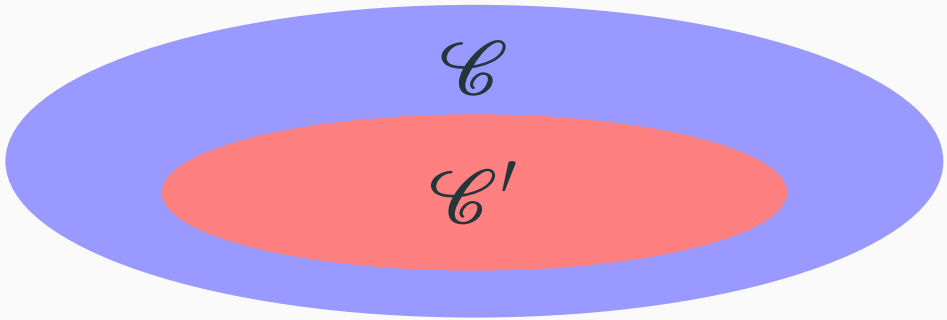
Continous rational functions are deterministic regular

Olivier Carton (IRIF, Université Paris Cité)
& Gaëtan Douéneau-Tabot (IRIF & Direction générale de l'armement)

DeLTA meeting, June 1st, 2022

Class membership problems for functions

Consider two classes of functions $\mathscr{C}$ (computed by a class of machines $\mathscr{M}$) and $\mathscr{C}'$ (computed by a class of machines $\mathscr{M}'$).



- **Characterization:** find a simple property P such that for all $f \in \mathscr{C}$, $P(f) \iff f \in \mathscr{C}'$;
- **Decidability:** given $f \in \mathscr{C}$, can we decide if $f \in \mathscr{C}'$?
- **Synthesis:** given $f \in \mathscr{C}'$ computed by a machine of $\mathscr{M}$, build a machine of $\mathscr{M}'$ which computes it.

Class membership problems for transductions

→ here, the machine models are transducers (= automata with outputs)
the functions are transductions.

Why are such problems interesting & relevant?

- they correspond to program "optimization" or "synthesis",
  which is useful in practice;
- there are more difficult for functions (computed by transducers)
  than for languages (accepted by automata),
  because there are generally no (known) canonical models;
- the proof techniques provide a rich understanding
  of the transducer models.

In this talk

Determine if a rational function over infinite words can be
computed by a deterministic two-way transducer.

Outline

1. Regular and rational functions

2. Deterministic regular functions

3. Rational functions which are continuous

Regular and rational functions

Finite words: sequential and rational functions

Sequential functions:

Functions $A^* \multimap B^*$ computed by one-way deterministic transducers.

Example:

Removing the 0s : $1020301 \mapsto 1231$.

Rational functions:

Functions $A^* \multimap B^*$ computed by (unambiguous) one-way non-deterministic transducers.

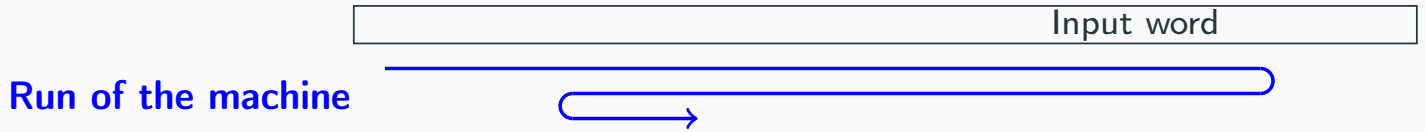
Example:

Removing the 0s if ends by 0 : $1020301 \mapsto 1020301$, $102030 \mapsto 123$.

Finite words: regular functions

Regular functions:

Functions $A^* \to B^*$ computed by deterministic two-way transducers.



Examples:

Reversing the input: $1020301 \mapsto 1030201$.

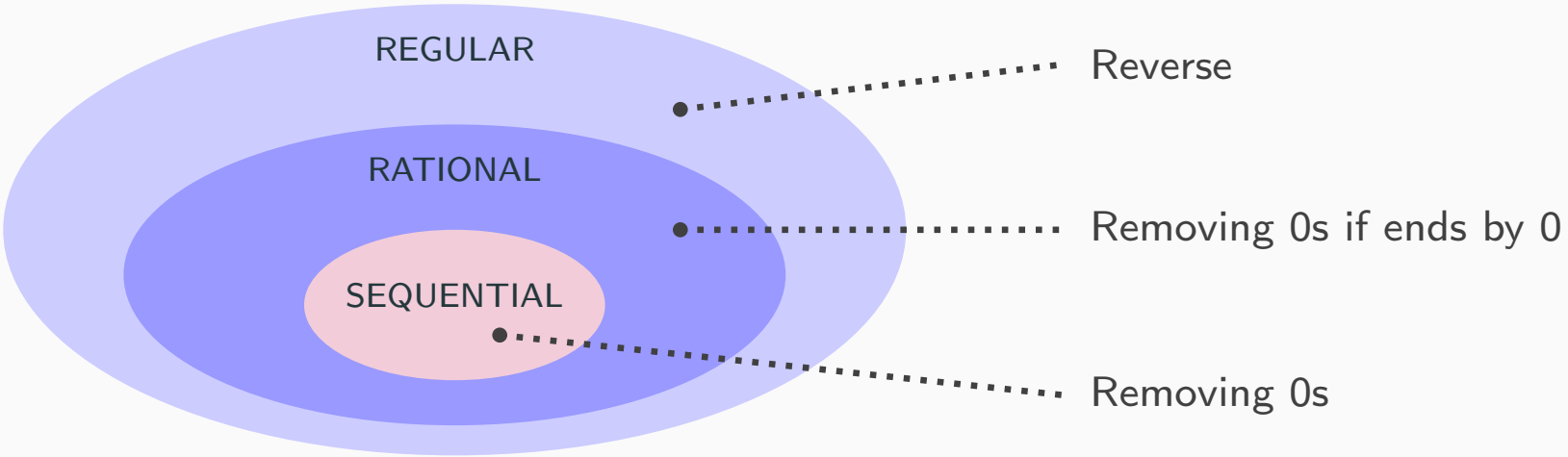
Removing the 0s if ends by 0 : $1020301 \mapsto 1020301$, $102030 \mapsto 123$.

→ Equivalent models:

- MSO-transductions **[Engelfriet and Hoogeboom, 2001]**;
- deterministic streaming string transducers **[Alur and Cerný, 2010]**.

→ Any rational function is regular (lookahead removal).

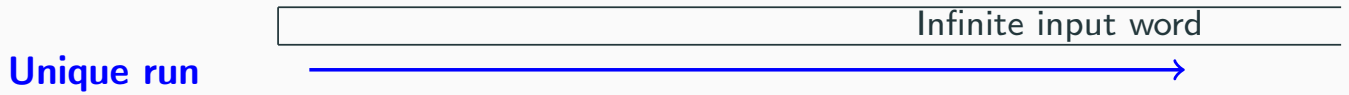
Finite words: summary



Infinite words: sequential functions

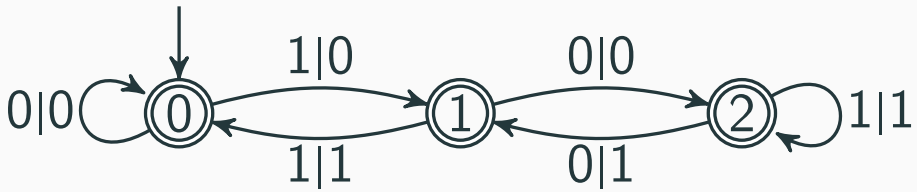
Sequential functions:

Functions $A^\omega \to B^\omega$ computed by one-way deterministic transducers.



Example:

Division by 3 in base 2.

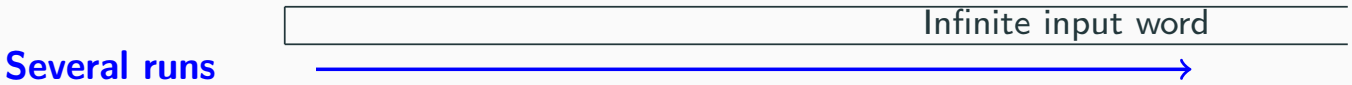


→ A sequential function is computable by a deterministic Turing machine.

Infinite words: rational functions

Rational functions:

Functions $A^\omega \to B^\omega$ computed by (unambiguous) one-way non-deterministic transducers.



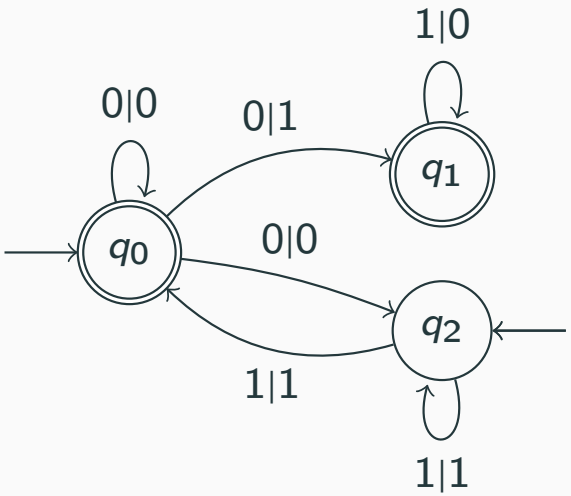
A run is accepting iff it visits infinitely often an accepting state (Büchi condition).

→ may not be computable by a deterministic Turing machine (that's a problem for applications to streaming algorithms...).

Infinite words: rational functions

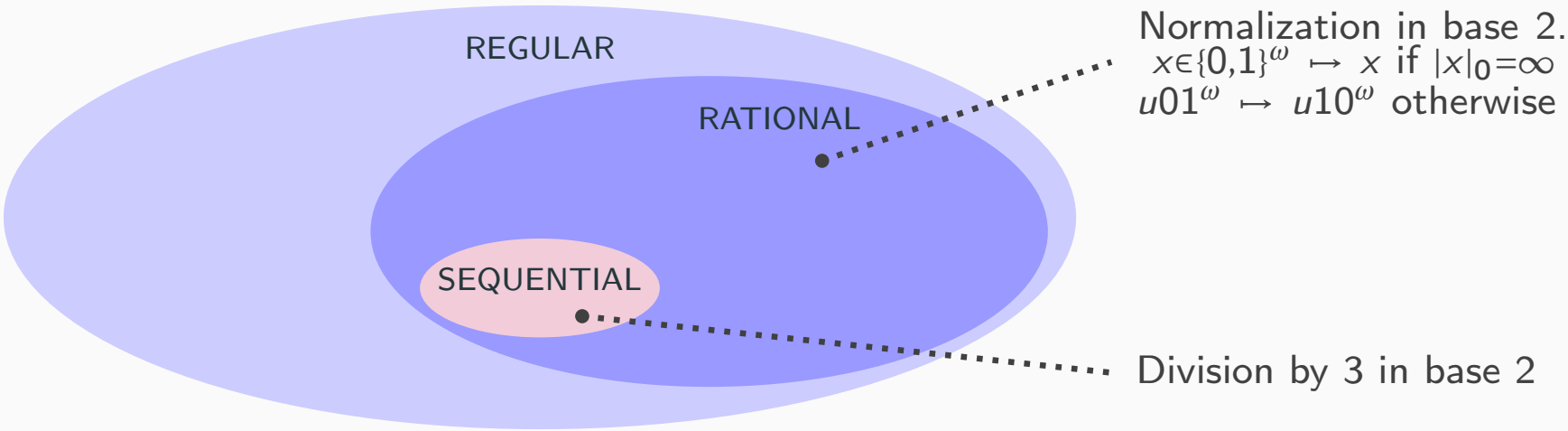
Example:

normalize : $\{0,1\}^\omega \rightharpoonup \{0,1\}^\omega$ with $\mathrm{Dom}(\text{normalize}) = \{0,1\}^\omega$
defined by $x \mapsto x$ if $|x|_0 = \infty$ and $u01^\omega \mapsto u10^\omega$ for $u \in \{0,1\}^*$.



→ Not computable by a deterministic Turing machine
(need to guess if $|x|_0 = \infty$ or not).

Infinite words: summary



→ Regular functions are defined equivalently by two-way deterministic transducers with ω-regular lookaheads, or streaming string transducers with Müller conditions, or MSO-transductions 🕮 **[Alur et al., 2012]**.

→ Some functions are not computable by deterministic Turing machines.

Continuity vs computability

→ When is a regular/rational function computable?

Theorem **[Dave et al., 2020]**

A regular function $f : A^\omega \rightharpoonup B^\omega$ is continuous if and only if it can be extended to a computable function (by a Turing machine).

Continuity

$f : A^\omega \rightharpoonup B^\omega$ is continuous if for all $x \in \mathrm{Dom}(f)$ and $n \geqslant 0$, there exists $p \geqslant 0$ such that $\forall y \in \mathrm{Dom}(f)$, $|x \wedge y| \geqslant p \Rightarrow |f(x) \wedge f(y)| \geqslant n$

where $\wedge$ is the longest common prefix.

→ Continuity is decidable for rational functions **[Prieur, 2001]** and even regular functions **[Dave et al., 2020]**.

Continuity of rational functions

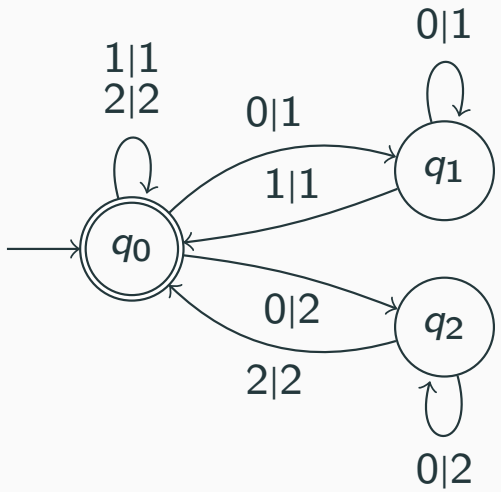
Examples:

- normalize : $\{0,1\}^\omega \rightharpoonup \{0,1\}^\omega$
  defined by $x \mapsto x$ if $|x|_0 = \infty$ and $u01^\omega \mapsto u10^\omega$ otherwise

  is not continuous in 01^ω;

- replace : $\{0,1,2\}^\omega \rightharpoonup \{1,2\}^\omega$ with
  $\mathrm{Dom}(\mathrm{replace}) = \{x : |x|_1 = \infty \text{ or } |x|_2 = \infty\}$ and
  $0^{n_1} a_1 0^{n_2} a_2 \cdots \mapsto {a_1}^{n_1+1} {a_2}^{n_2+1} \cdots$ if $a_i \in \{1,2\}$, $n_i \in \mathbb{N}$

  is continuous.



Open problem of Dave, Filiot, Krishna & Lhote

Given a continuous rational/regular function can we do better than a Turing machine? A more efficient model?

In particular, can we compute it with a deterministic two-way transducer?

→ In this talk: for continuous rational functions, **yes we can**.

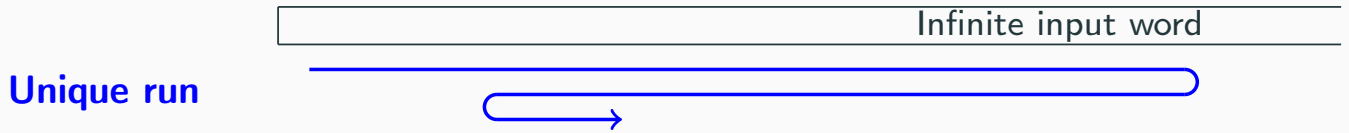
(seems obvious, but far from being so...).

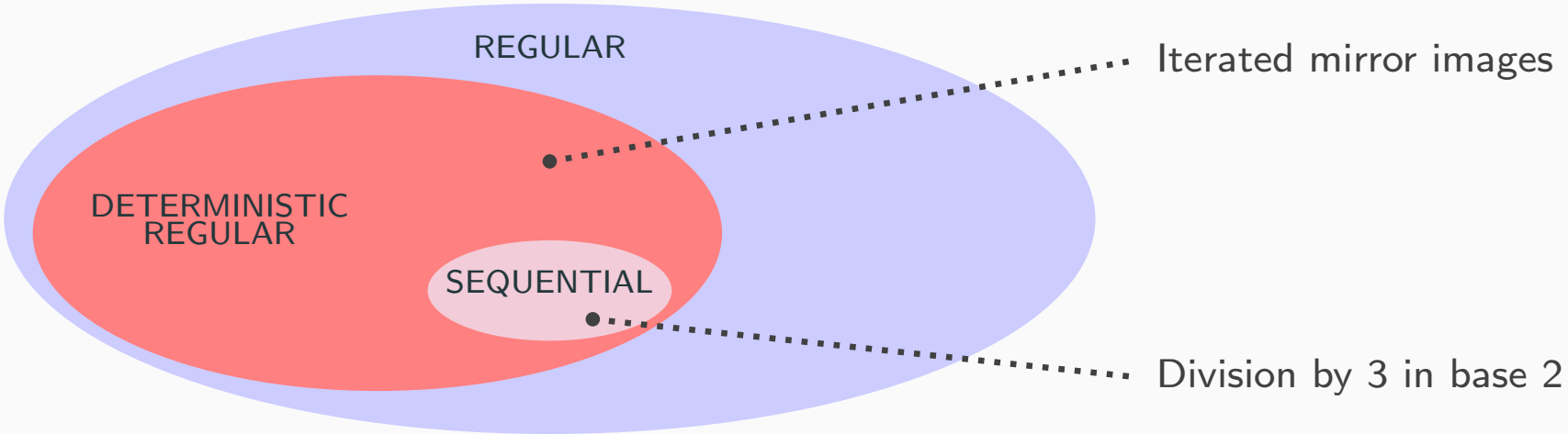
Deterministic regular functions

Deterministic regular functions

Deterministic regular functions:

Functions $A^\omega \to B^\omega$ computed by deterministic two-way transducers.



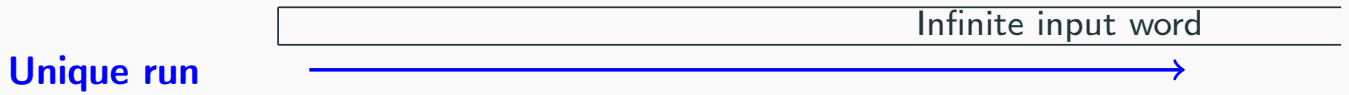


Streaming string transducers

Deterministic streaming string transducers (dSST**):**

One-way deterministic automaton + registers and updates.

Specific register $\mathfrak{out}$ which stores the output.



→ Copyless if a register value is never duplicated.

Example:

Iterated mirror images: $u_1\#u_2\#u_3\#\cdots \mapsto \widetilde{u_1}\#\widetilde{u_2}\#\widetilde{u_3}\#\cdots$

- when reading u_i, a register $\mathfrak{r}$ storing $\widetilde{u_i}$
- when reading $\#$, $\mathfrak{out} \leftarrow \mathfrak{out}\ \mathfrak{r}\ \#$ and $\mathfrak{r} \leftarrow \varepsilon$.

Streaming string transducers

Theorem

A function $f : A^\omega \to B^\omega$ is deterministic regular if and only if it can be computed by a copyless dSST. The conversions are effective.

→ Also works with "bounded copy".

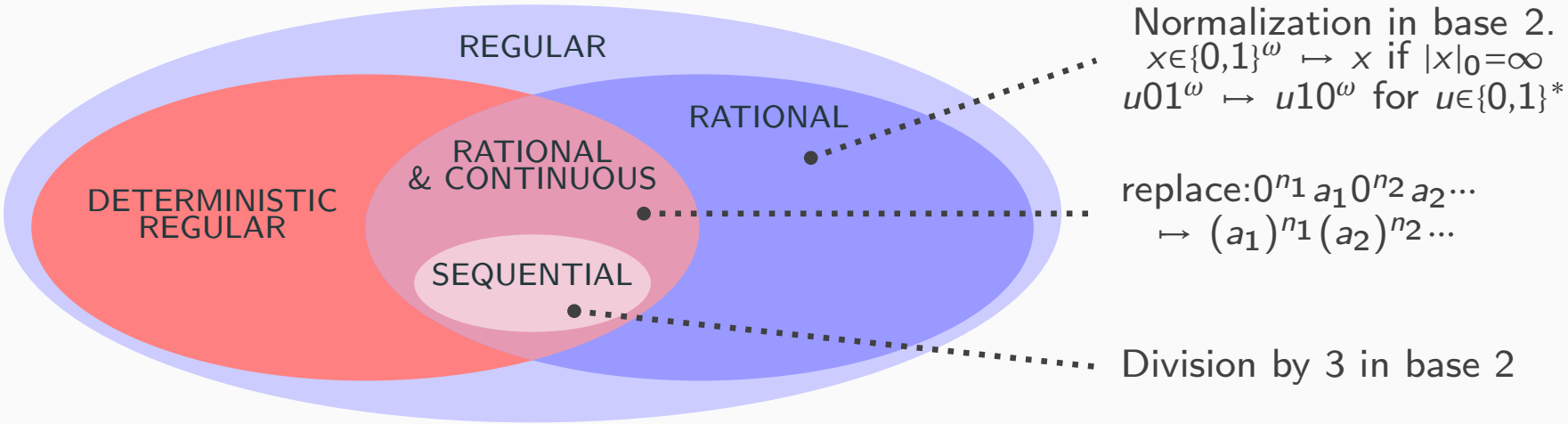
→ Quite similar to the results for regular functions over finite words 🗞 **[Alur and Cerný, 2010]** or infinite words 🗞 **[Alur et al., 2012]**
but here, lookaheads cannot be used!

Rational functions which are continuous

Continuous rational functions are deterministic regular

Theorem

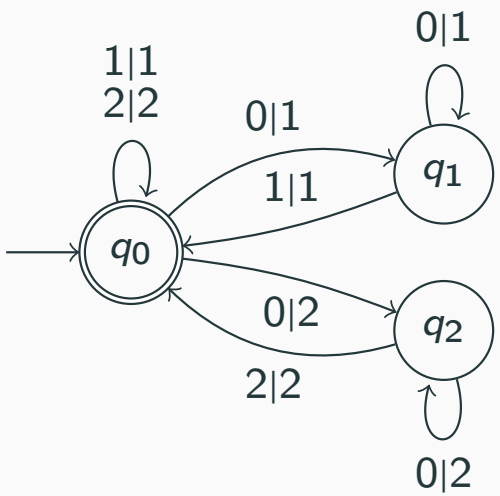
A rational function $f : A^\omega \rightharpoonup B^\omega$ is continuous if and only if it can be extended to a deterministic regular function.
The construction is effective.



Proof idea: finite non-determinism (easy case)

Example:

replace : $\{0,1,2\}^\omega \rightharpoonup \{1,2\}^\omega$ with $0^{n_1} a_1 0^{n_2} a_2 \cdots \mapsto {a_1}^{n_1+1} {a_2}^{n_2+1} \ldots$



When reading a block 0^{n_i}:

- either it ends with a 1 and we must output 1^{n_i};
- or it ends with a 2 and we must output 2^{n_i};
- or it never ends (0^ω) but then not in Dom(replace).

→ Choices can be checked after looking at a finite number of letters (if not, we are not in the domain).

Proof idea: finite non-determinism (easy case)

How to compute replace with a dSST?

- when reading 0^{n_i}, store 1^{n_i} in $\mathfrak{r}_1$ and 2^{n_i} in $\mathfrak{r}_2$;
- when reading 1, update $\mathfrak{out} \leftarrow \mathfrak{out}\ \mathfrak{r}_1\ 1$; and $\mathfrak{r}_1 \leftarrow \varepsilon,\ \mathfrak{r}_2 \leftarrow \varepsilon$
- when reading 2, update $\mathfrak{out} \leftarrow \mathfrak{out}\ \mathfrak{r}_2\ 2$ and $\mathfrak{r}_1 \leftarrow \varepsilon,\ \mathfrak{r}_2 \leftarrow \varepsilon$.

$\rightarrow$ simulate in parallel all choices, output the correct one when checked.

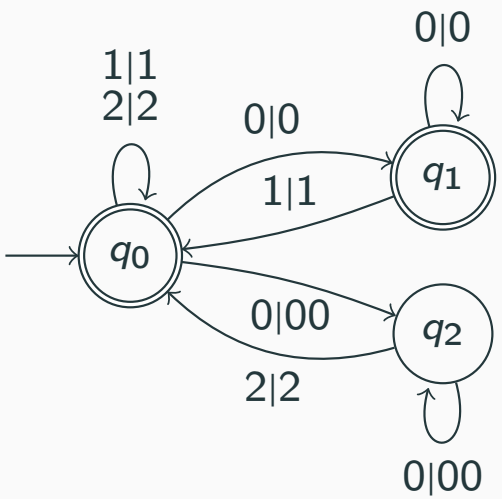
$\rightarrow$ we do not use the full power of a dSST/two-way transducer, but only a one-way deterministic transducer with "finite lookaheads".

Proof idea: infinite non-determinism (tricky case)

Example:

double : $\{0,1,2\}^\omega \to \{0,1,2\}^\omega$ mapping:

- $0^{n_1} a_1 0^{n_2} a_2 \cdots \mapsto 0^{a_1 n_1} a_1 0^{a_2 n_2} a_2 \cdots$;
- $0^{n_1} a_1 \cdots 0^{n_m} a_m 0^\omega \mapsto 0^{a_1 n_1} a_1 \cdots 0^{a_m n_m} a_m 0^\omega$ (if finitely many 1 or 2).



When reading a block 0^{n_i}:

- either it ends with a 1 and we must output 0^{n_i};
- or it ends with a 2 and we must output 0^{2n_i};
- or it never ends (0^ω) and we must output 0^ω.

Proof idea: infinite non-determinism (tricky case)

→ When reading 0^ω, two runs live in parallel and one is accepting.
→ Combinatorial property: these runs produce iterations of a word.

Two constraints on the construction:

- we must output something infinitely often when reading 0^{n_i};
- we must be able at each time to "rebuild" 0^{n_i} and 0^{2n_i}.

How to compute double with a dSST?

- when reading 0^{n_i}, output 0^{n_i} and store 0^{n_i} in a register $\mathfrak{r}$;
- when reading 1, update $\mathfrak{out} \leftarrow \mathfrak{out}\ 1$ and $\mathfrak{r} \leftarrow \varepsilon$;
- when reading 2, update $\mathfrak{out} \leftarrow \mathfrak{out}\ \mathfrak{r}\ 2$ and $\mathfrak{r} \leftarrow \varepsilon$.

Outlook

Outlook

New tools developed in this paper:

- the new class of deterministic regular functions;
- combinatorial properties of the runs which have "infinite non-determinism" in a one-way non-deterministic transducer computing a continuous function;
- use of the registers of a dSST to simulate such runs (complex algorithm relying on the tree of compatibles).

Outlook: work in progress

Almost proven conjecture:

A rational function is uniformly continous if and only if it can be extended to a function computable by a copyless dSST without ε-loops in $\mathfrak{out}$.

Conjecture:

A regular function is continous if and only if it can be extended to a deterministic regular function.