Some Annoying Transducers

Klaus Sutner

Carnegie Mellon University

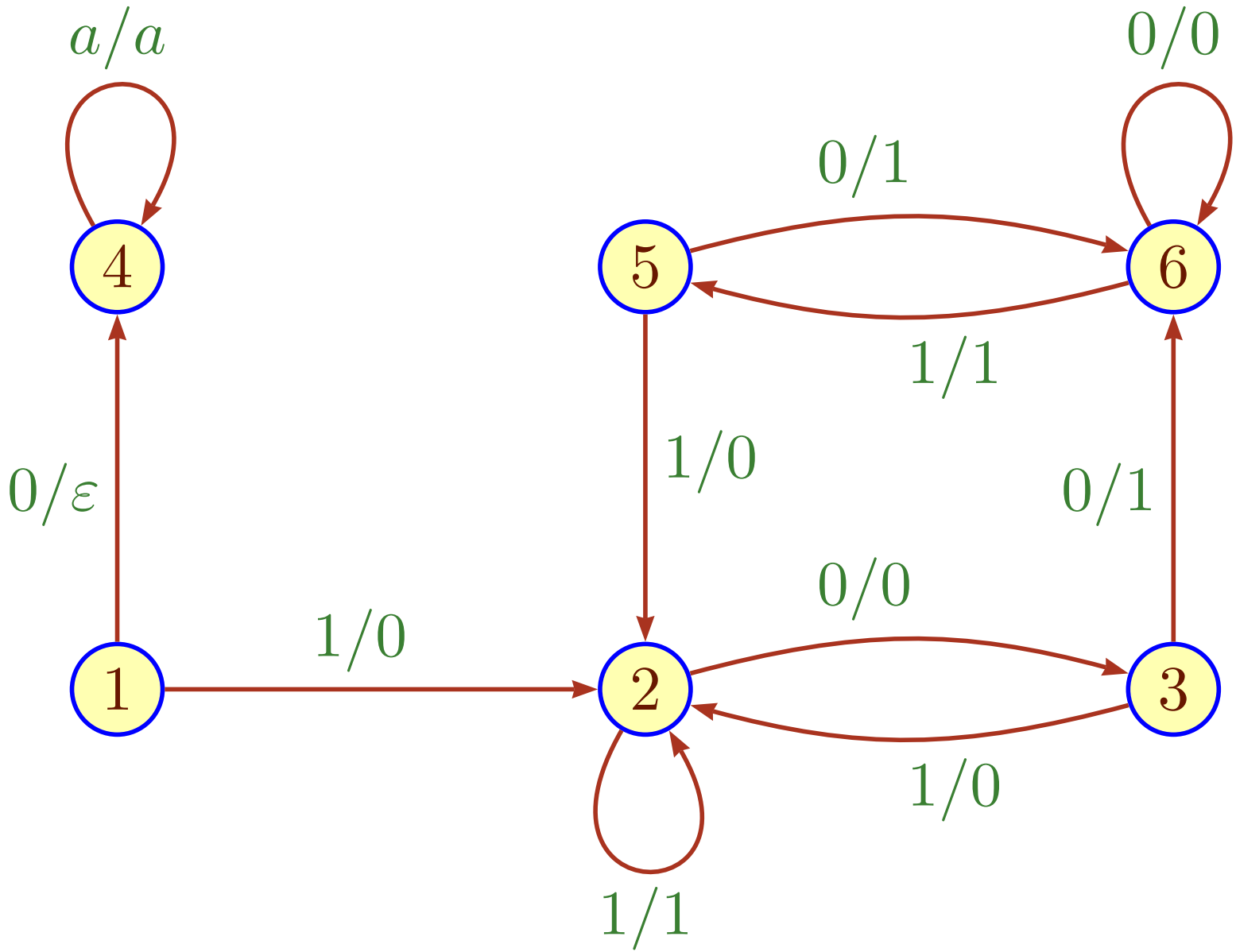
A transducer is a finite state machine that generates output: the transitions are labeled by two letters:

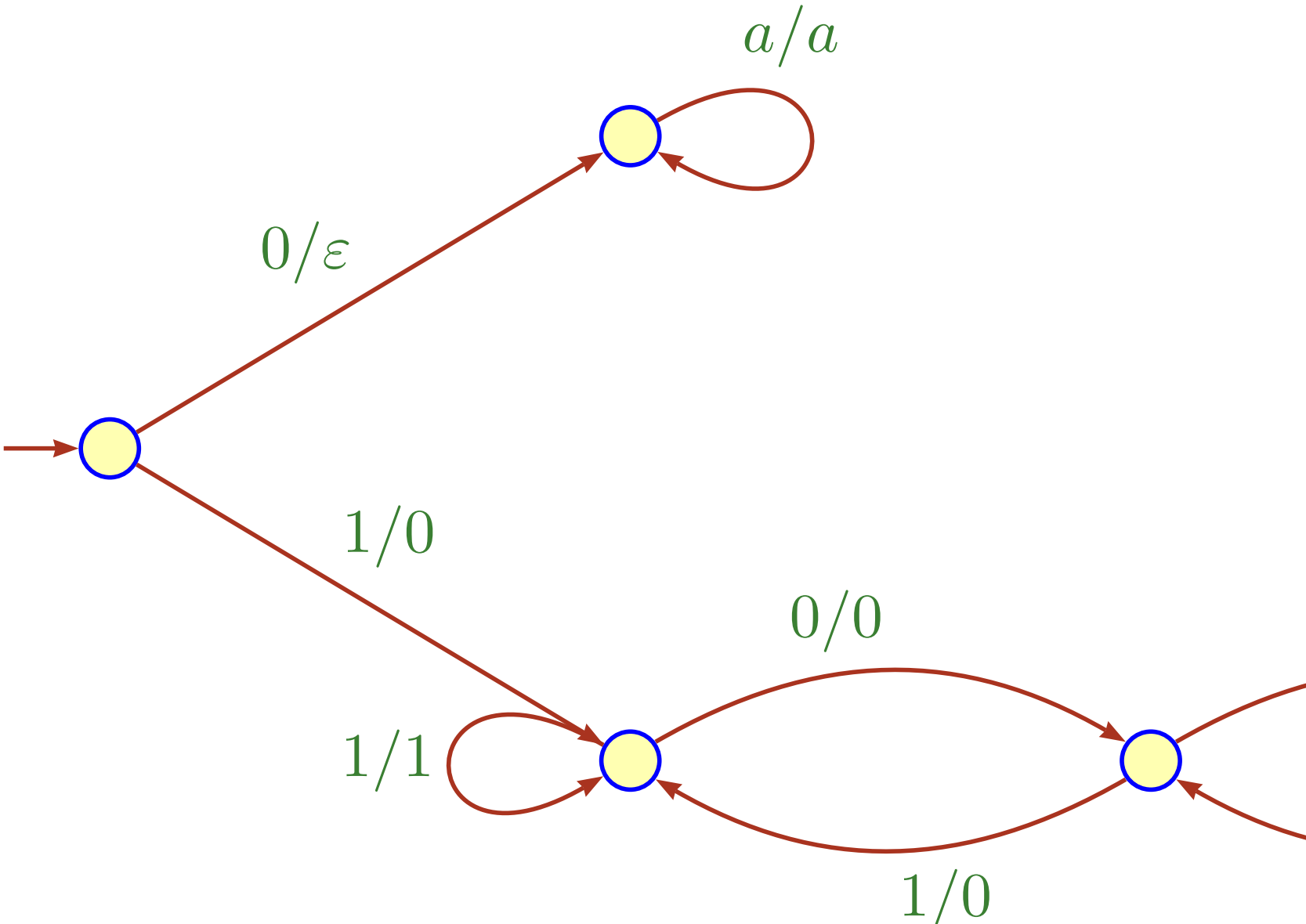
$$p \xrightarrow{a/b} q$$

Interpretation: in state p, reading letter a, the device moves to state q and outputs b.

It is often convenient to allow whole words for b.

We only consider input deterministic transducers: if we ignore the output labels, the resulting machine is deterministic (otherwise implementation becomes tricky).


a/a
0/0
0/1
4
5
6
1/1
1/0
0/ε
0/1
0/0
1/0
1
2
3
1/0
1/1


a/a
$0/\varepsilon$
$1/0$
$0/0$
$1/1$
$1/0$

Input and output alphabet is always $\mathbf{2} = \{0, 1\}$.

We are interested in the special case where the transitions out of state p are of the form

$$p \xrightarrow{0/0} q \qquad p \xrightarrow{1/1} q'$$

or

$$p \xrightarrow{0/1} q \qquad p \xrightarrow{1/0} q'$$

We call p an identity or flip state, respectively.

Definition

A IODA has the form $T = \langle Q, \tau, q_0 \rangle$ where Q is a finite set of states, q_0 the initial state and $\tau \subseteq Q \times \mathbf{2} \times \mathbf{2} \times Q$ a set of transitions as above.

The transducer defines a transduction (also denoted by τ)

$$\tau : \mathbf{2}^\star \to \mathbf{2}^\star$$

in the obvious way (start at q_0, follow transitions according to input labels, return output labels).

We write $\tau^\star$ for the iterated map

$$\tau^\star : \mathbf{2}^\star \to \mathfrak{P}(\mathbf{2}^\star)$$

Note that $\tau^\star(x)$ is always finite (cardinality at most $2^{|x|}$).

Preserving Length Only

Length-preserving transducers are much more powerful than IODAs. E.g., an iterated length-preserving transducer can reverse a string.

The Reachability Problem (can v be obtained from u by repeated application of the transduction) here is trivial in a sense: we only need to check finitely many steps.

Theorem (Latteux, Simplot, Terlutte)

Let τ be a length-preserving transduction and L a regular language. It is undecidable whether $\tau^\star(L)$ is regular.

Of course, Reachability in the general case is undecidable: we can simulate computationally universal devices such as Turing machines with a general transducer.

What is surprising, though, is that even fairly simple transducers lead to very messy problems when iterated.

For any IODA, the output function τ has special properties:

- τ is trivially length-preserving
- τ is injective

In fact, getting the inverse transducer is simple: switch

$$p \xrightarrow{0/1} q \qquad p \xrightarrow{1/0} q'$$

to

$$p \xrightarrow{0/1} q' \qquad p \xrightarrow{1/0} q$$

At any rate, τ is just a permutation of $\mathbf{2}^\star$ and the components of its diagram are all cycles, each contained in $\mathbf{2}^k$ for some k.

- How many cycles are there?
- How long are these cycles?
- How hard is it to test membership in a cycle?
- How hard is it to compute the least element?

Note well: We want computational answers.

The components involve only configurations of the same length k, so k is the key parameter.

Suppose

$$u_0, u_1, \ldots, u_{n-1}$$

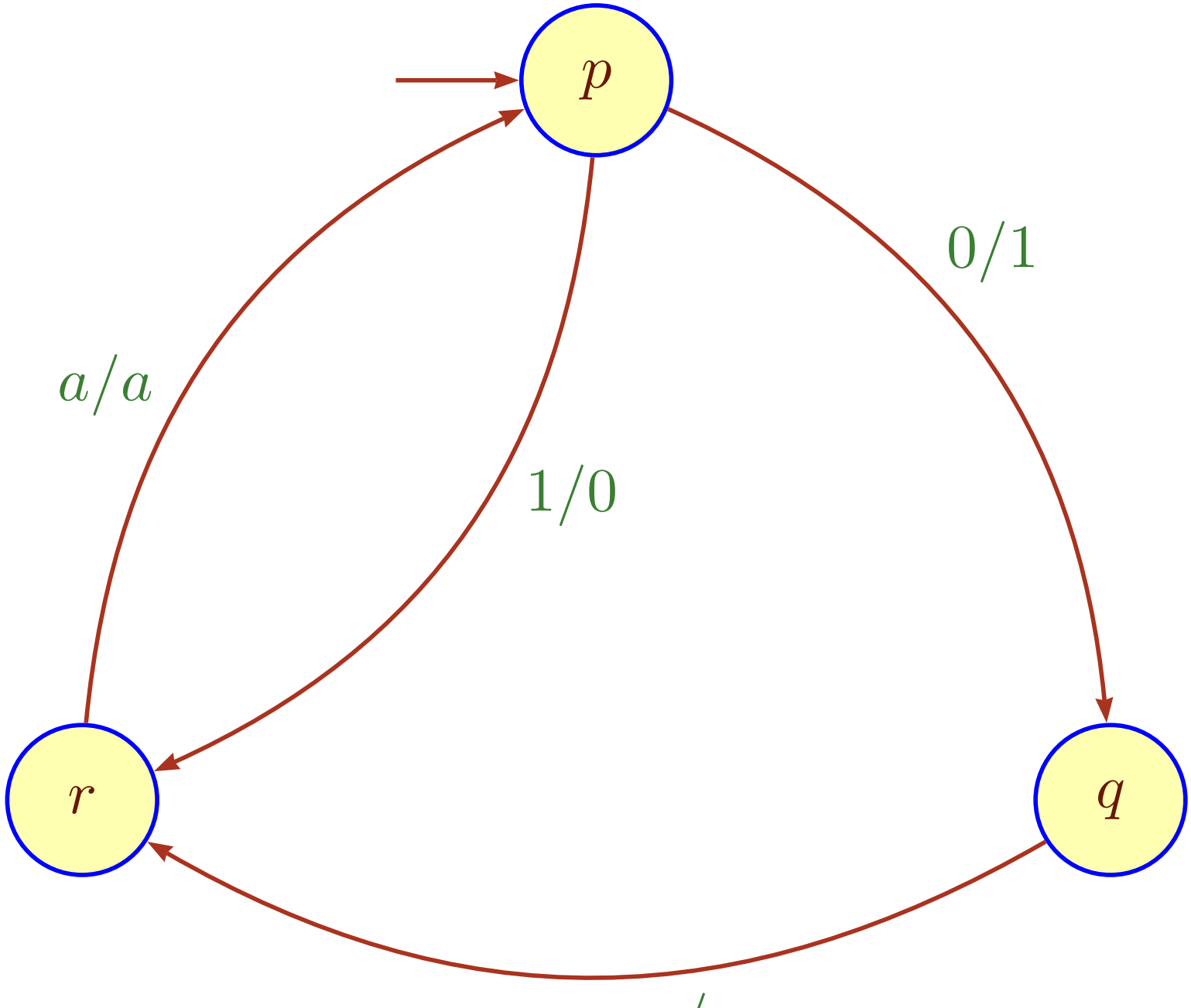
is a cycle. Then either there are two cycles

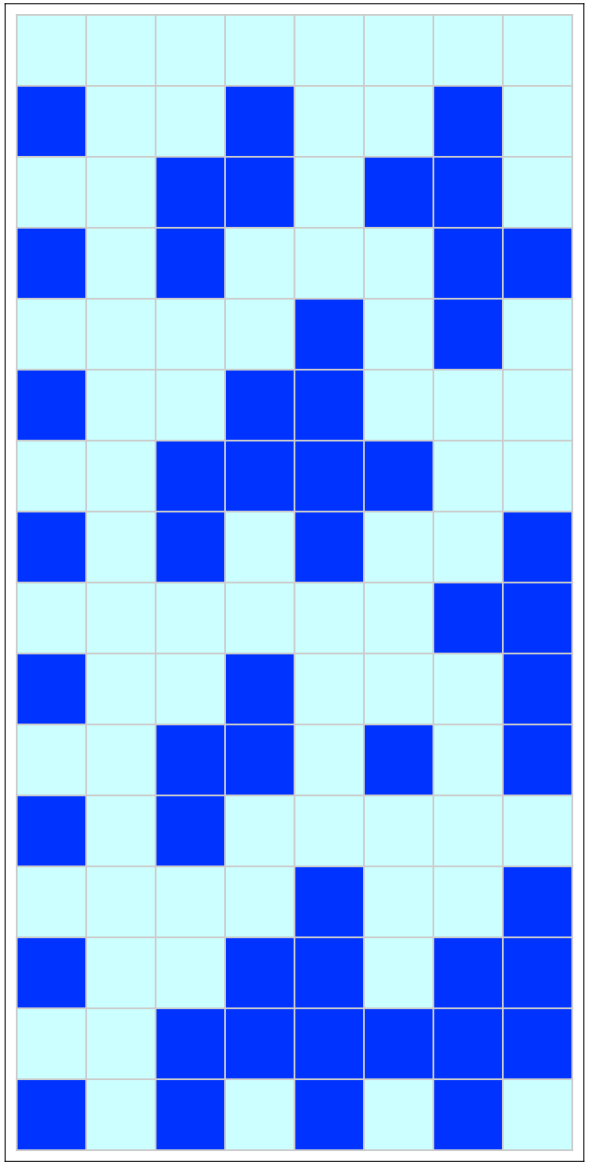
$$u_0 0, u_1 b_1, \ldots, u_{n-1} b_{n-1} \qquad u_0 1, u_1 \overline{b_1}, \ldots, u_{n-1} \overline{b_{n-1}}$$

or there is a single cycle

$$u_0 0, u_1 b_1, \ldots, u_{n-1} b_{n-1}, u_0 1, u_1 \overline{b_1}, \ldots, u_{n-1} \overline{b_{n-1}}.$$

Hence we can organize the cycles into a binary tree.


p
q
r
0/1
1/0
a/a



The root corresponds to the fixed point ε.

For any orbit, define its root to be the lexicographically least word on the orbit.
For a word u define root(u) to be the root of the orbit of u.

Lemma

There are $2^{\lfloor k/2 \rfloor}$ cycles on words of length k.
The length of each cycle is $2^{\lceil k/2 \rceil}$.

In fact, given the regularity of the cycle tree it is easy to show that the number of cycles of length 2^y on words of length x is given by the generating function

$$\frac{xy+1}{1-2x^2y}$$

Done, up to Isomorphism

Up to isomorphism the generating function describes the diagram of #3820 completely.

But how about the actual points? How about computational answers?

Specifically: How hard is it to test if u and v lie on the same cycle?

An obvious upper bound for words of length $2k$ is

$$O(k\,2^k)$$

Can we do better than this?

Suppose u has length $2k$, write elements in the cycle of u as the shuffle of two words a and b of length k:

$$v = a \parallel b = a_1 b_1 a_2 b_3 \ldots a_k b_k$$

- For each orbit, all possible a-words occur exactly once. Hence, for any $a \in \mathbf{2}^k$, there is a unique b such that $a \parallel b$ lies on the orbit.

- The lexicographically smallest element on the orbit has $a = \mathbf{0}$.

A Crazy Permutation

Again, fix some orbit in $\mathbf{2}^{2k}$.

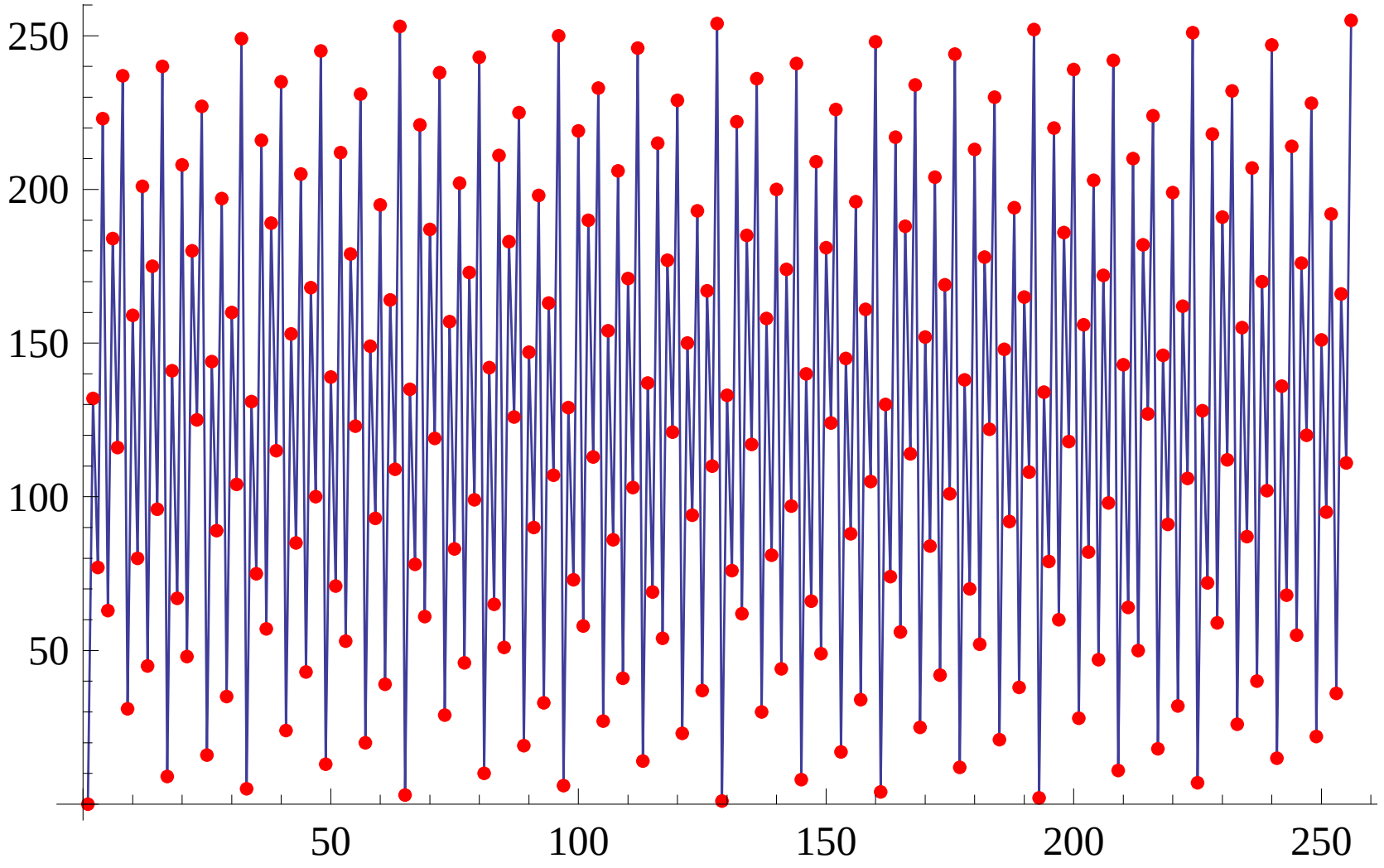
Then for each $a \in \mathbf{2}^k$ there is a unique $b \in \mathbf{2}^k$ such that $a \parallel b$ lies on the orbit.

If we interpret a and b as binary expansions of some integer we get a map

$$\{0, 1, \ldots, 2^k - 1\} \longrightarrow \{0, 1, \ldots, 2^k - 1\}$$

What would this map look like?

Bouncing Around



Fix an orbit and some anchor point u of length k.

Lemma

The t^{th} point on the cycle of a such that u can be computed in time $O(k^3)$.

Lemma

Given orbit element v and some a, the unique x such that $v = a \parallel x$ can be computed in time $O(k^3)$.

In particular the root can be computed in time $O(k^3)$.

The claims above are all "obvious" from a bit of experimental computation.

But the proofs are annoyingly difficult. E.g., the following counting function seems to play a role in the argument:

$$\frac{1}{5}\left(2^{k/2}\left(1+\sqrt{2}-(-1)^k\left(-1+\sqrt{2}\right)\right)+ i^k\left(3\cos\left(k\pi/4\right)+\sin\left(k\pi/4\right)\right)\left(\cos\left(k\pi/2\right)+i\sqrt{2}\sin\left(k\pi/2\right)\right)\right)$$

As we have seen, finite state machines can be used to decide certain simple relations such as lexicographic order.

How about Reachability for IODAs?

As usual, we exploit the convolution of words to produce input for our machines (padding is not necessary here):

$x{:}y =$

x_1	x_2	x_3	$\ldots$	x_{k-1}	x_k
y_1	y_2	y_3	$\ldots$	y_{k-1}	y_k

We can think of a IODA T as a recognizer over $\mathbf{2} \times \mathbf{2}$. Hence the following language is regular:

$$\mathcal{L}(T) = \{\, x{:}y \mid y = \tau(x) \,\}$$

But how about

$$\mathcal{L}(T^\star) = \{\, x{:}y \mid x, y \text{ on same cycle} \,\}$$

The regularity of the orbit relation is closely connected to the root function.

Lemma

The orbit relation is regular if, and only if, the root function can be computed by a transducer.

The root language root$(\mathbf{2}^\star)$ is connected to the cycle tree.

Lemma

The cycle tree is regular if, and only if, the root language is regular.

If the root function is a transduction then the root language is regular, so we have:

Theorem

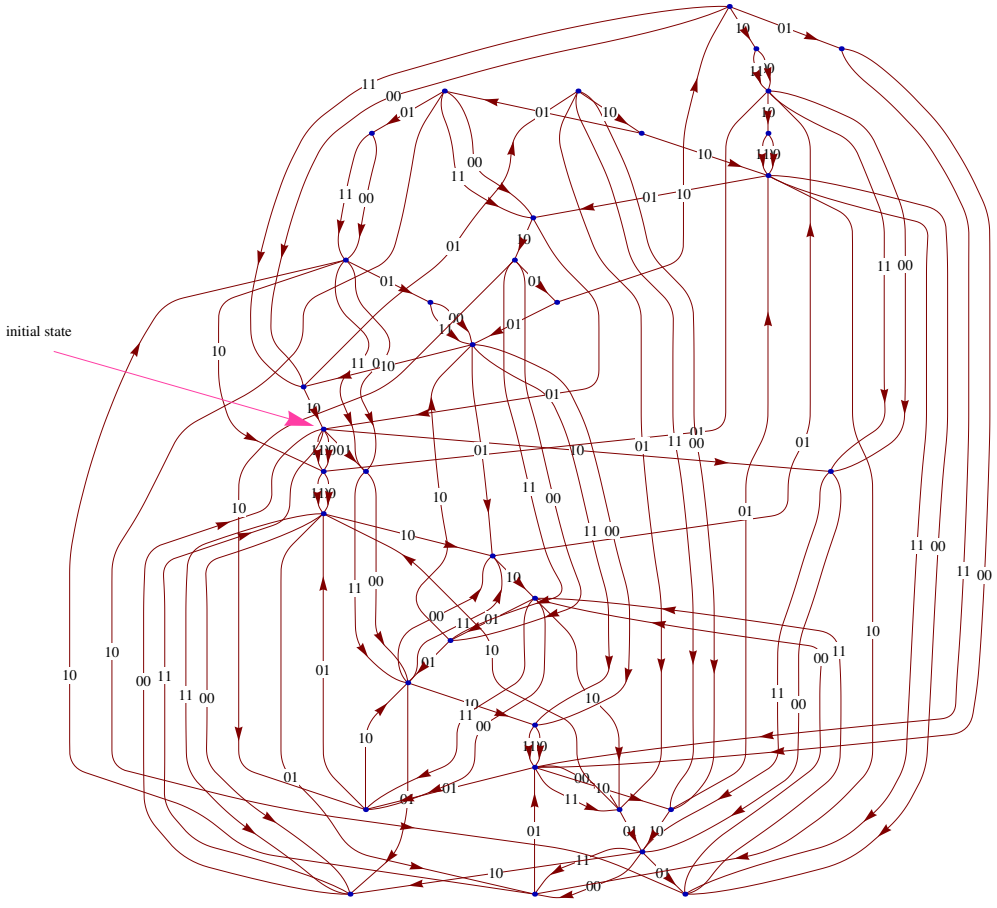
If an IODA transducer has a regular orbit language then its cycle tree is also regular.

Conjecture (as of 6 months ago)

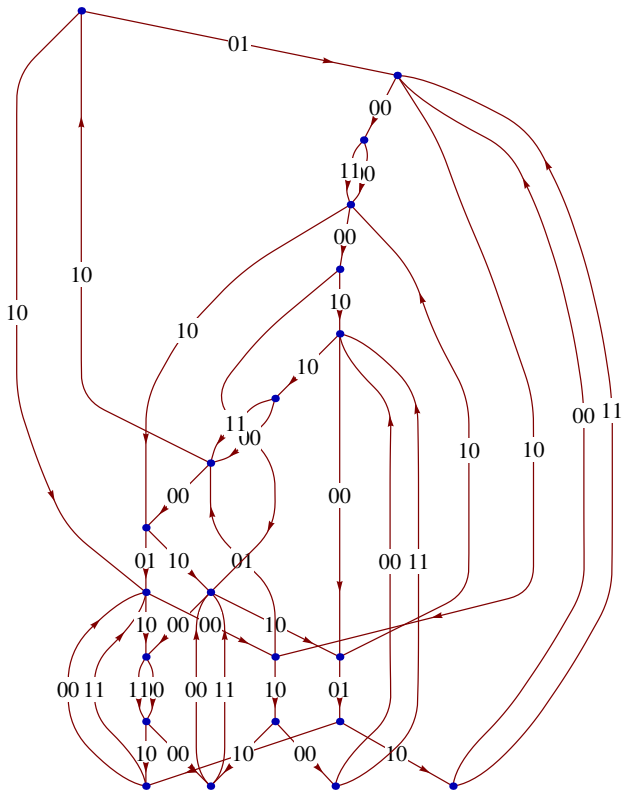
The opposite implication is false.
Specifically, #3820 has regular cycle tree but the orbit relation is irregular.

Alas . . .

As it turns out, #3820 has regular orbit relation. The minimal recognizer has 35 states:



Here is the corresponding root transducer:



But the root language is simply $(0\mathbf{2})^\star(0+\varepsilon)$.

How can a 3-state transducer produce a 35-state recognizer for the orbit relation?

The 35-state recognizer was found by trying to approximate the allegedly infinite recognizer for $\mathcal{L}(T^\star)$: the longer the words, the bigger the diagram should be.

True, but it turned out to be periodic after a while. Rewiring the periodic part and minimizing produces the 35-state machine.

We can associate three transductions P, Q and R with T_{3820} by moving the initial state around.

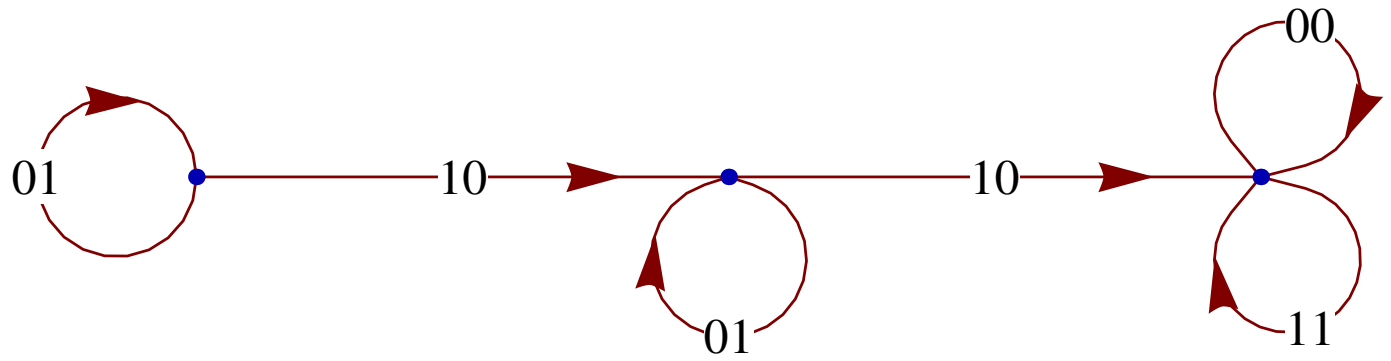
Claim

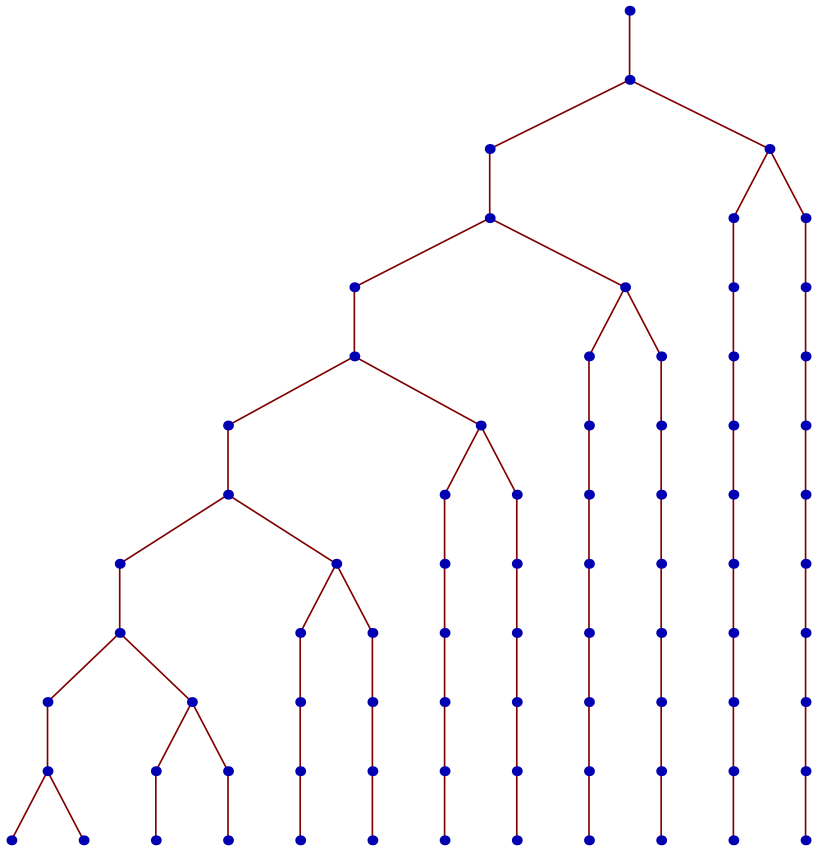
The functions P, Q and R commute. Moreover $P^2QR^2 = I$.

As a consequence, the monoid generated by P, Q and R is a commutative group.

This is enough to show that the number of quotients of $\mathcal{L}(T^\star)$ is finite.

Here is an example of another transducer that has regular orbit relation and hence a regular cycle tree.

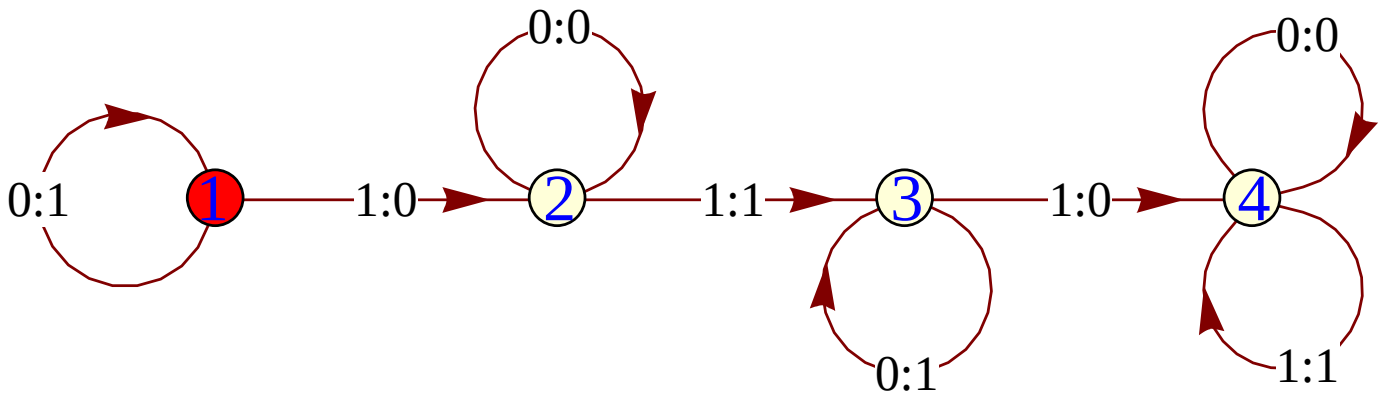


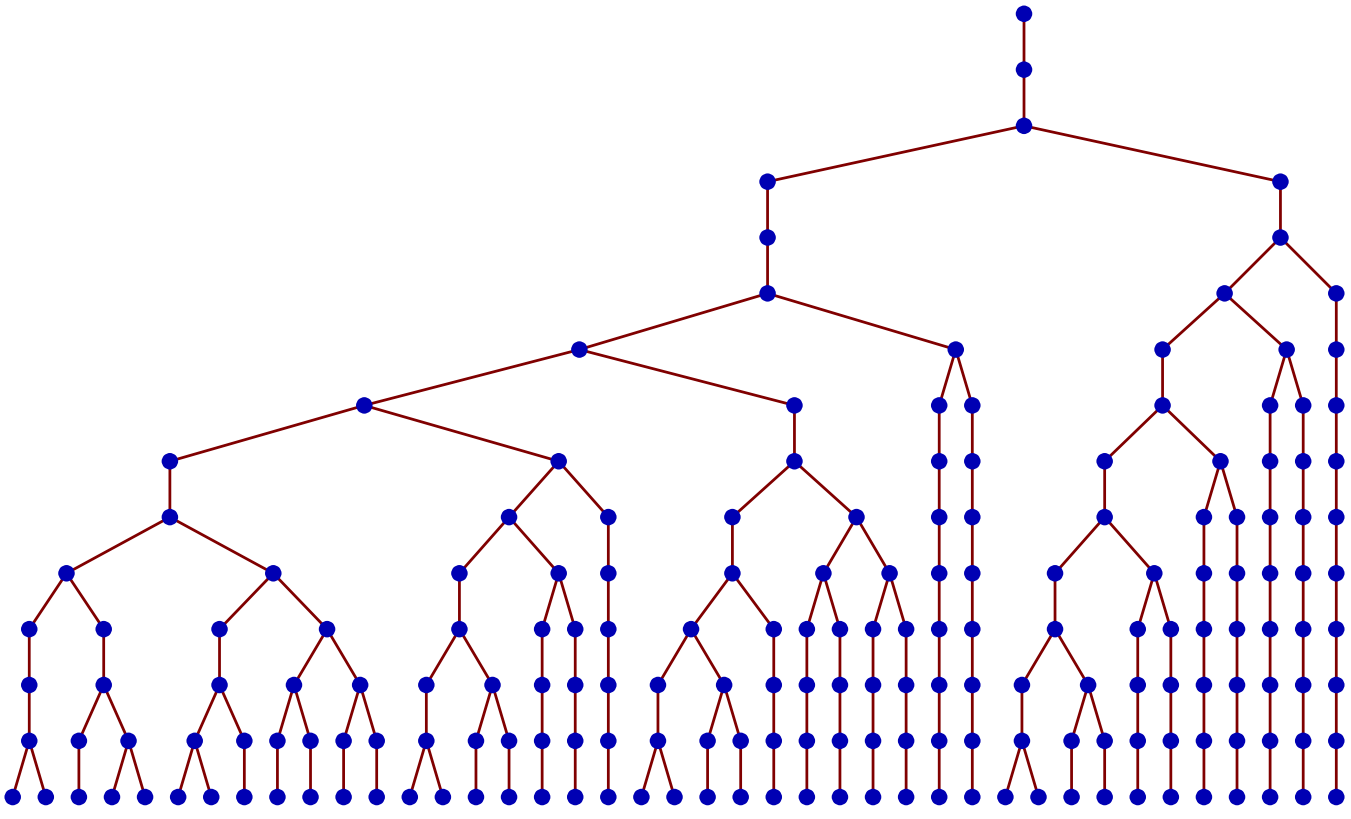


	0	1	2	3	4	5	6	7	8
1	0	1							
2	0	2							
3	0	2	1						
4	0	0	4						
5	0	0	2	3					
6	0	0	0	4	2				
7	0	0	0	2	3	2			
8	0	0	0	0	4	2	2		
9	0	0	0	0	2	3	2	2	
10	0	0	0	0	0	4	2	2	2

Example Irregular and Irregular

An example of a transducer that has an irregular cycle tree and hence an irregular orbit relation.





	0	1	2	3	4	5	6	7	8	9	10
1	0	1									
2	0	0	1								
3	0	0	2								
4	0	0	0	2							
5	0	0	0	2	1						
6	0	0	0	2	3						
7	0	0	0	2	5	1					
8	0	0	0	0	6	3	1				
9	0	0	0	0	4	6	2	1			
10	0	0	0	0	2	9	3	2	1		
11	0	0	0	0	2	11	6	2	2	1	
12	0	0	0	0	0	6	13	4	2	2	1

Does regular cycle tree imply regular orbit relation?

Is it decidable whether the cycle tree (orbit relation) is regular?

Is there any hope to get some insights into the Collatz problem via transducers?