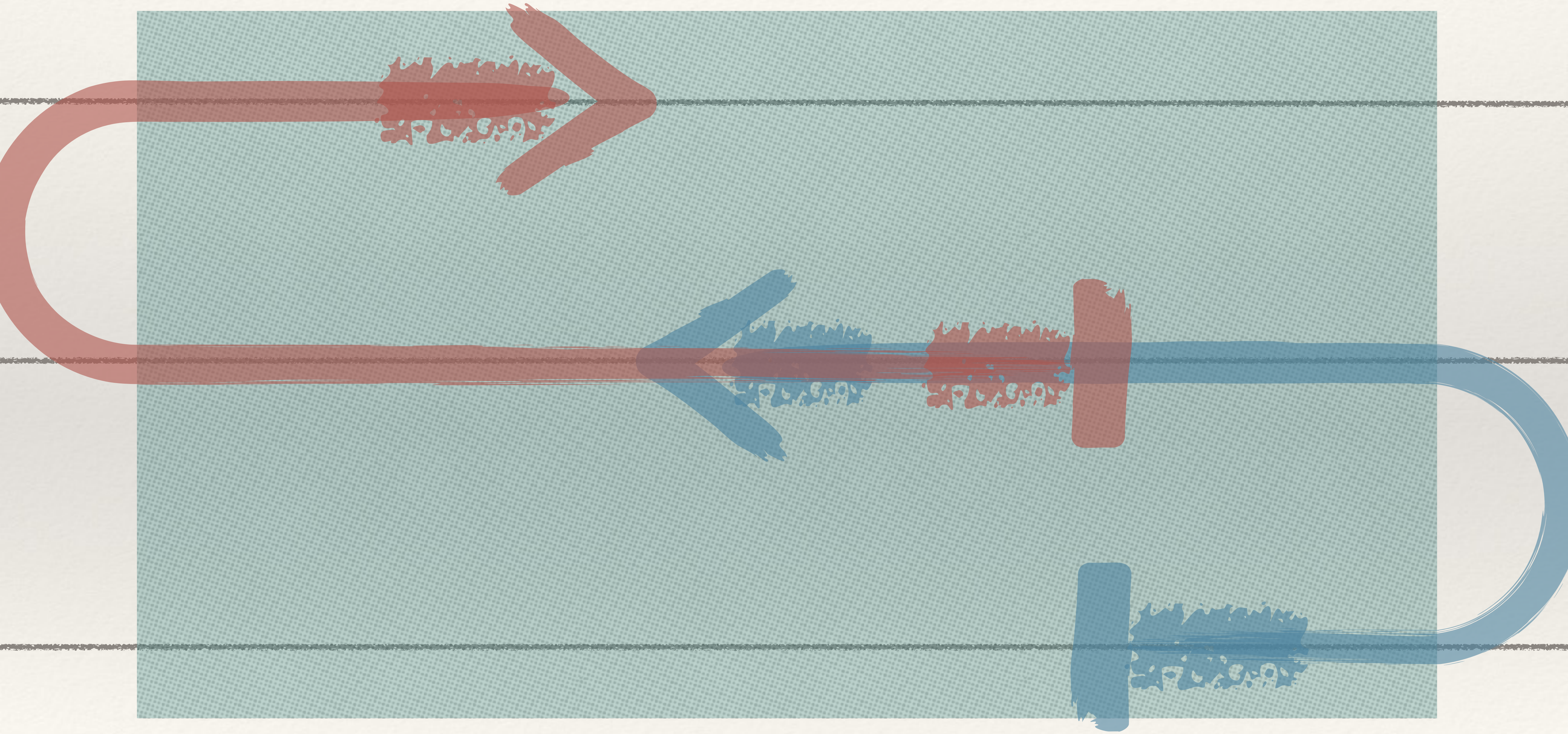




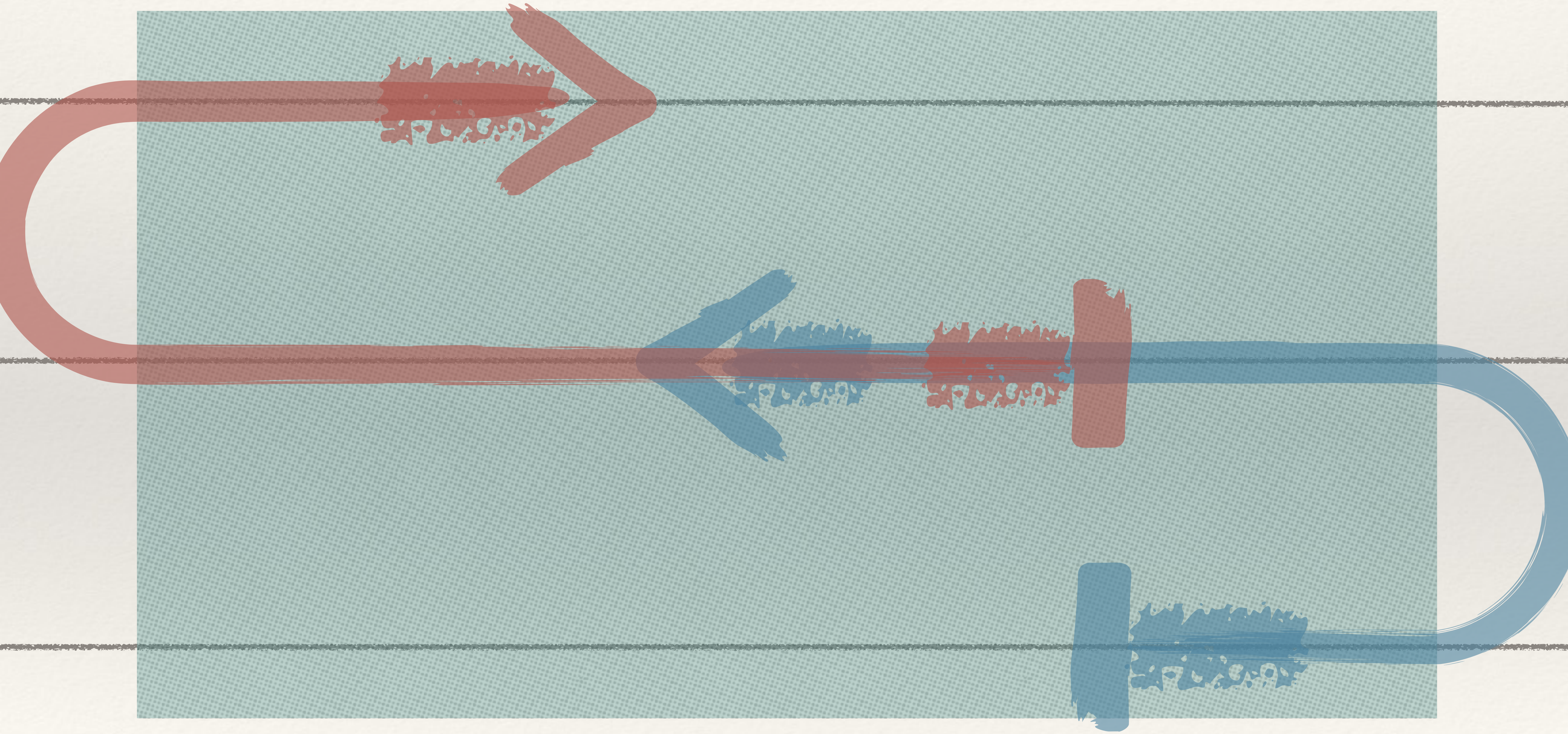
The characterisation



The characterisation

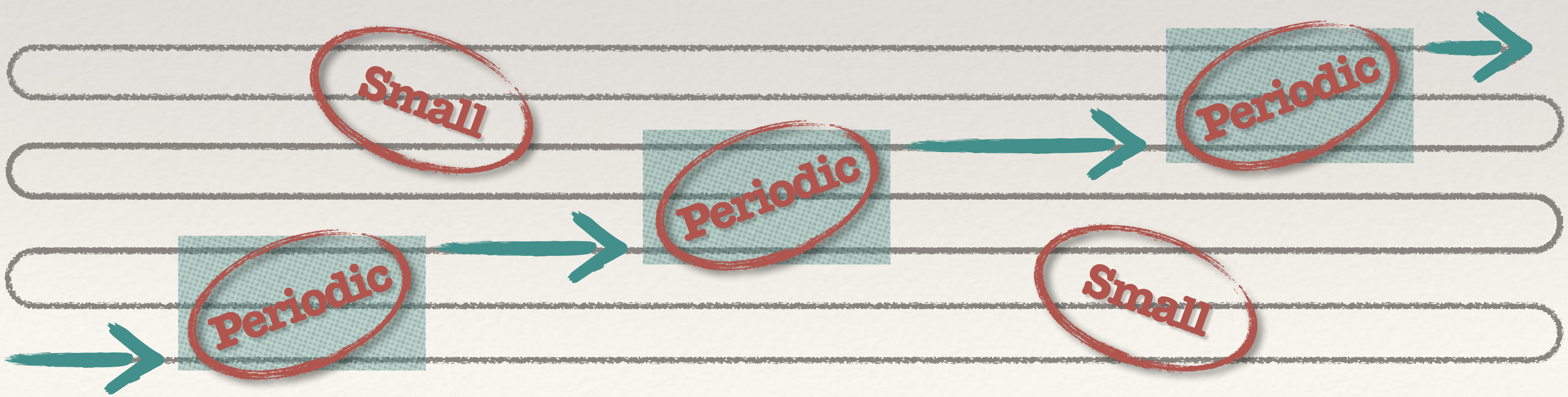


Outputs entirely covered by inversions are periodic…



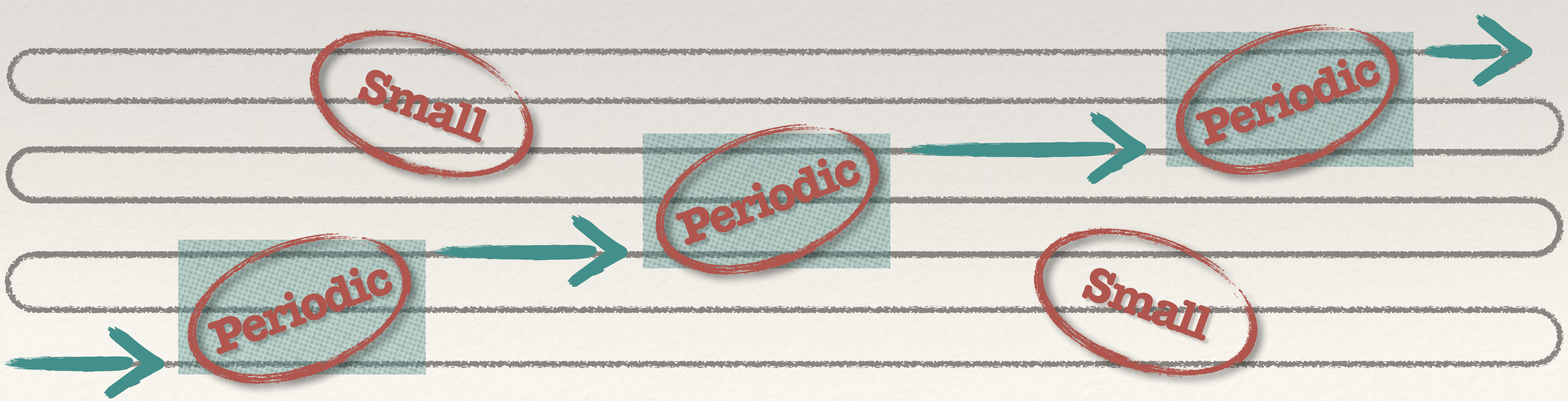
T.F.A.E.:

- T is 1-way definable
- every inversion produces an output of bounded period
- every run admits a stair-like decomposition



T.F.A.E.:

- T is 1-way definable
- every inversion produces an output of bounded period
- every run admits a stair-like decomposition **can be guessed in ExpSpace**



$\boxed{\text{Whether a non-functional 2NFT is 1-way definable is undecidable.}}$

Reduction from PCP — given morphisms $f, g : \Sigma^* \to \Delta^*$
does $\exists\, w \in \Sigma^+ \quad f(w) = g(w)$?

$\boxed{\text{Whether a non-functional 2NFT is 1-way definable is undecidable.}}$

Reduction from PCP — given morphisms $f, g : \Sigma^* \to \Delta^*$
does $\exists\, w \in \Sigma^+ \quad f(w) = g(w)$?

- Encodings : $w.u \mapsto w.\$^m$

$\boxed{\text{Whether a non-functional 2NFT is 1-way definable is undecidable.}}$

Reduction from PCP — given morphisms $f, g : \Sigma^* \to \Delta^*$
does $\exists\, w \in \Sigma^+ \quad f(w) = g(w)$?

- Encodings : $w.u \mapsto w.\$^m$ good if $m = |u|$
  and $u = f(w)$
  and $u = g(w)$

$\boxed{\text{Whether a non-functional 2NFT is 1-way definable is undecidable.}}$

Reduction from PCP — given morphisms $f, g : \Sigma^* \to \Delta^*$
does $\exists\, w \in \Sigma^+ \quad f(w) = g(w)$?

- Encodings : $w.u \mapsto w.\$^m$ bad if $m \neq |u|$
  or $u \neq f(w)$
  or $u \neq g(w)$

$\boxed{\text{Whether a non-functional 2NFT is 1-way definable is undecidable.}}$

Reduction from PCP — given morphisms $f, g : \Sigma^* \to \Delta^*$
does $\exists\, w \in \Sigma^+ \quad f(w) = g(w)$?

- Encodings : $w.u \mapsto w.\$^m$ bad if $m \neq |u|$
  or $u \neq f(w)$
  or $u \neq g(w)$

All encodings are bad

iff

T is 1-way definable

Whether a non-functional 2NFT is 1-way definable is undecidable.

Reduction from PCP — given morphisms $f, g : \Sigma^* \to \Delta^*$
does $\exists\, w \in \Sigma^+ \quad f(w) = g(w)$?

- Encodings : $\quad w.u \mapsto w.\$^m \qquad$ bad if $m \neq |u|$
  or $u \neq f(w)$
  or $u \neq g(w)$

All encodings are bad

iff

T is 1-way definable

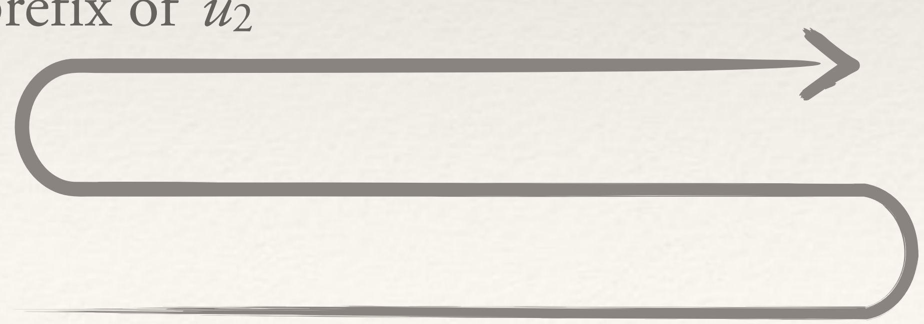
read $w.u$ output w

$\boxed{\text{Whether a non-functional 2NFT is 1-way definable is undecidable.}}$

Reduction from PCP — given morphisms $f, g : \Sigma^* \to \Delta^*$
does $\exists\, w \in \Sigma^+ \quad f(w) = g(w)$?

- Encodings : $\quad w.u \mapsto w.\$^m \qquad$ bad if $m \neq |u|$
  or $u \neq f(w)$
  or $u \neq g(w)$

guess $w = w_1.\mathtt{a}.w_2\,,\ u = u_1.u_2$
check $f(\mathtt{a})$ not a prefix of u_2
output $\$^{|f(w_1)|}\,\$^{|u_2|}$



read $w.u$ output w

$\boxed{\begin{array}{c}\text{All encodings are bad}\\ \text{iff}\\ \text{T is 1-way definable}\end{array}}$

What do we mean by resource ?

- number of control states
- amount of non-determinism
- number of sweeps
- number of registers
- …

What do we mean by resource ?

- number of control states
- amount of non-determinism
- number of sweeps
- number of registers
- …

interesting… but poorly understood

What do we mean by resource ?

- number of control states
- amount of non-determinism
- number of sweeps
- number of registers
- …

interesting… but poorly understood

next focus!

Given a deterministic SST over a *unary* output alphabet, one can compute the minimum number of registers in EXPTIME.

[Alur, Raghothaman '13]

Given a deterministic SST over a *unary* output alphabet,
one can compute the minimum number of registers in EXPTIME.

[Alur, Raghothaman '13]

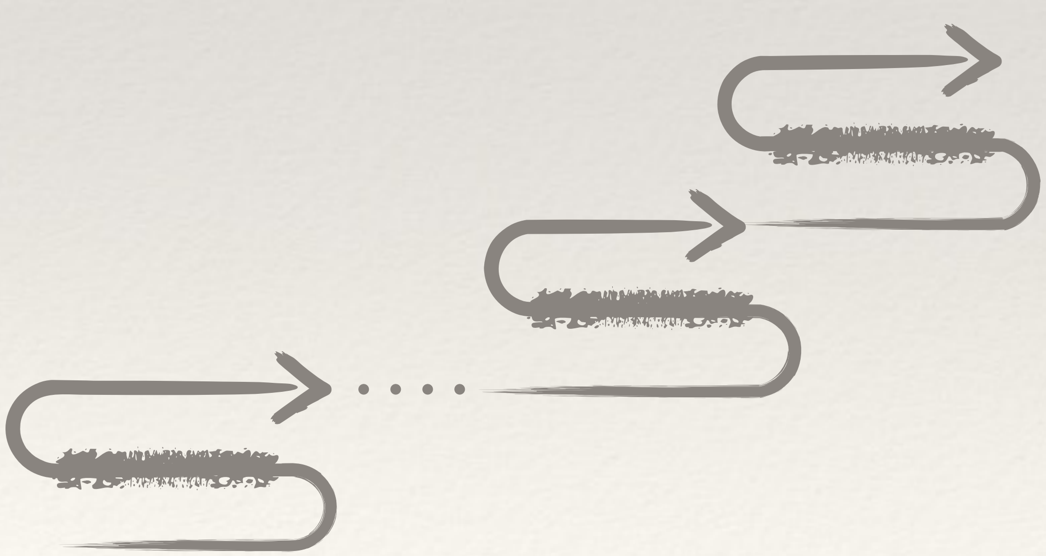
Our setting:

- arbitrary alphabet
- weak restriction on updates…
- non-deterministic (but still functional) SST

Recall 2NFT ≈ SST in the functional case

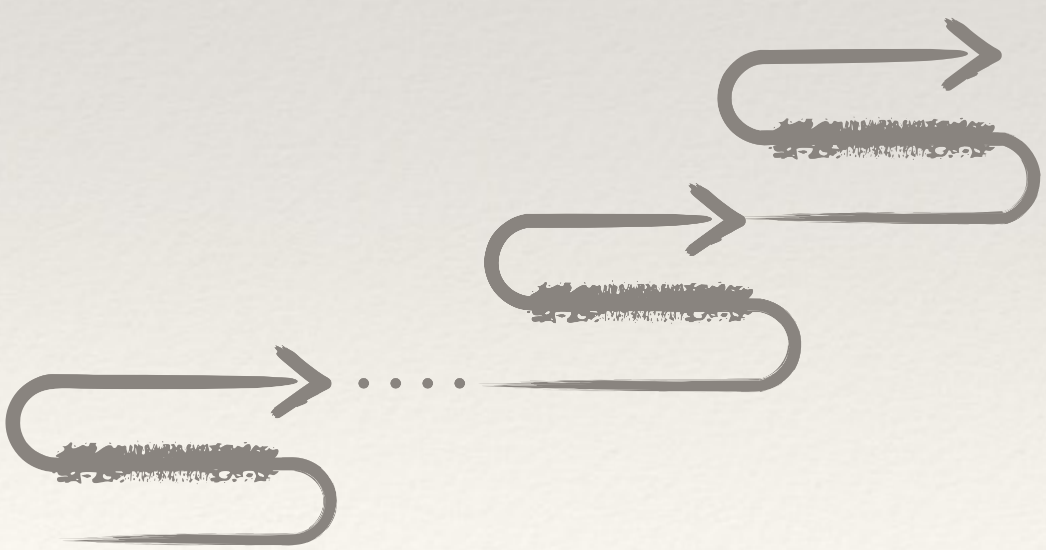
Recall 2NFT ≈ SST in the functional case

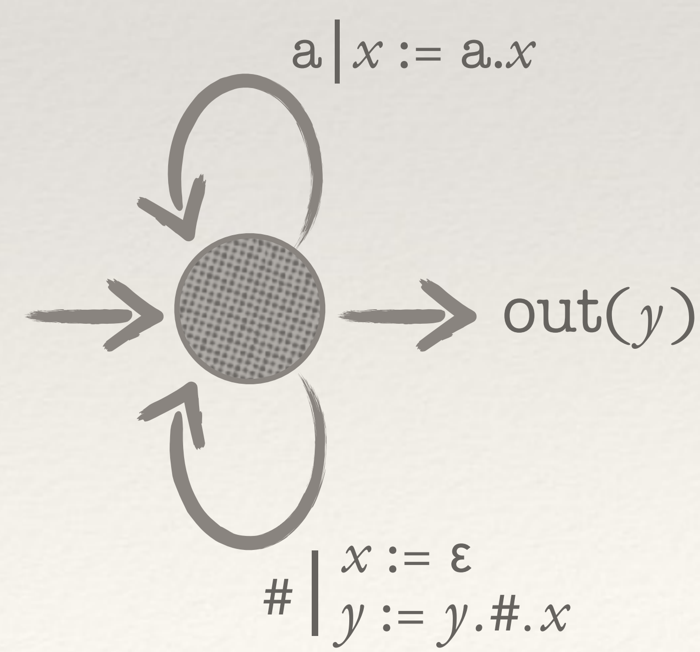
$$w_1 \# w_2 \# \ldots \# w_n \mapsto \mathrm{rev}(w_1) \# \mathrm{rev}(w_2) \# \ldots \# \mathrm{rev}(w_n)$$



Recall 2NFT ≈ SST in the functional case

$$w_1 \# w_2 \# \ldots \# w_n \mapsto \mathrm{rev}(w_1) \# \mathrm{rev}(w_2) \# \ldots \# \mathrm{rev}(w_n)$$





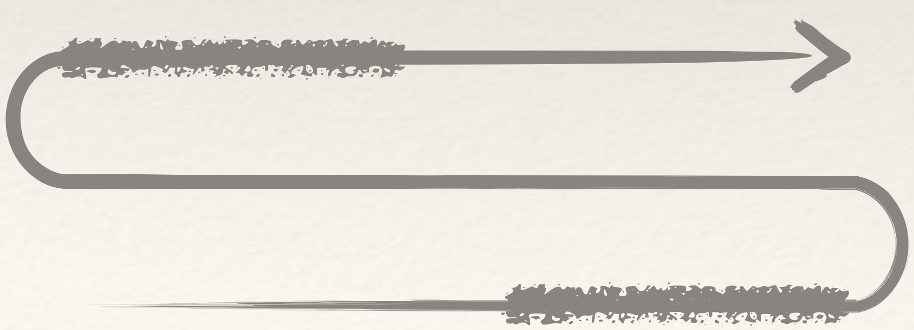
The following are also equally expressive:

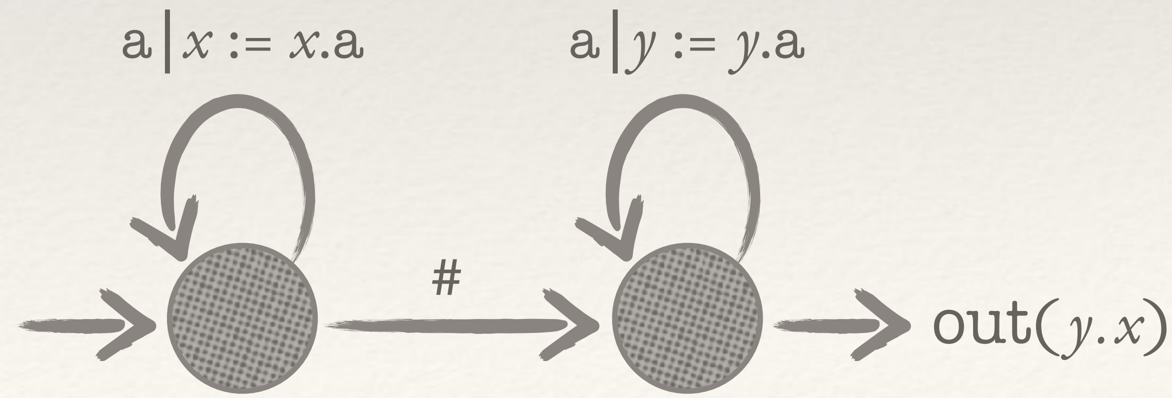
- concatenation-free SST $x := a.y.b$ ✓ $\cancel{x := y.z}$
- sweeping 2NFT
- bounded reversal 2NFT

The following are also equally expressive:

- concatenation-free SST $x := a.y.b$ $x := y.z$
- sweeping 2NFT
- bounded reversal 2NFT

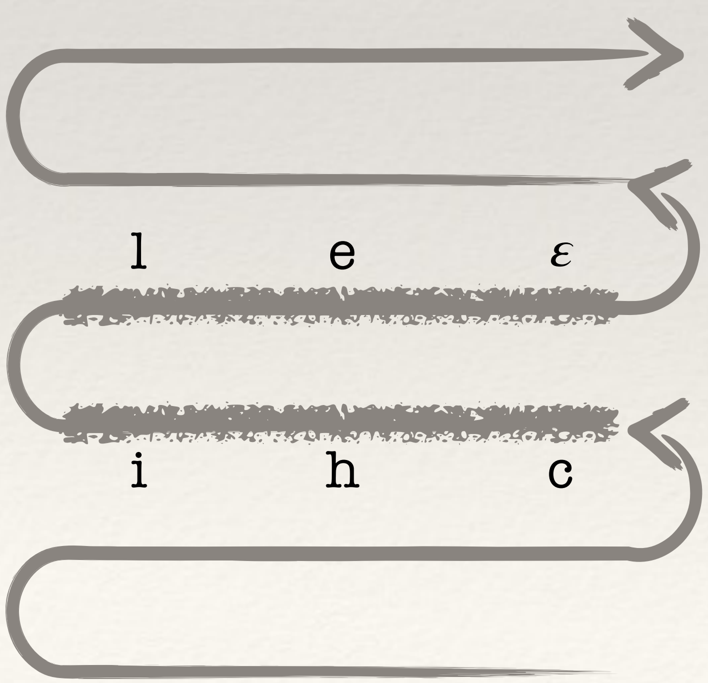
$$u \# v \mapsto v \# u$$

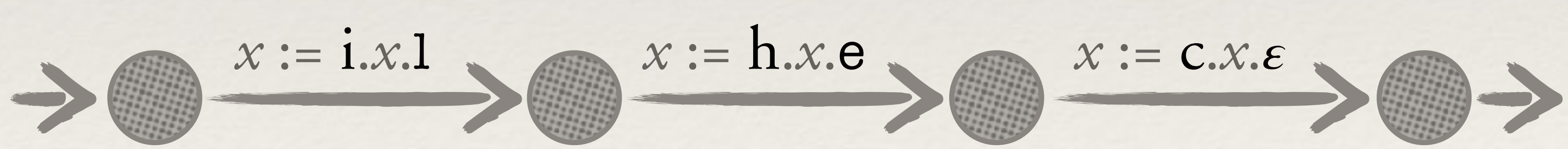




$2k$-sweep 2NFT	can be transformed into	k-register SST
k-register SST	can be transformed into	$2k$-sweep 2NFT

$2k$-sweep 2NFT	can be transformed into	k-register SST
k-register SST	can be transformed into	$2k$-sweep 2NFT

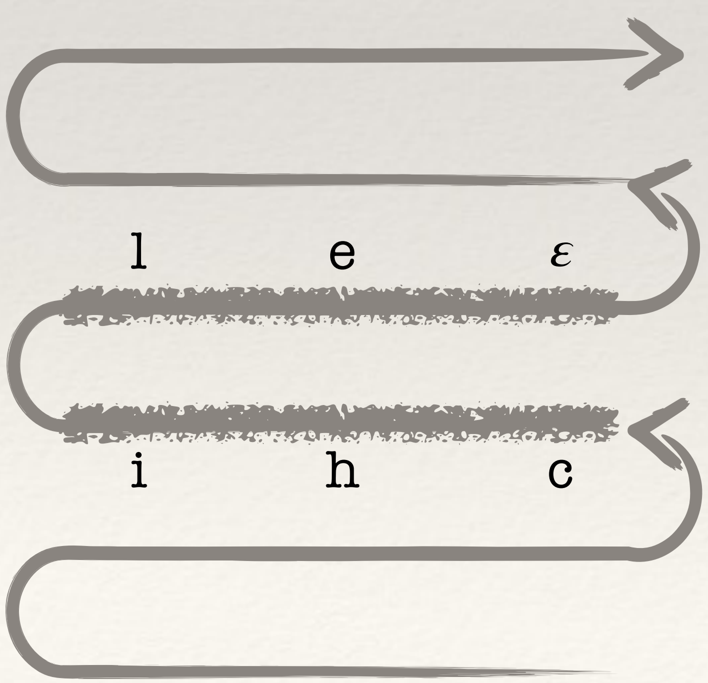


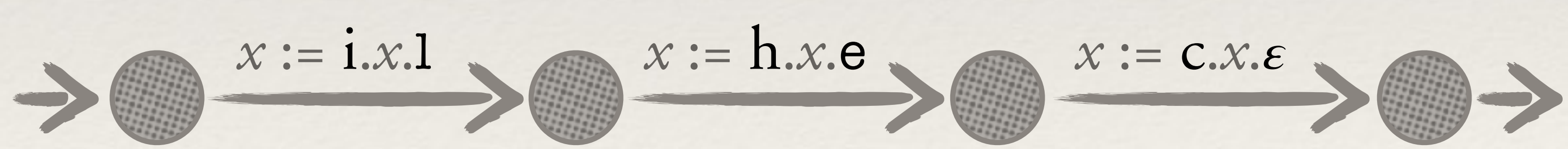


$2k$-sweep 2NFT	can be transformed★ into	k-register SST
k-register SST	can be transformed★ into	$2k$-sweep 2NFT

★ in 2EXPTIME

★ in EXPTIME





A characterization similar to 1-way definability:

Given a functional sweeping 2NFT T and a number *k*

- we can construct a *k*-sweep NFT $T' \subsetneq T$ (2EXPTIME)
- T is *k*-sweep definable iff $T' = T$
- we can decide the latter (EXPSPACE)

Given a sweeping 2NFT, we can compute:

- the minimum # of sweeps (ExpSpace)
- a sweeping 2NFT with the min. # of sweeps (2ExpTime)

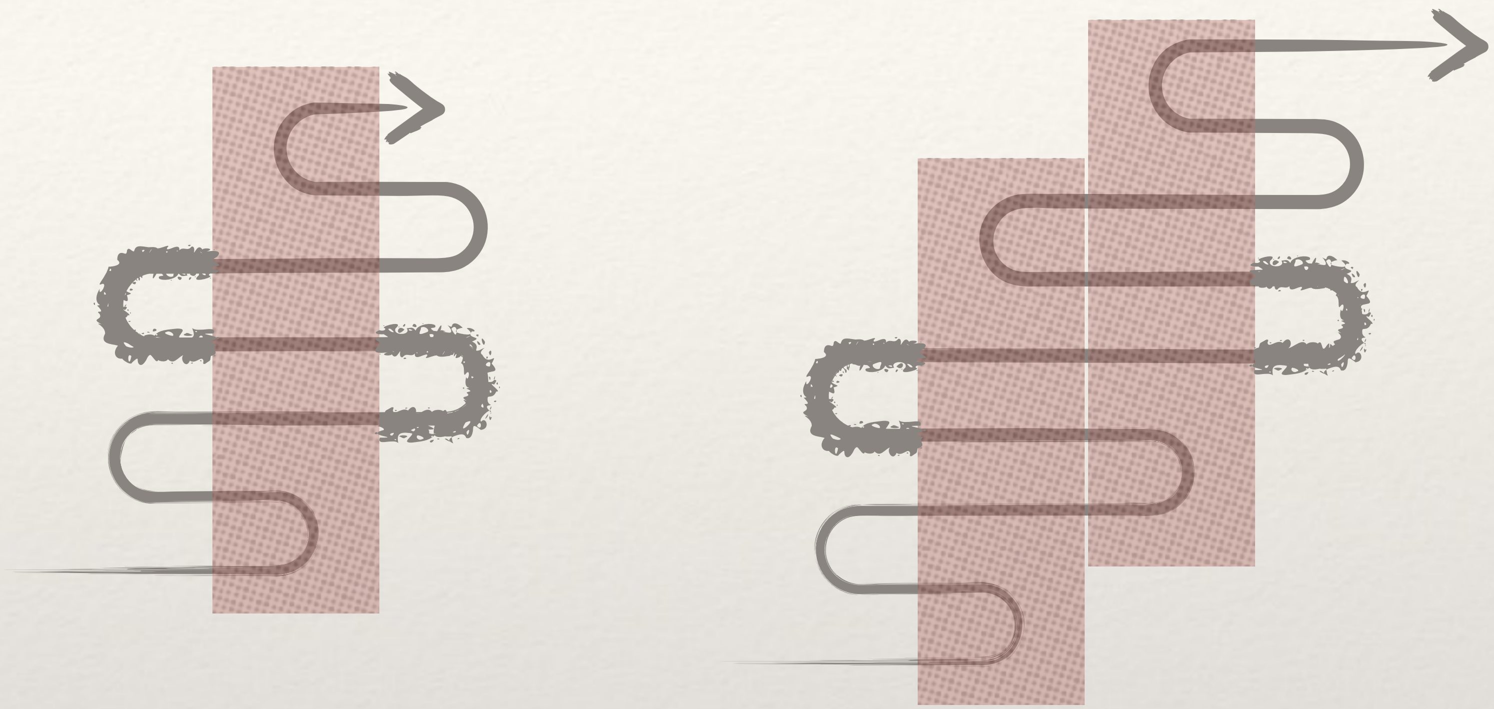
Given a sweeping 2NFT, we can compute:

- the minimum # of sweeps (EXPSPACE)
- a sweeping 2NFT with the min. # of sweeps (2EXPTIME)

Given a concatenation-free SST, we can compute:

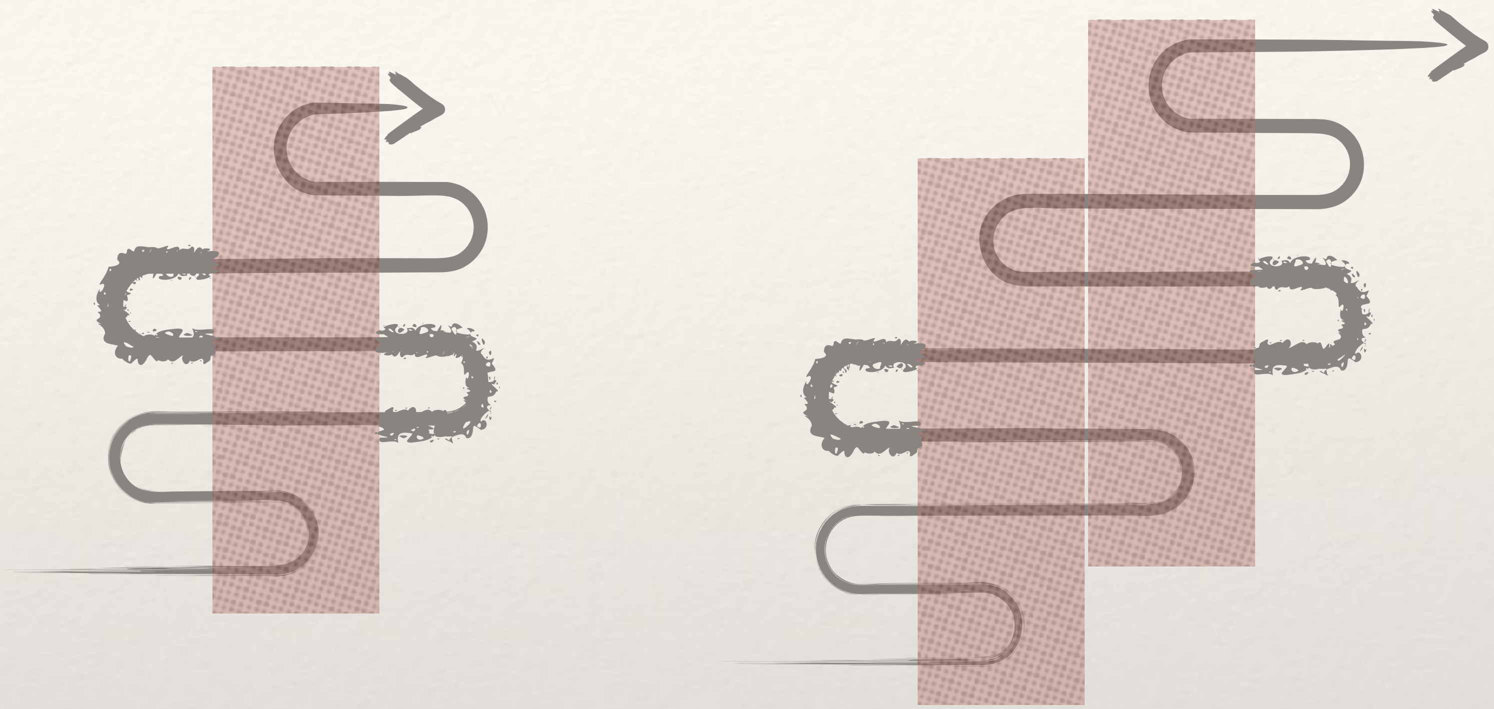
- the minimum # of registers (2EXPSPACE)
- a concatenation-free SST with the min. # of registers (3EXPTIME)

- Formalise the results for 2NFT (non-sweeping)



- Characterise *sweepingness* with unknown # of passes
- Minimise # of registers of SST (non concatenation-free)
- Find decidable non-functional cases (*k*-valuedness ?)

- Formalise the results for 2NFT (non-sweeping)



- Characterise *sweepingness* with unknown # of passes
- Minimise # of registers of SST (non concatenation-free)
- Find decidable non-functional cases (*k*-valuedness ?)