From Two-Way to One-Way Finite State Transducers

Emmanuel Filiot[1], Olivier Gauwin[3], P.-A. Reynier[2], Frédéric Servais[1]

[1]Université Libre de Bruxelles [2]Marseille Universty [3]Bordeaux University

Finite State Automata

- finite string acceptors over a finite alphabet Σ
- read-only input tape, left-to-right
- finite set of states

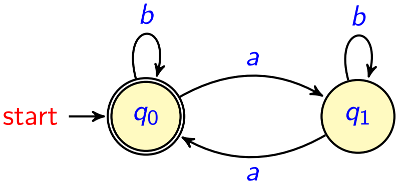
Definition (Finite State Automaton)

A finite state automaton (FA) on Σ is a tuple $A = (Q, I, F, \delta)$ where

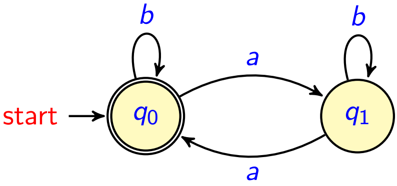
- Q is the set of states,
- $I \subseteq Q$, reps. $F \subseteq Q$ is the set of initial, resp. final, states,
- $\delta : Q \times \Sigma \to Q$ is the transition relation.

$$L(A) \;=\; \{w \in \Sigma^* \mid \text{there exists an accepting run on } w\}$$

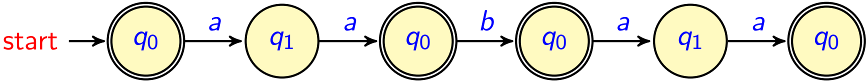
Finite State Automata – Example



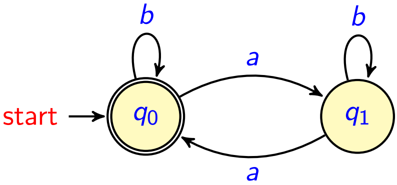
Finite State Automata – Example



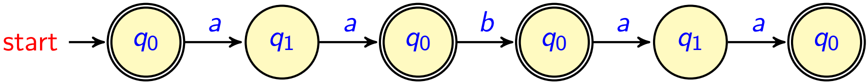
Run on *aabaa*:



Finite State Automata – Example



Run on *aabaa*:



$$L(A) = \{w \in \Sigma^* \mid w \text{ contains an even number of } a\}$$

From Languages to Transductions

Let Σ and Δ be two finite alphabets.

Definition

Language on Σ	Transduction from Σ to Δ
function from Σ^* to $\{0,1\}$	relation $R \subseteq \Sigma^* \times \Delta^*$
defined by automata	defined by transducers
accept strings	transform strings

transducer $=$ <u>automaton</u> $+$ output mechanism.

One-Way Finite State Transducers

Finite State Transducers

- read-only left-to-right input head
- write-only left-to-right output head
- finite set of states

Finite State Transducers

- read-only left-to-right input head
- write-only left-to-right output head
- finite set of states

Definition (Finite State Transducers)

A finite state transducer from Σ to Δ is a pair $T = (A, O)$ where

- $A = (Q, I, F, \delta)$ is the underlying automaton
- O is an output morphism from δ to Δ^*.

- If $t = q \xrightarrow{a} q' \in \delta$, then $O(t)$ defines its output.
- $q \xrightarrow{a|w} q'$ denotes a transition whose output is $w \in \Delta^*$.

Finite State Transducers

- read-only left-to-right input head
- write-only left-to-right output head
- finite set of states

Definition (Finite State Transducers)

A finite state transducer from Σ to Δ is a pair $T = (A, O)$ where

- $A = (Q, I, F, \delta)$ is the underlying automaton
- O is an output morphism from δ to Δ^*.

- If $t = q \xrightarrow{a} q' \in \delta$, then $O(t)$ defines its output.
- $q \xrightarrow{a|w} q'$ denotes a transition whose output is $w \in \Delta^*$.

Two classes of transducers:

- DFT if A is deterministic
- NFT if A is non-deterministic.

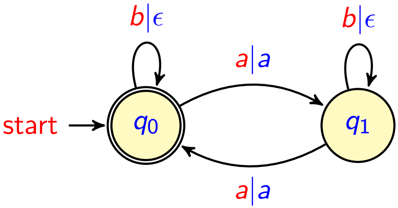
Some applications

- language and speech processing (e.g. see work by Mehryar Mohri)
- model-checking infinite state-space systems[1]
- string pattern matching
- verification of web sanitizers[2]

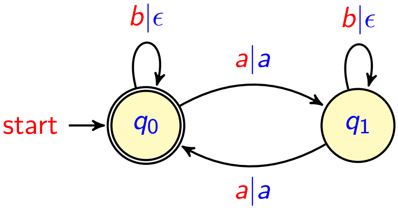
[1] A survey of regular model checking, P. Abdulla, B. Jonsson, M. Nilsson, M. Saksena. 2004

[2] see BEK, developped at Microsoft Research

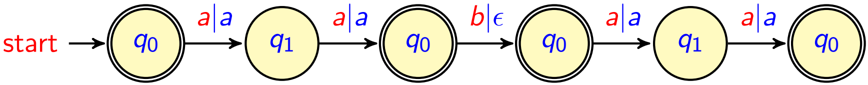
Finite State Transducers – Example 1



Finite State Transducers – Example 1

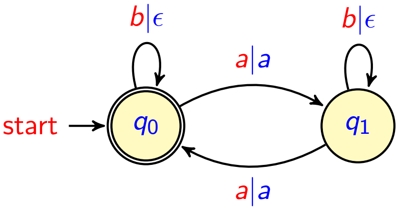


Run on *aabaa*:

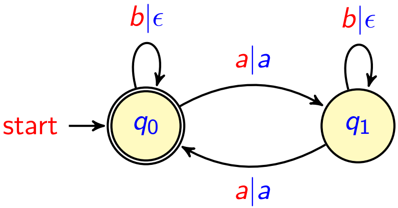


$$T(aabaa) = a.a.\epsilon.a.a = aaaa.$$

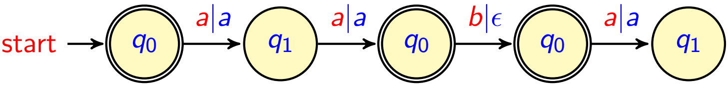
Finite State Transducers – Example 1



Finite State Transducers – Example 1

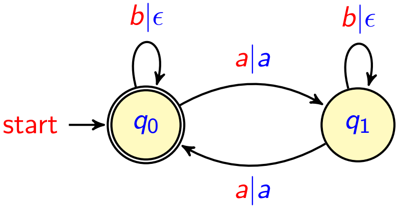


Run on *aaba*:



$T(aaba) =$ **undefined**

Finite State Transducers – Example 1

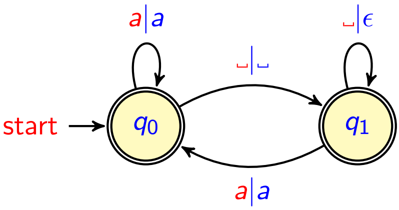


Semantics

$$
\begin{aligned}
dom(T) &= \{w \in \Sigma^* \mid \#_a w \text{ is even}\} \\
R(T) &= \{(w, a^{\#_a w}) \mid w \in dom(T)\}
\end{aligned}
$$

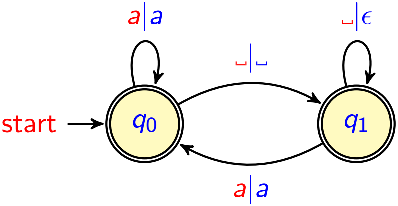
Finite State Transducers – Example 2

␣ = white space



Finite State Transducers – Example 2

$\textvisiblespace$ = white space



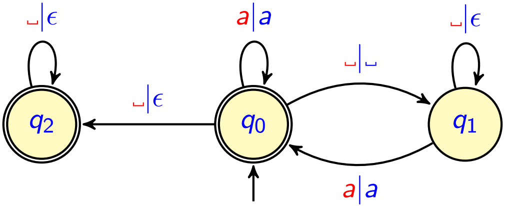
Semantics

Replace blocks of consecutive white spaces by a single white space.

$$T(\textvisiblespace\textvisiblespace aa\textvisiblespace\textvisiblespace\textvisiblespace a\textvisiblespace\textvisiblespace) = \textvisiblespace aa\textvisiblespace a\textvisiblespace$$

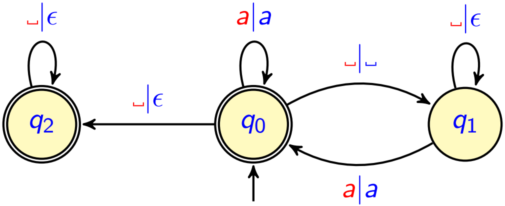
Finite State Transducers – Example 3

␣ = white space



Finite State Transducers – Example 3

$\llcorner\!\lrcorner$ = white space



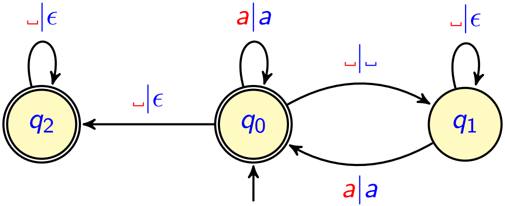
Semantics

Replace blocks of consecutive white spaces by a single white space
and
remove the last white spaces (if any).

$$T(\llcorner\!\lrcorner\llcorner\!\lrcorner aa \llcorner\!\lrcorner\llcorner\!\lrcorner\llcorner\!\lrcorner a \llcorner\!\lrcorner\llcorner\!\lrcorner) = \llcorner\!\lrcorner aa \llcorner\!\lrcorner a$$

Finite State Transducers – Example 3

␣ = white space



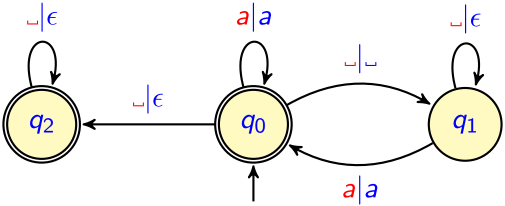
Semantics

Replace blocks of consecutive white spaces by a single white space
and
remove the last white spaces (if any).

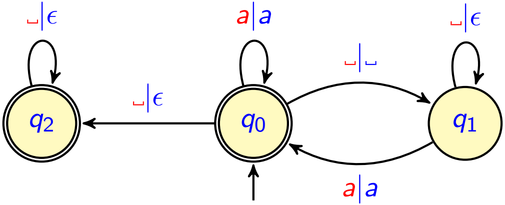
$$T(\text{␣␣}aa\text{␣␣␣}a\text{␣␣}) = \text{␣}aa\text{␣}a$$

Non-deterministic but still defines a function: functional NFT

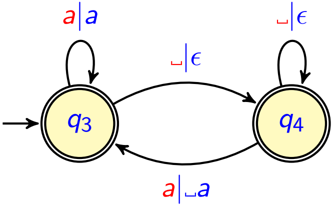
Is non-determinism needed ?



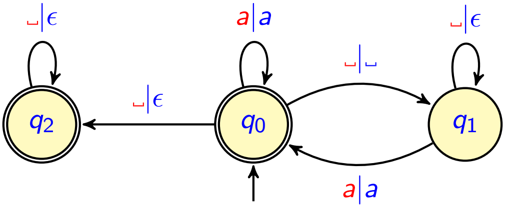
Is non-determinism needed ?



$\equiv$

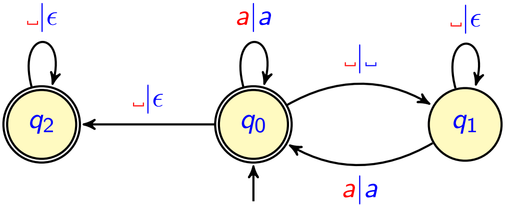


How to get a deterministic FT ?

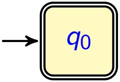


- extend automata subset construction with outputs
- output the longest common prefix

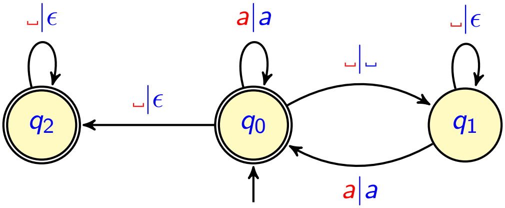
How to get a deterministic FT ?



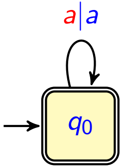
- extend automata subset construction with outputs
- output the longest common prefix



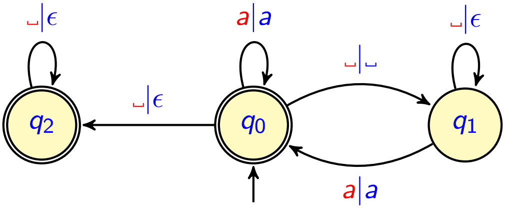
How to get a deterministic FT ?



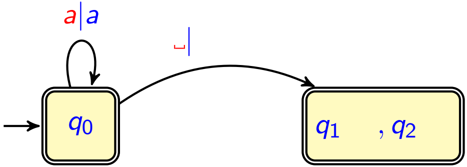
- extend automata subset construction with outputs
- output the longest common prefix



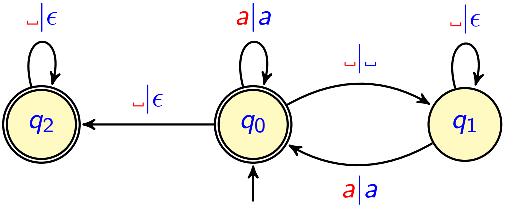
How to get a deterministic FT ?



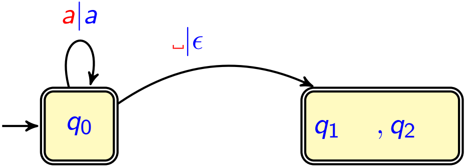
- extend automata subset construction with outputs
- output the longest common prefix



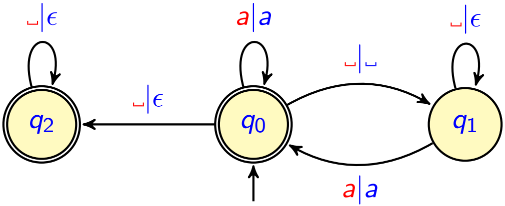
How to get a deterministic FT ?



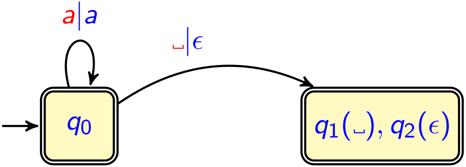
- extend automata subset construction with outputs
- output the longest common prefix



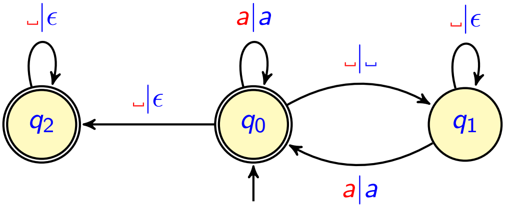
How to get a deterministic FT ?



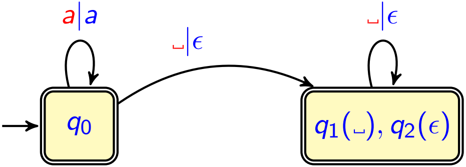
- extend automata subset construction with outputs
- output the longest common prefix



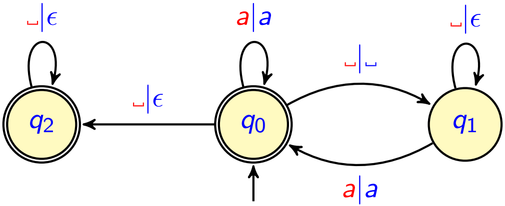
How to get a deterministic FT ?



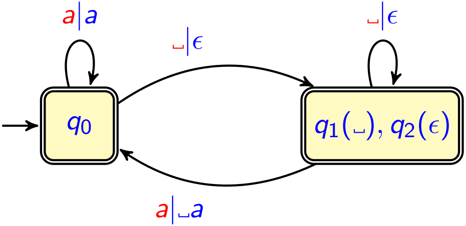
- extend automata subset construction with outputs
- output the longest common prefix



How to get a deterministic FT ?



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



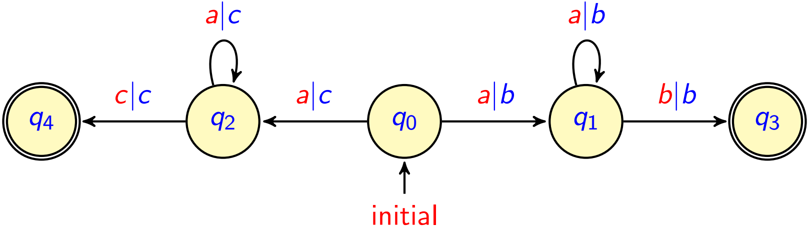
Can we always get an equivalent deterministic FT ?

Can we always get an equivalent deterministic FT ?

- not in general: DFT define functions, NFT define relations
- what about functional NFT ?

Can we always get an equivalent deterministic FT ?

- not in general: DFT define functions, NFT define relations
- what about functional NFT ?

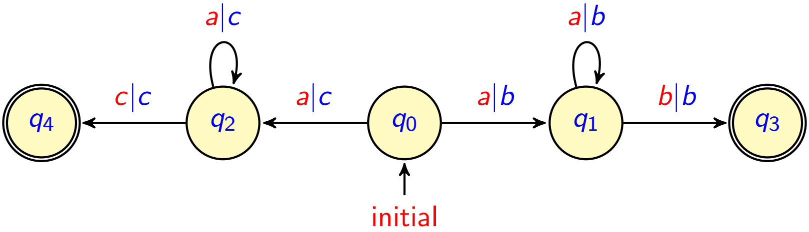


Semantics

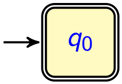
$$R(T) : \begin{cases} a^n b \mapsto b^{n+1} \\ a^n c \mapsto c^{n+1} \end{cases}$$

functional but not determinizable

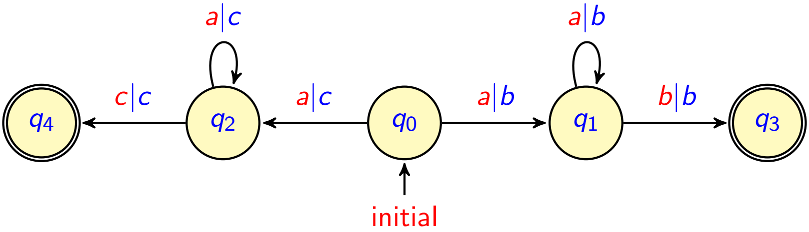
Subset construction fails ...



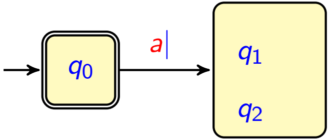
Subset construction:



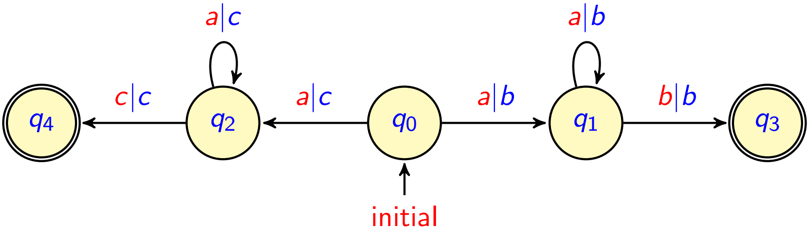
Subset construction fails ...



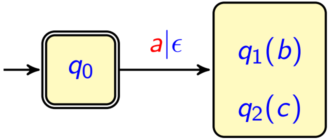
Subset construction:



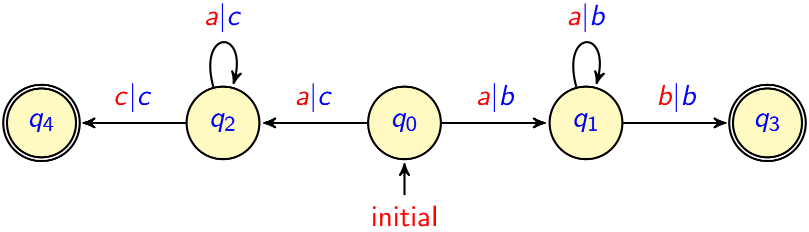
Subset construction fails ...



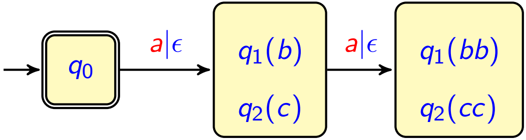
Subset construction:



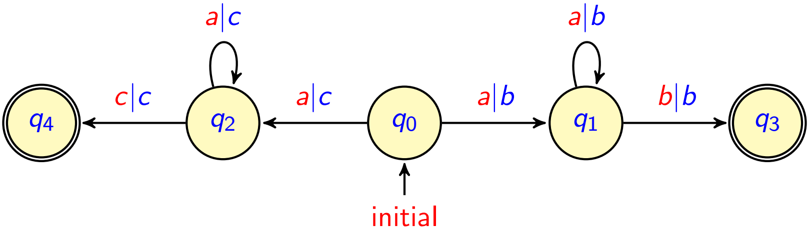
Subset construction fails ...



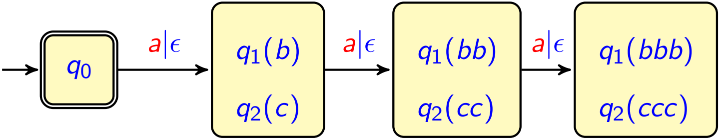
Subset construction:



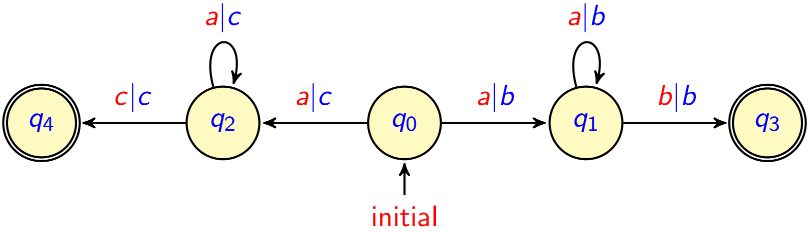
Subset construction fails ...



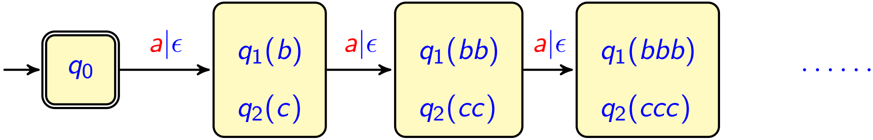
Subset construction:



Subset construction fails ...



Subset construction:



How to guarantee termination of subset construction?

LAG

$LAG(u, v) = (u', v')$ such that $u = \ell u'$, $v = \ell v'$ and $\ell = lcp(u, v)$.

E.g. $LAG(abbc, abc) = (bc, c)$.

How to guarantee termination of subset construction?

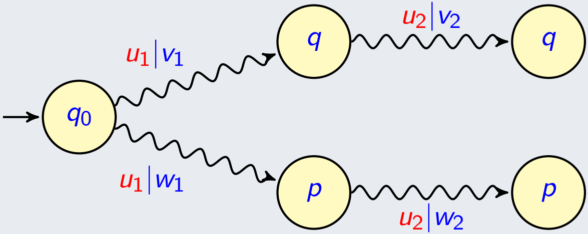
LAG

$LAG(u,v) = (u',v')$ such that $u = \ell u'$, $v = \ell v'$ and $\ell = lcp(u,v)$.

E.g. $LAG(abbc, abc) = (bc, c)$.

Lemma (Twinning Property)

Subset construction terminates **iff** *for all such situations*



it is the case that $LAG(v_1, w_1) = LAG(v_1v_2, w_1w_2)$.

Determinizability is decidable

Theorem (Choffrut 77, Beal Carton Prieur Sakarovitch 03)

Given a functional NFT T, the following are equivalent:

1. *it is determinizable*
2. *the twinning property holds.*

Moreover, the twinning property is decidable in PTime.

Application: analysis of streaming transformations

Bounded Memory Problem

Hypothesis:

- input string is received as a (very long) **stream**
- output string is produced as a stream

Input: a transformation defined by some functional NFT

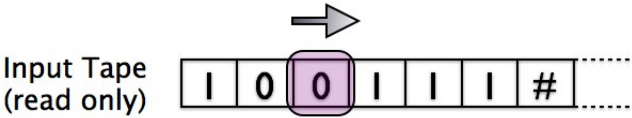
Output: can I realize this transformation with bounded memory ?

$$\exists B \in \mathbb{N} \cdot \forall u \in dom(T)$$

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

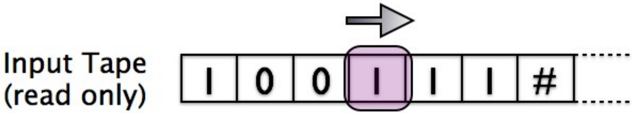
Streaming Model

Deterministic Turing Transducer



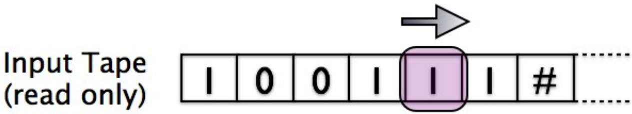
Streaming Model

Deterministic Turing Transducer



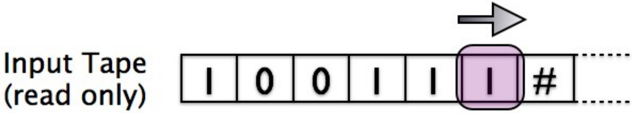
Streaming Model

Deterministic Turing Transducer



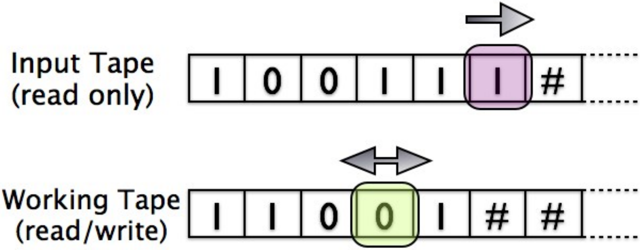
Streaming Model

Deterministic Turing Transducer



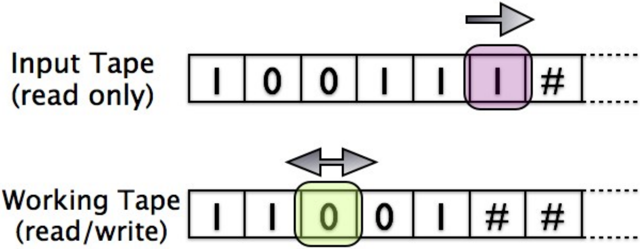
Streaming Model

Deterministic Turing Transducer



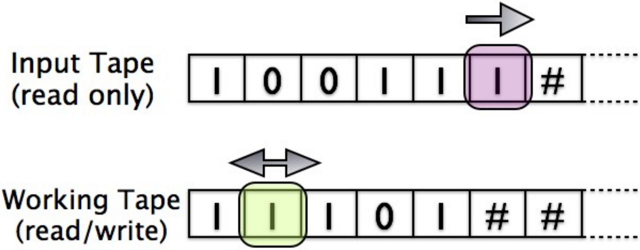
Streaming Model

Deterministic Turing Transducer



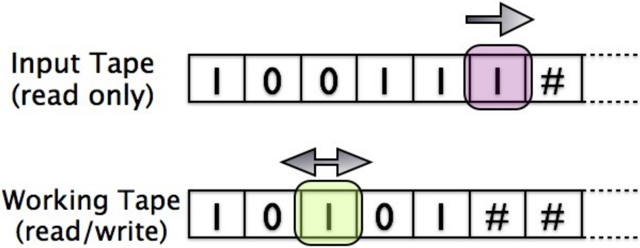
Streaming Model

Deterministic Turing Transducer



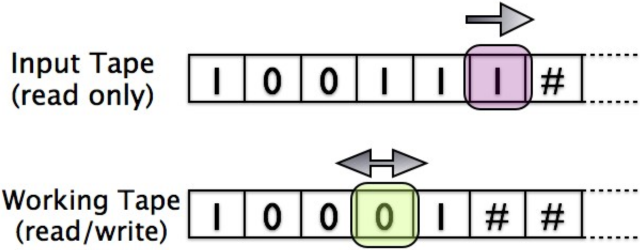
Streaming Model

Deterministic Turing Transducer



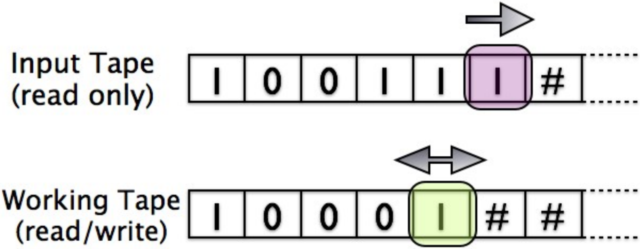
Streaming Model

Deterministic Turing Transducer



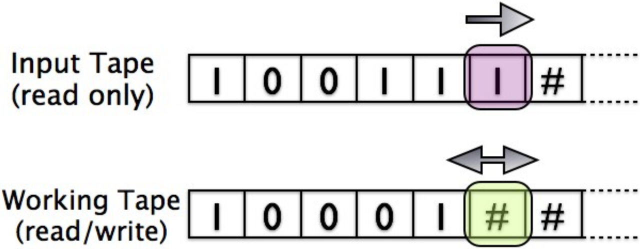
Streaming Model

Deterministic Turing Transducer



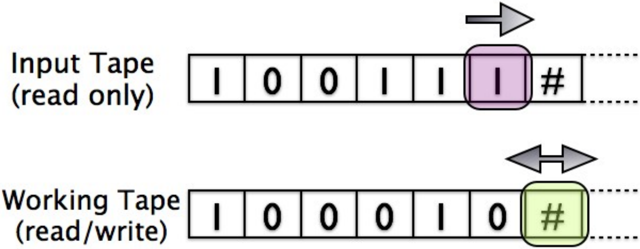
Streaming Model

Deterministic Turing Transducer



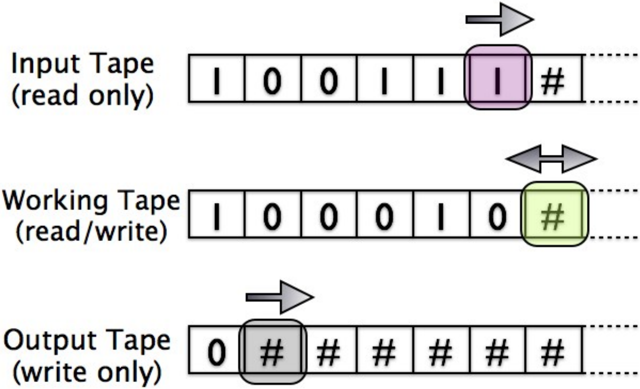
Streaming Model

Deterministic Turing Transducer



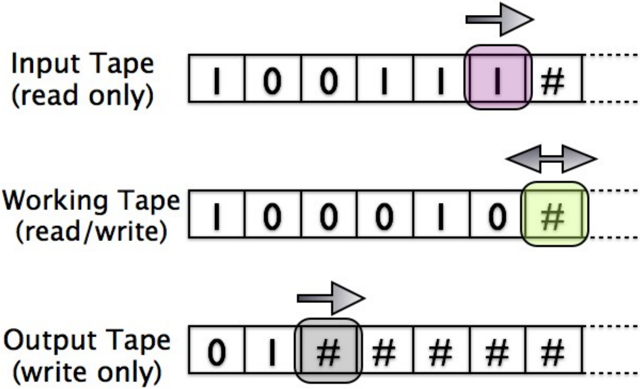
Streaming Model

Deterministic Turing Transducer



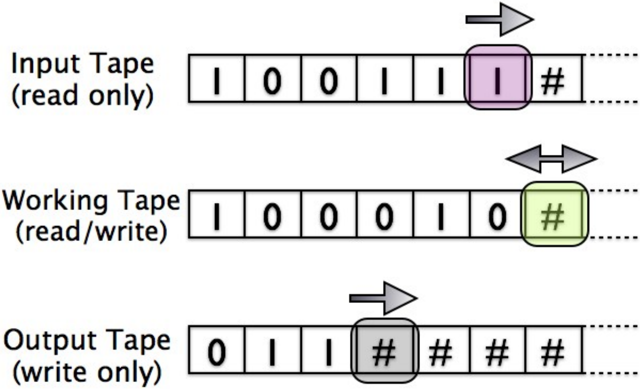
Streaming Model

Deterministic Turing Transducer



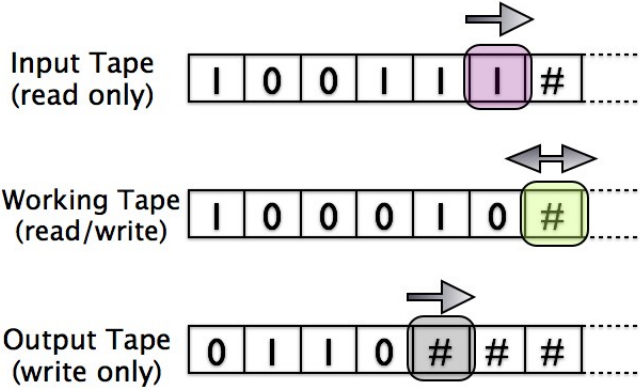
Streaming Model

Deterministic Turing Transducer



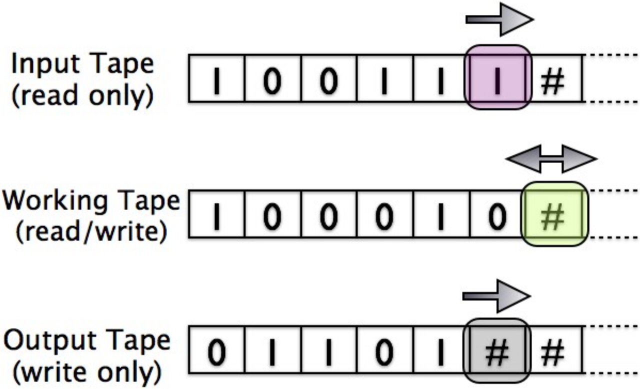
Streaming Model

Deterministic Turing Transducer



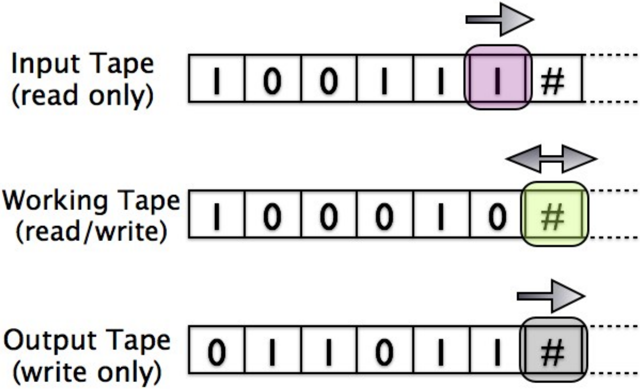
Streaming Model

Deterministic Turing Transducer



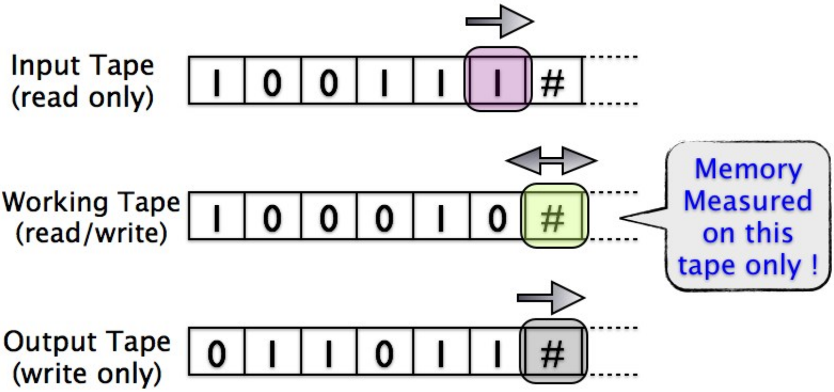
Streaming Model

Deterministic Turing Transducer



Streaming Model

Deterministic Turing Transducer



Bounded Memory Problem – Examples

$T_1 : \begin{cases} a^n b \mapsto b^{n+1} \\ a^n c \mapsto c^{n+1} \end{cases}$	Not bounded memory
$T_2 : \textvisiblespace\textvisiblespace a \textvisiblespace\textvisiblespace\textvisiblespace b \textvisiblespace\textvisiblespace \mapsto \textvisiblespace a \textvisiblespace b$	Bounded memory

Bounded Memory Problem – Examples

$T_1 : \begin{cases} a^n b \mapsto b^{n+1} \\ a^n c \mapsto c^{n+1} \end{cases}$	Not bounded memory
$T_2 : \sqcup\sqcup a \sqcup\sqcup\sqcup b \sqcup\sqcup \mapsto \sqcup a \sqcup b$	Bounded memory

Theorem

For all functional NFT T, the following are equivalent:

1. T *is bounded memory*
2. T *is determinizable*
3. T *satisfies the twinning property.*

Proof based on the following two observations:

1. any DFT is bounded memory
2. bounded memory Turing Transducer $\equiv$ DFT

Corollary

Corollary

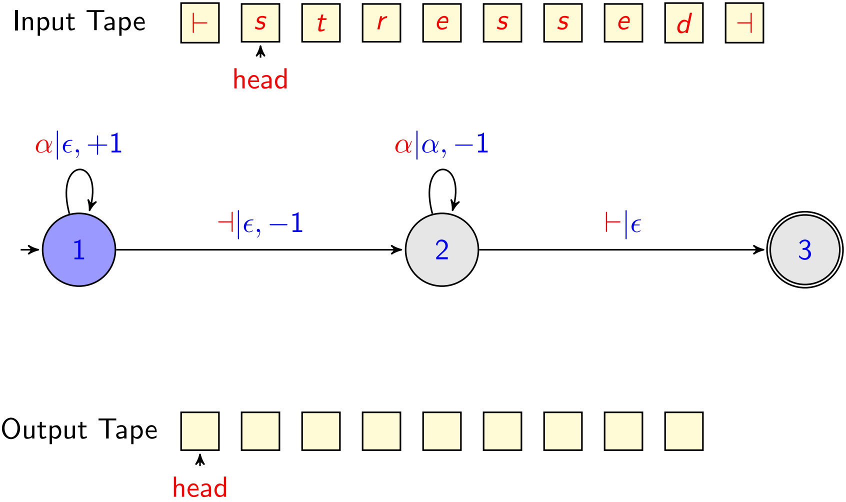
For all transductions R, the following are equivalent:

1. R *is computable with bounded memory*
2. R *is definable by some DFT*

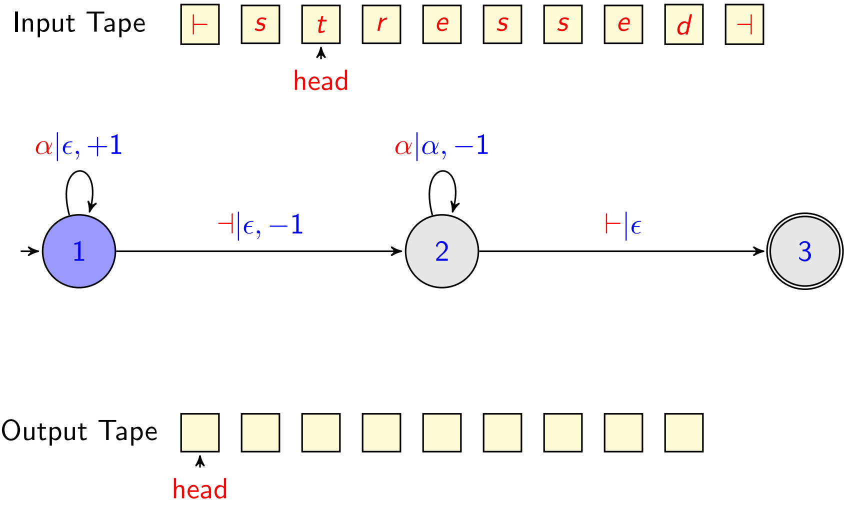
Two-Way Finite State Transducers

extending finite state transducers with a two-way input tape.

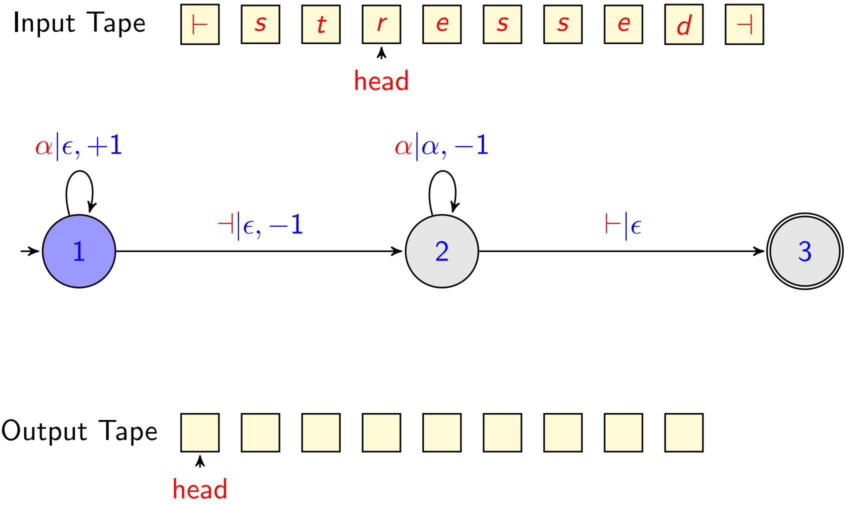
Two-way finite state transducers (2NFT)



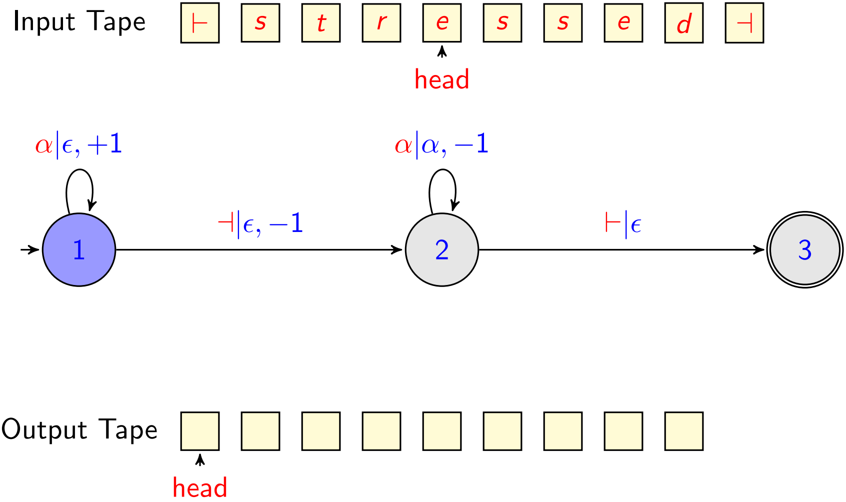
Two-way finite state transducers (2NFT)



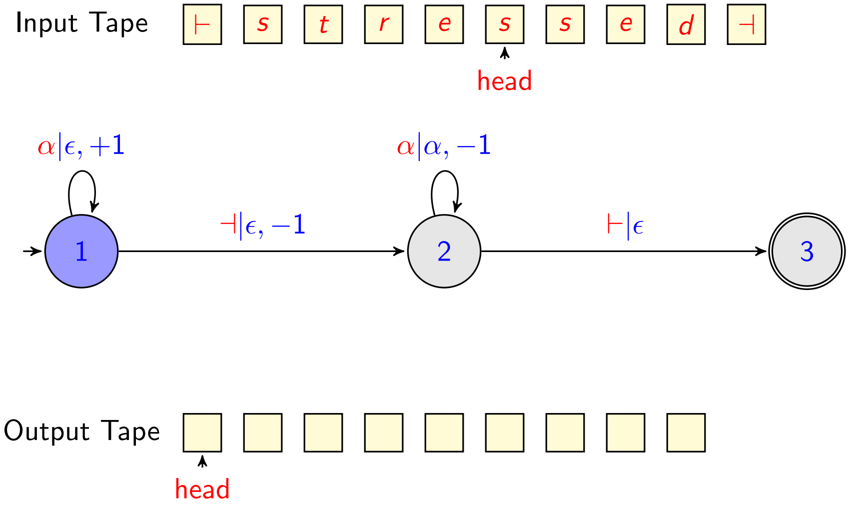
Two-way finite state transducers (2NFT)



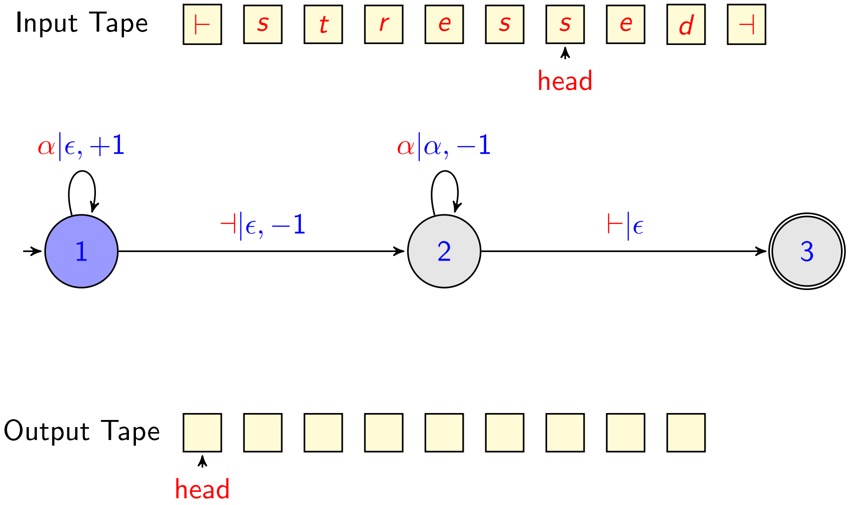
Two-way finite state transducers (2NFT)



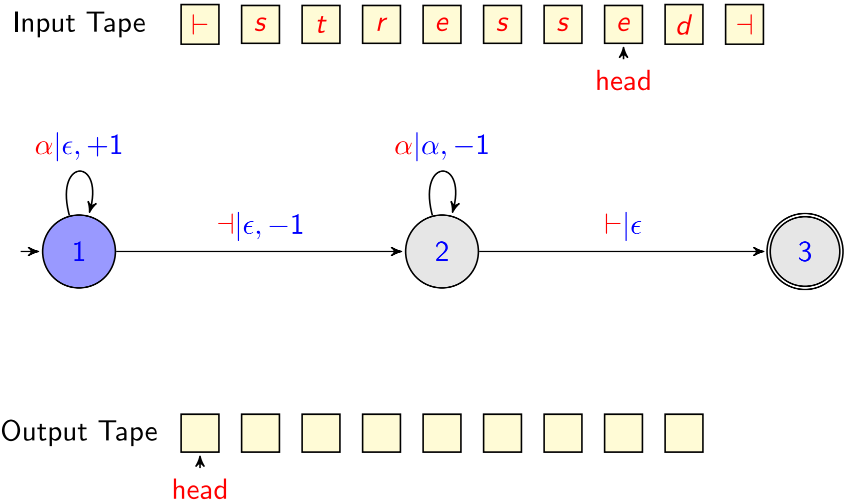
Two-way finite state transducers (2NFT)



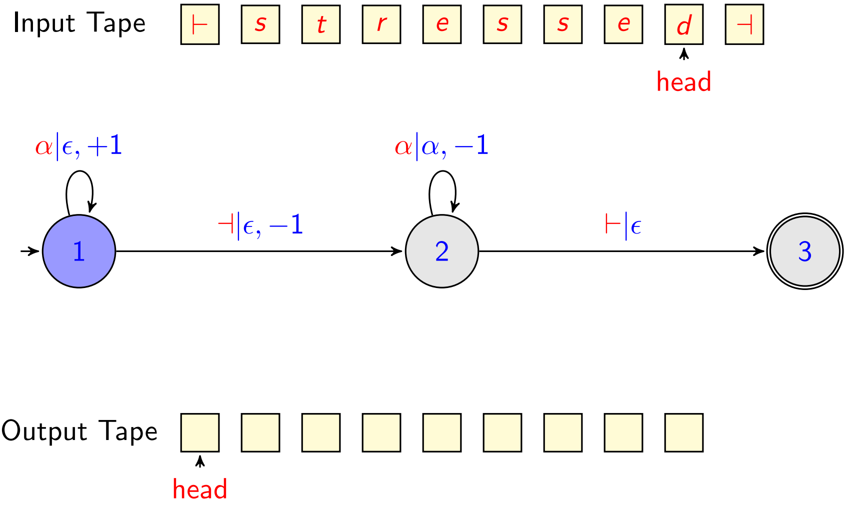
Two-way finite state transducers (2NFT)



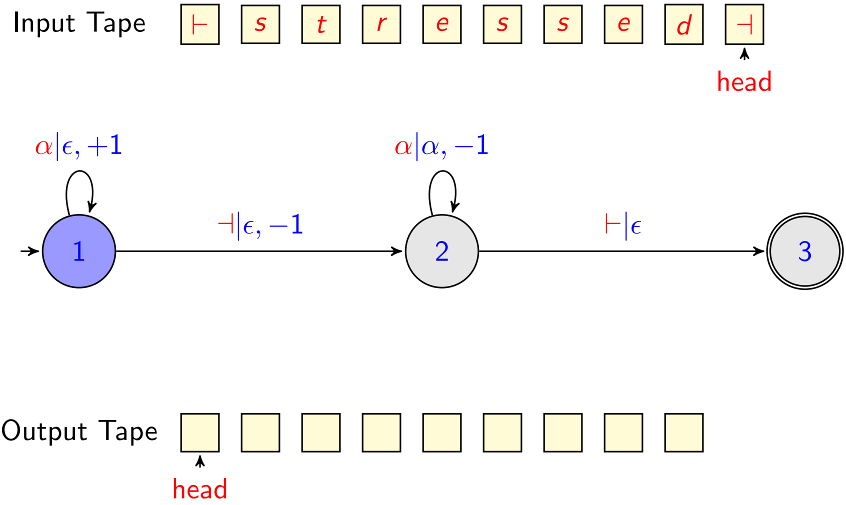
Two-way finite state transducers (2NFT)



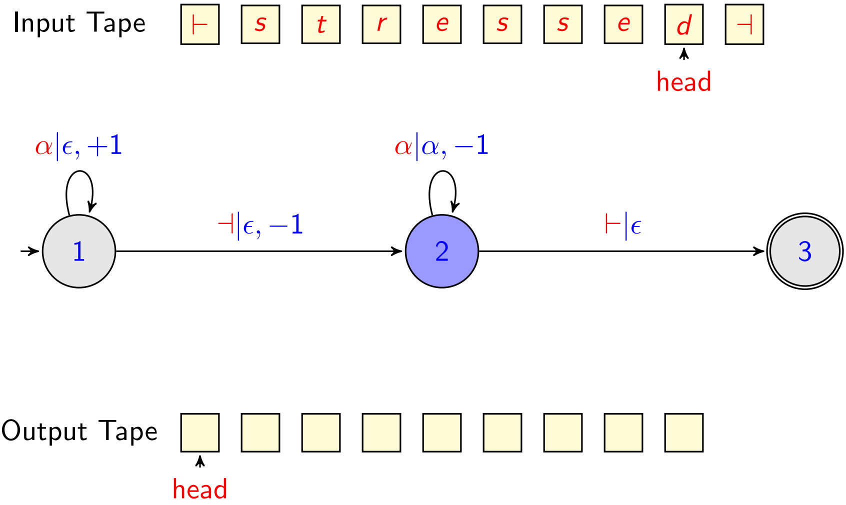
Two-way finite state transducers (2NFT)



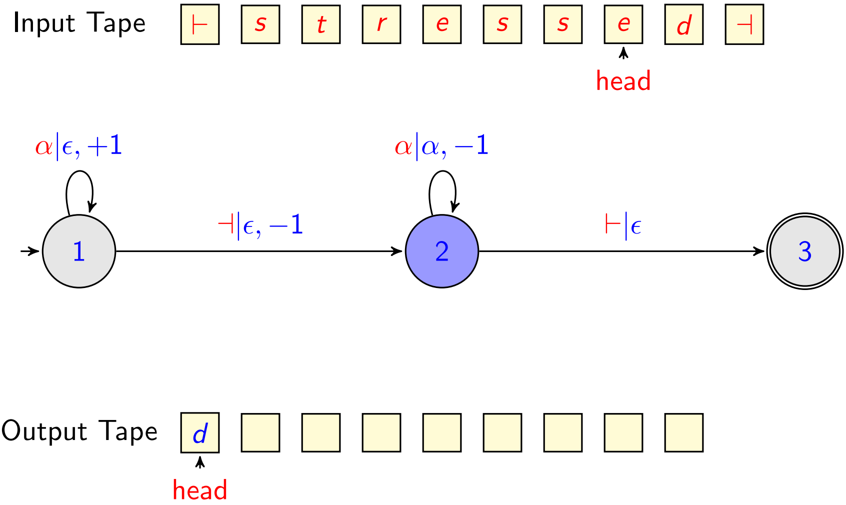
Two-way finite state transducers (2NFT)



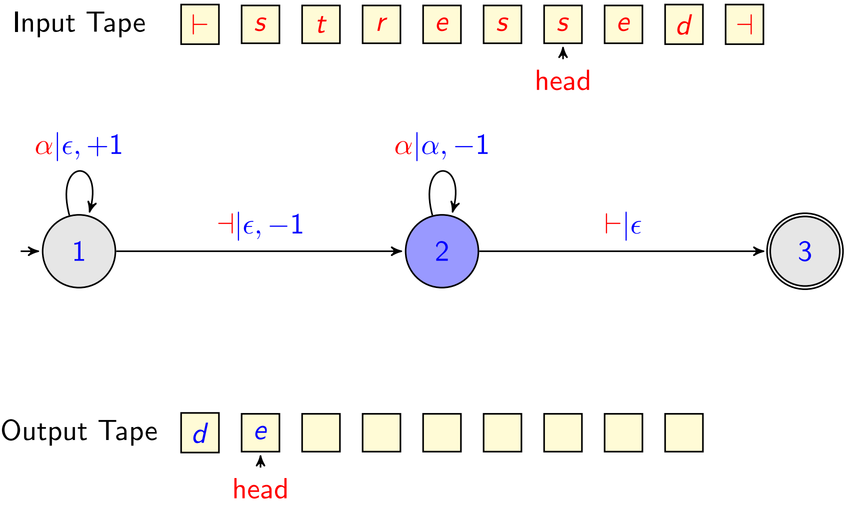
Two-way finite state transducers (2NFT)



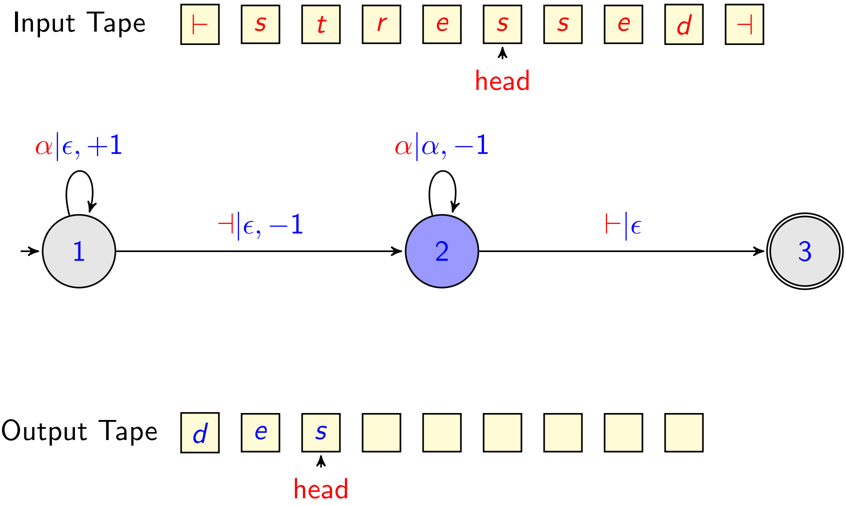
Two-way finite state transducers (2NFT)



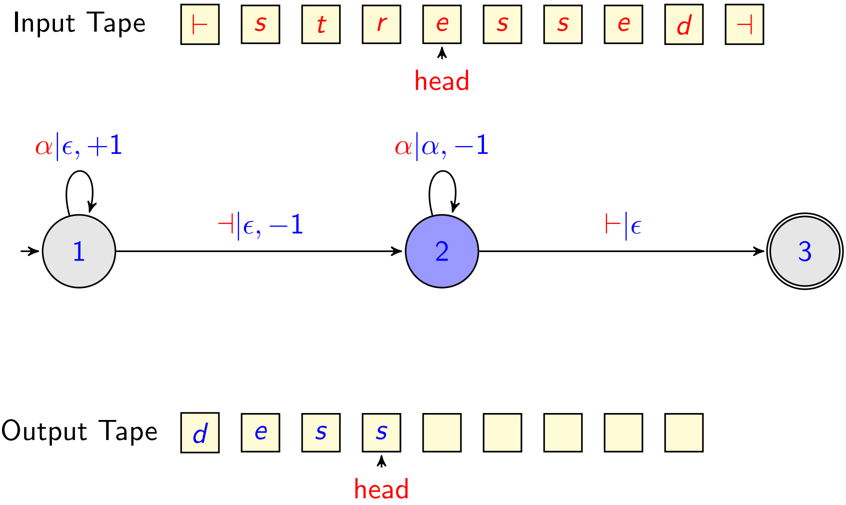
Two-way finite state transducers (2NFT)



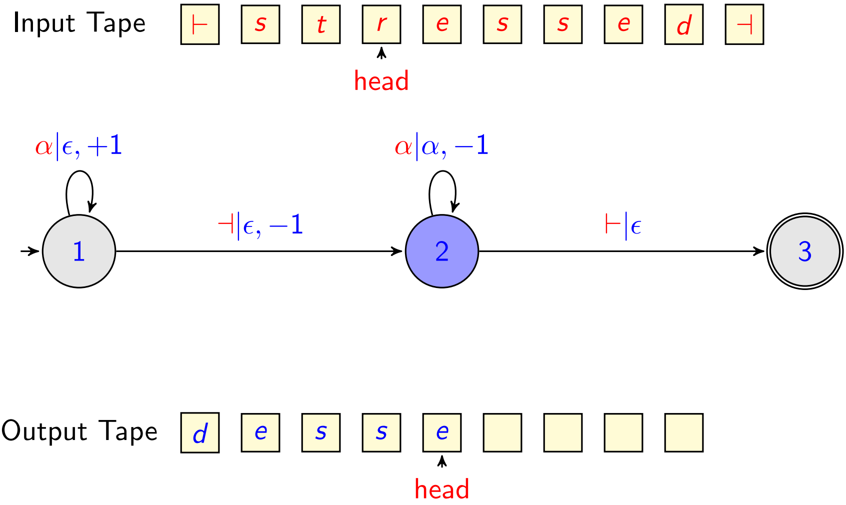
Two-way finite state transducers (2NFT)



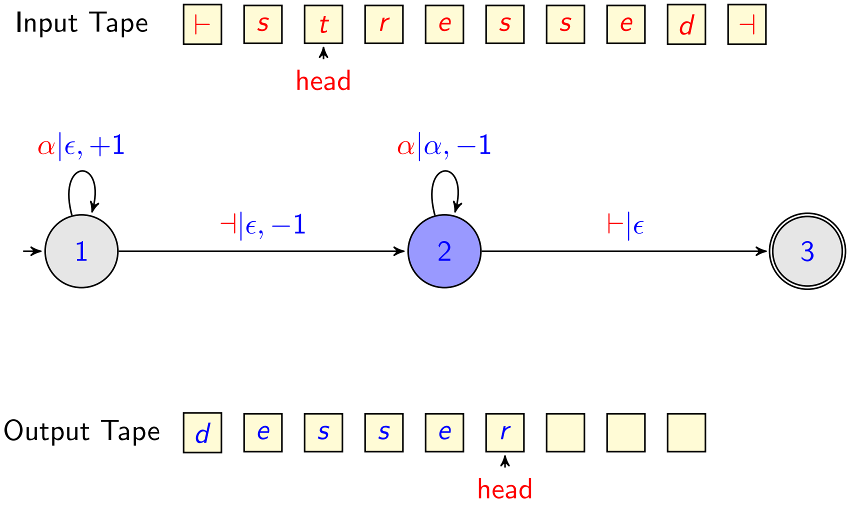
Two-way finite state transducers (2NFT)



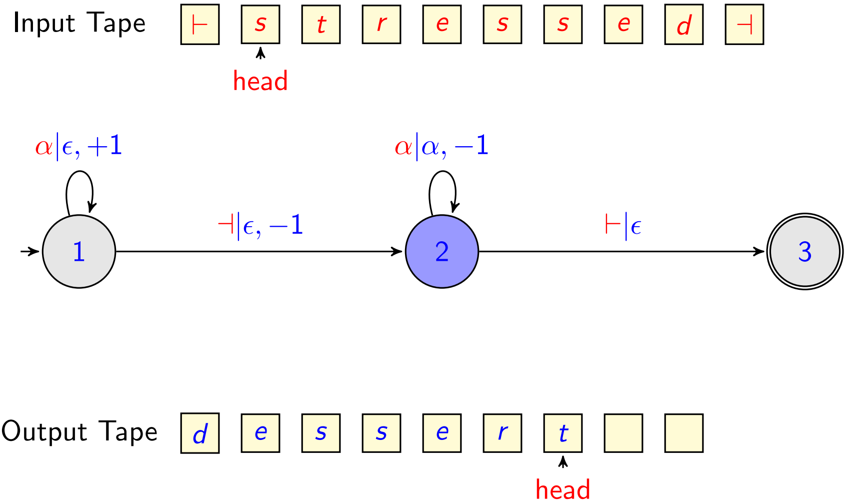
Two-way finite state transducers (2NFT)



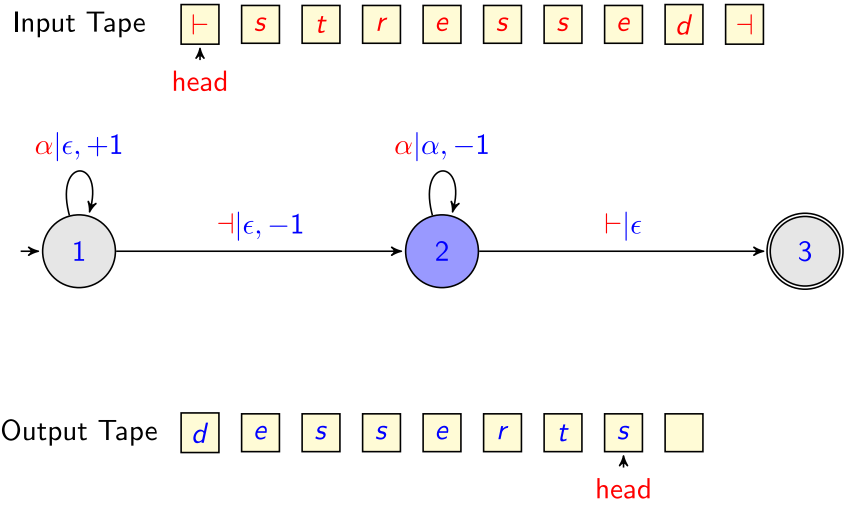
Two-way finite state transducers (2NFT)



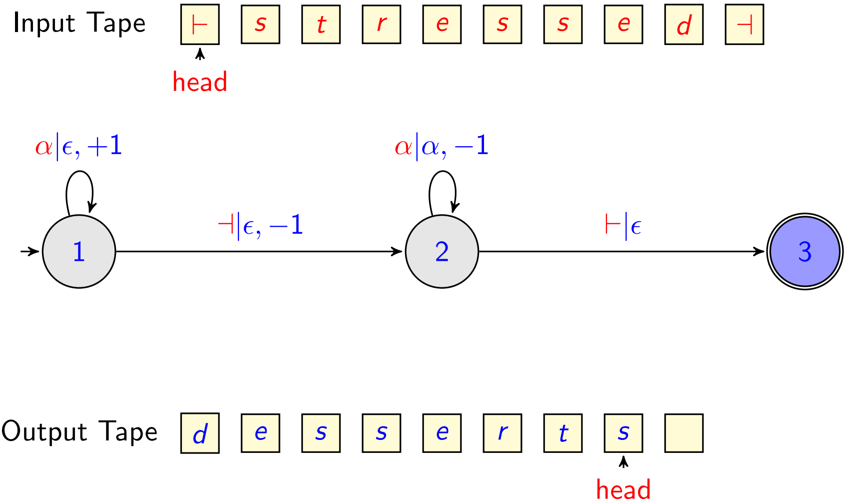
Two-way finite state transducers (2NFT)



Two-way finite state transducers (2NFT)



Two-way finite state transducers (2NFT)



Two-way finite state transducers – Properties

Main Properties of 2NFT

1. closed under composition (Chytil Jakl 77)
2. equivalence of functional 2NFT is decidable (Culik, Karhumaki, 87)
3. functional 2NFT $\equiv$ 2DFT (Hoogeboom Engelfriet 01, De Souza 13)

Logical Characterization (Hoogeboom Engelfriet 01)

$$\text{2DFT} \equiv \text{MSO transductions}$$

2DFT define regular functions.

Two-way finite state transducers – Properties

Main Properties of 2NFT

1. closed under composition (Chytil Jakl 77)
2. equivalence of functional 2NFT is decidable (Culik, Karhumaki, 87)
3. functional 2NFT $\equiv$ 2DFT (Hoogeboom Engelfriet 01, De Souza 13)

Logical Characterization (Hoogeboom Engelfriet 01)

$$\text{2DFT} \equiv \text{MSO transductions}$$

2DFT define regular functions.

Also characterized by (deterministic) streaming string transducers (Alur Cerny 10).

Motivation: bounded memory problem

Question

Is the bounded memory problem decidable for 2DFT ?

Motivation: bounded memory problem

Question

Is the bounded memory problem decidable for 2DFT ?

Necessary Condition

The transduction must be definable by a one-way finite state transducer.

Indeed, bounded memory computable transductions $\equiv$ 1DFT-definable transductions

Summary

D=”(input) deterministic”

f=”functional”

DFTs	fNFTs	NFTs
2DFTs	f2NFTs	2NFTs

Summary

D="(input) deterministic"

f="functional"

$\vdash u \dashv \;\mapsto\;$ mirror(u)

DFTs	fNFTs	NFTs
$\cap$	$\cap$	$\cap$
2DFTs	f2NFTs	2NFTs

Summary

D="(input) deterministic"

f="functional"

$\{(a,a),(a,b)\}$

DFTs		fNFTs	$\subsetneq$	NFTs
$\cap$		$\cap$		$\cap$
2DFTs		f2NFTs	$\subsetneq$	2NFTs

Summary

D="(input) deterministic"

f="functional"

$a^n\alpha \mapsto \alpha^{n+1}$

$$
\begin{array}{ccccc}
\text{DFTs} & \subset & \text{fNFTs} & \subset & \text{NFTs} \\
\cap & & \cap & & \cap \\
\text{2DFTs} & & \text{f2NFTs} & \subset & \text{2NFTs}
\end{array}
$$

Summary

D=”(input) deterministic”

f=”functional”

DFTs	$\subset$	fNFTs	$\subset$	NFTs
$\cap$		$\cap$		$\cap$
2DFTs	$\equiv$	f2NFTs	$\subset$	2NFTs

[De Souza (13)]

$\equiv$MSOT [Engelfriet,Hoogeboom (01)]

$\equiv$ Streaming String Transducers [Alur, Černý, 2010]

Summary

D="(input) deterministic"

f="functional"

PTIME

[Choffrut (77)]

[Weber, Klemm (95)]

[Beal,Carton,Prieur,Sakarovitch(03)]

?

$$\text{DFTs} \subset \text{fNFTs} \subset \text{NFTs}$$

$$\cap \qquad\qquad \cap \qquad\qquad \cap$$

$$\text{2DFTs} \equiv \text{f2NFTs} \subset \text{2NFTs}$$

[De Souza (13)]

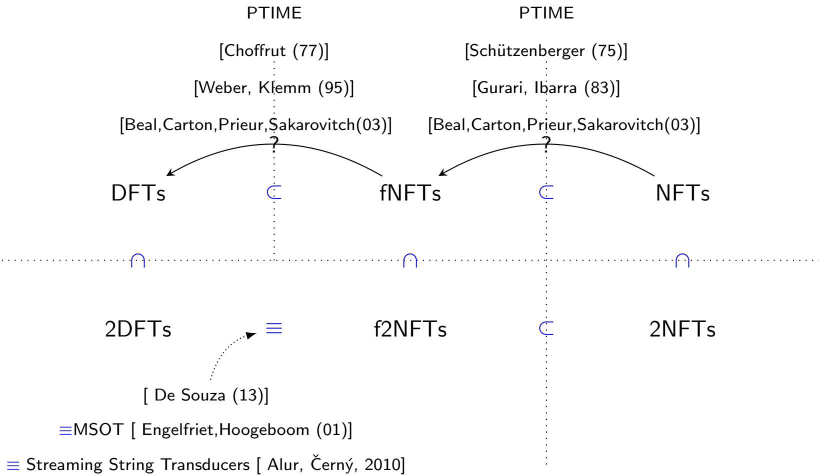
≡MSOT [Engelfriet,Hoogeboom (01)]

≡ Streaming String Transducers [Alur, Černý, 2010]

Summary

D="(input) deterministic"

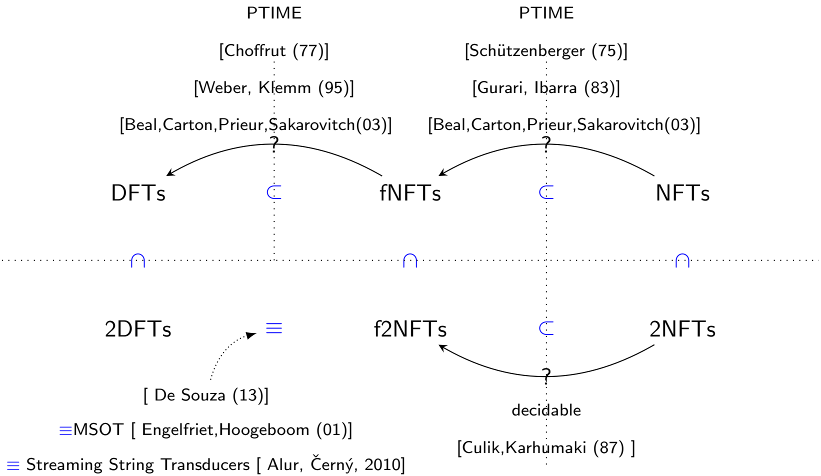
f="functional"



Summary

D="(input) deterministic"

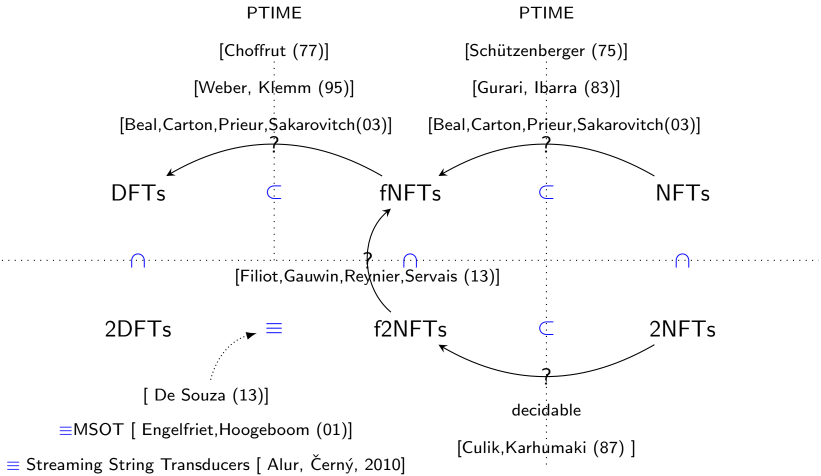
f="functional"



Summary

D="(input) deterministic"

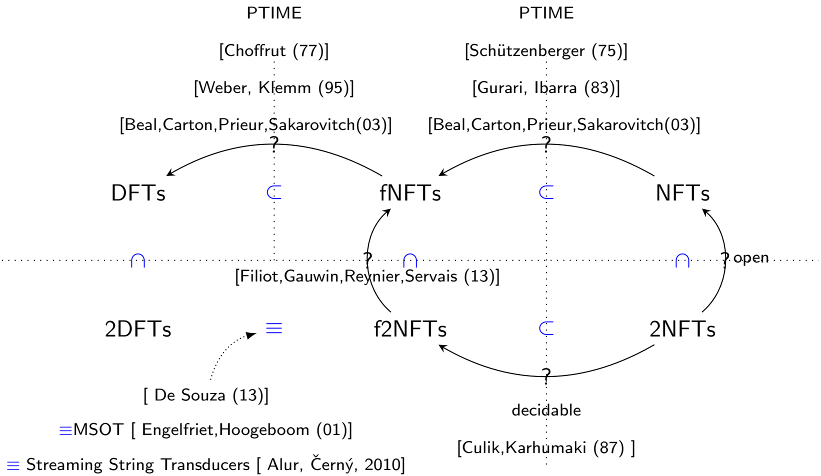
f="functional"



Summary

D="(input) deterministic"

f="functional"



From Two-Way to One-Way Finite State Automata

1DFA vs 2DFA

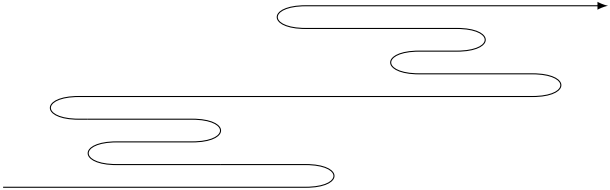
Theorem

For every 2NFA *there exists an equivalent* 1NFA.

- The first proof was done by Rabin and Scott (1959).
- In the same journal Shepherdson (1959) also published a (simpler) proof. Also rephrased, in an even simpler way, by Ullman.
- Vardi (1981) presented a different proof.
- The R&S proof is more easily adapted to transducers.

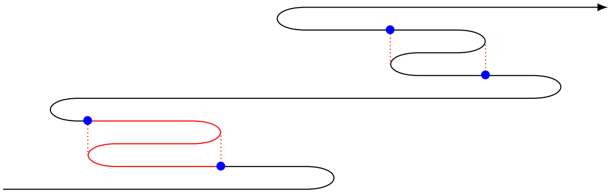
Rabin and Scott's proof for 2-Automata

- a run is made of many zigzags (moves of the input head)



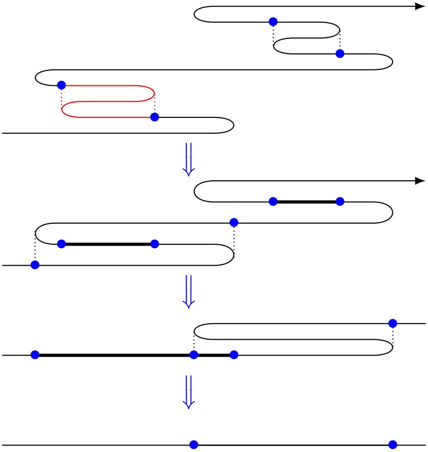
Rabin and Scott's proof for 2-Automata

- a run is made of many zigzags (moves of the input head)

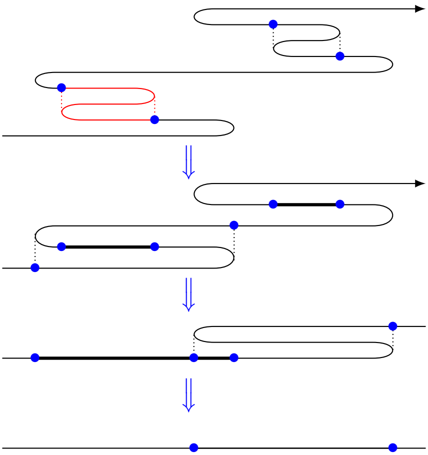


- A z-motion is an elementary zigzag.

Rabin and Scott's Proof: z-motions removal

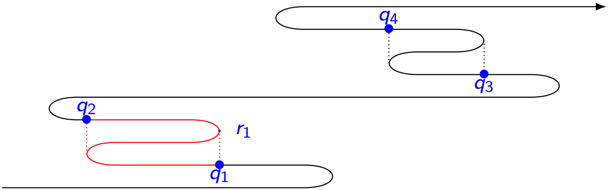


Rabin and Scott's Proof: z-motions removal



- **Def**: A shape is k crossing if any position is visited at most k times.
- **Thm**: Any k-crossing shape can be reduced to a line in k^2 steps.
- **Prop**: $\forall w \in L(A)$, w is accepted by a $|Q|$ crossing run.

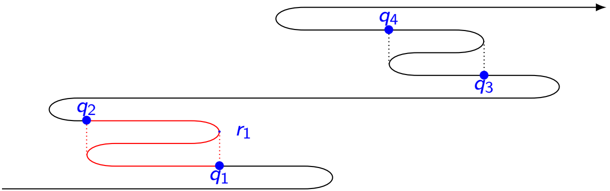
squeeze(A)



$S = \mathrm{squeeze}(A)$ removes some z-motions of A.

1. simulates A
2. non-deterministically guesses that a z-motion starts (e.g. from q_1 to q_2)
3. **checks** that is indeed a z-motion and **simulates** it one-way
4. goes back to mode 1

squeeze(A)

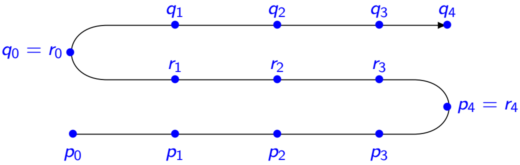


$S = \text{squeeze}(A)$ removes some z-motions of A.

Iterate squeeze(A)

- Every accepted word has a one-way run in $\text{squeeze}^{|Q|^2}(A)$ $\Longrightarrow$ remove backward transitions to obtain a 1NFA equivalent to A.

How to simulate a z-motion run in one-way ?



Simulate the three passes in parallel ! (with triple of states)

Extension to transducers

- same canvas (Rabin and Scott)
- removal of z-motion:
  ➜ translate a z-motion transducer into a fNFT
- not always possible $\rightarrow$ decision procedure

Extension to transducers

- same canvas (Rabin and Scott)
- removal of z-motion:
  ➜ translate a z-motion transducer into a fNFT
- not always possible $\rightarrow$ decision procedure

Remarks:

- if local z-motion transductions are 1-way definable, then $\mathrm{squeeze}(T)$ can be defined
- iterate $\mathrm{squeeze}(T)$ $|Q|^2$ times (if possible), you get an equivalent 1-NFT

Extension to transducers

- same canvas (Rabin and Scott)
- removal of z-motion:
  ➜ translate a z-motion transducer into a fNFT
- not always possible $\rightarrow$ decision procedure

Remarks:

- if local z-motion transductions are 1-way definable, then $\mathrm{squeeze}(T)$ can be defined
- iterate $\mathrm{squeeze}(T)$ $|Q|^2$ times (if possible), you get an equivalent 1-NFT

Results:

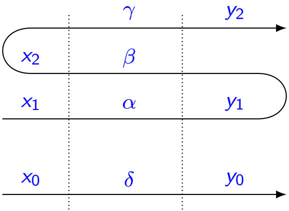
- decisition procedure to test whether a z-motion-transducer is 1-way definable
- the algorithm is complete

Decision procedure

Let T be a f2NFT.

1. repeat $|Q|^2$ times:
   - are all z-motion transductions of T NFT-definable?
     - yes: $T \leftarrow \mathrm{squeeze}(T)$
     - no: STOP: the initial 2NFTwas not NFT-definable!
2. remove backward transitions: you get an equivalent NFT

Towards a characterization of 1-way definable z-motion-transductions



$$x_1 \cdot \alpha^n \cdot y_1 \cdot \beta^n \cdot x_2 \cdot \gamma^n \cdot y_2 = x_0 \cdot \delta^n \cdot y_0$$

Lemma (Fine and Wilf (56))

Let $u, v \in \Sigma^*$. If u^ω and v^ω have a sufficiently large common factor, then $u \in (w_1 w_2)^*$ and $v \in (w_2 w_1)^*$ for some $w_1, w_2 \in \Sigma^*$.

$\Longrightarrow$ α,β,γ,δ have conjugate primitive roots (if $\neq \epsilon$).

$\rightarrow$ case analysis, depending on the emptiness of α, β, γ

Conclusion

Theorem

1. *It is decidable whether a 2DFT is definable by a 1NFT.*
2. *It is decidable whether a 2DFT is definable by a 1DFT.*
3. *Bounded memory is decidable for regular string transductions.*

- Complexity: non-elementary upper-bound, PSpace-hard.

Conclusion

Theorem

1. *It is decidable whether a 2DFT is definable by a 1NFT.*
2. *It is decidable whether a 2DFT is definable by a 1DFT.*
3. *Bounded memory is decidable for regular string transductions.*

- Complexity: non-elementary upper-bound, PSpace-hard.

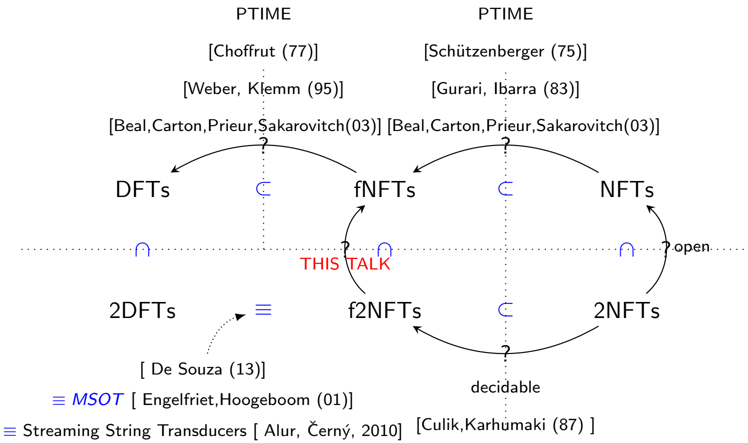
Future Work

- Lower complexity (Shepherdson)
- What about 2NFT(even non functional) ?
- Consider other structures: infinite strings, trees
- Variable minimization in streaming string transducers

Classes of Transductions

D="(input) deterministic"

f="functional"

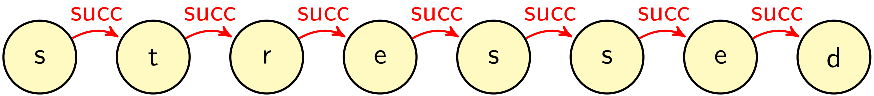


MSO Transductions (Courcelle)

- 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

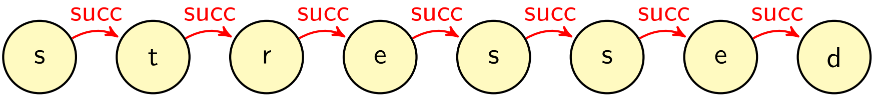
MSO Transductions (Courcelle)

- 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)

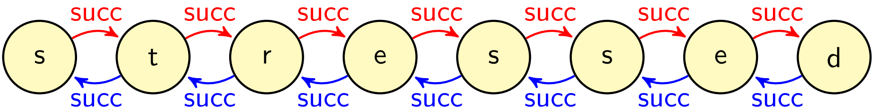
- input string seen as the logical structure over $\{succ, (lab_a)_{a\in\Sigma}\}$
- output predicates defined with MSO formulas interpreted over the input structure



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

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

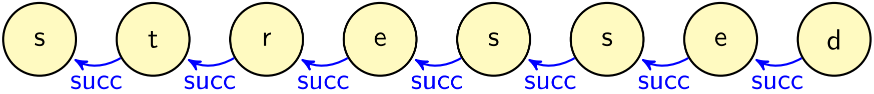


$$\phi_{succ}(x,y) \equiv succ(y,x)$$

$$\phi_{lab_a}(x) \equiv lab_a(x)$$

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



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

$$\phi_{lab_a}(x) \equiv lab_a(x)$$

Streaming String Transducers (Alur, Cerny, 2010)

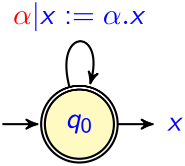
On every transitions, a finite set of variables can be updated by

- appending a string: $x := x.u$
- prepending a string: $x := u.x$
- concatenating two variables: $x := yz$

Streaming String Transducers (Alur, Cerny, 2010)

On every transitions, a finite set of variables can be updated by

- appending a string: $x := x.u$
- prepending a string: $x := u.x$
- concatenating two variables: $x := yz$

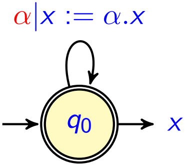


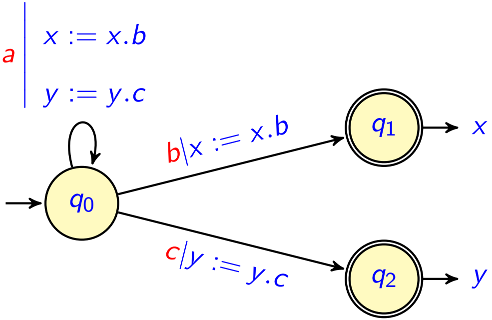
$R(T) = mirror$

Streaming String Transducers (Alur, Cerny, 2010)

On every transitions, a finite set of variables can be updated by

- appending a string: $x := x.u$
- prepending a string: $x := u.x$
- concatenating two variables: $x := yz$





$R(T) = mirror$

$R(T) = a^n\alpha \mapsto \alpha^{n+1}$

Streaming String Transducers

Theorem (Alur Cerny 2010)

The following models are expressively equivalent:

1. *two-way DFT*
2. *MSO transductions*
3. *deterministic (one-way) streaming string transducers with copyless update*

Moreover, SSTs have good algorithmic properties and have been used to analyse list processing programs (Alur Cerny 2011).

A word about infinite strings

- most transducer models can be extended to (right-) infinite strings
- Büchi / Muller accepting conditions
- most of the results seen so far **still hold with some complications ...**

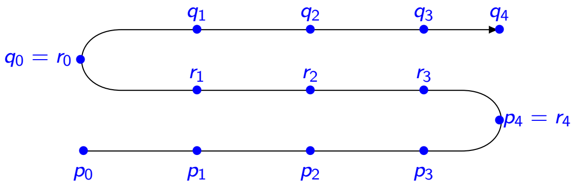
- determinization of one-way transducers: TP is too strong

- deterministic 2way $<$ functional 2way:

$$T \; : \; u \mapsto \begin{cases} a^\omega \text{ if infinite number of 'a'} \\ u \text{ otherwise} \end{cases}$$

- functional 2way $\equiv$ determinitic 2way + ω-regular look-ahead $\equiv$ ω-MSO transductions $\equiv$ ω-SST (Alur,Filiot,Trivedi,12)

z-motions simulation



It is possible to simulate a z-motion run with a one-way automaton

1. each state is a triple (p, r, q)
2. the initial state is (p_0, r_0, q_0) with $q_0 = r_0$
3. $(p_i, r_i, q_i) \xrightarrow{a} (p_{i+1}, r_{i+1}, q_{i+1})$ iff
   - $p_i \xrightarrow{a,+1} p_{i+1}$
   - $r_{i+1} \xrightarrow{a,-1} r_i$
   - $q_i \xrightarrow{a,+1} q_{i+1}$
   - final states: (p, p, q)