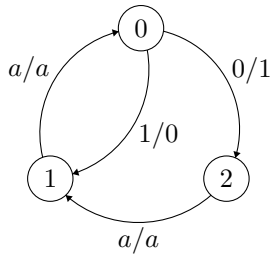
Notes on Transducer Orbit Checking

Evan Bergeron

March 27, 2016

A Simple Invertible Transducer

This is A_2^3.



Let $\underline{i}$ be the function induced by applying A_2^3 to some string with state i as the start state. $\underline{0}, \underline{1}, \underline{2}$ generate a semigroup under composition. For example, $\underline{0}^2$ is the result of applying $\underline{0}$ twice. We call such functions *transductions*.

If x is some string, call $\underline{0}^* = \{\underline{0}^t(x) \mid t \in \mathbb{N}\}$ the *orbit* of x under $\underline{0}$.

We prove that checking if $y \in \underline{0}^*(x)$ is in P. We then investigate orbit checking for arbitrary f in the semigroup.

A Necassary Condition

Let $f = \underline{0}$. Let $y \in f^*(x)$. Let c be a sequence where c_i is the number of times x_i is flipped on the way to y. Then we claim that

$$\begin{aligned} c_i = \lfloor (c_{i-3}/2) \rfloor + (c_{i-3} \mod 2) \cdot (1 - x_{i-3}) \\ + \lfloor (c_{i-2}/2) \rfloor + (c_{i-2} \mod 2) \cdot x_{i-2} \end{aligned}$$

where c_0 is the index of y in $f^*(x)$, c_1 is 0, and c_2 is $\lfloor c_0/2 \rfloor$.

x_0 is flipped upon every invocation of f. x_1 is never flipped. x_2 is flipped roughly every other time x_0 is flipped. That is, it's flipped every time x_0 changes from a 1 to a

0. Without loss, we may assume $x_0 = 0$. Thus, $c_2 = \lfloor c_0/2 \rfloor$ (it's not flipped the first time, but then is flipped every other time afterward).

Consider some application of f. c_i is flipped iff c_{i-2} is flipped from a 1 to a 0 or c_{i-3} is flipped from a 0 to a 1. If c_{i-2} and c_{i-3} are even, then this is precisely every other flip. If either c_{i-2} or c_{i-3} is odd, then c_i is dependent on both c_{i-2} and c_{i-3} as well as the initial conditions in x. We add one depending on whether or not the first flip of c_{i-2}, c_{i-3} causes c_i to flip as well.

A Sufficient Condition

Let p be a sequence where $p_i = c_i \mod 2$ for all i. Then if $x \oplus y$ looks like some p, then $y \in f^*(x)$. That is, if you fix your initial conditions and the above recurrence holds through $x \oplus y$, then $y \in f^*(x)$.

That being said, we necassarily don't know c_0.

$\underline{0}$ Orbit Checking is in NP

Our verifier takes in two strings x, y, and an index i. This index is the position of y in x's orbit. WLOG, suppose $x_0 = 1$. We first set $c_0 = i$, $c_1 = 0$, and $c_2 = \lfloor c_0/2 \rfloor$. We then calculate c_i and check that $c_i \mod 2 = x_i \oplus y_i$ for all i.

The certificate is poly length with respect to x and y, as the orbit of y has length at most 2^n (so the length of an index is at most n).

Semigroups Revisited

Recall that the states of A_2^3 form a semigroup $\mathcal{S}(A_2^3)$. This semigroup is commutative - it doesn't matter what order we apply the transductions in. This follows by commutativity of addition modulo 2 (suffices to consider each bit one at a time). This means that every element in $\mathcal{S}(A_2^3)$ looks like $\underline{0}^i\underline{1}^j\underline{2}^k$ for $i, j, k \in \mathbb{N}$.

Further, we have an identity element in $\mathcal{S}(A_2^3)$:

$$\underline{0}^2\underline{1}^2\underline{2} = I$$

Proof. Proof by induction - it suffices to show that c_i is even for all i. In fact, c_i will be 2.

c_0 is 2, each of the applications of $\underline{0}$ flip x_0 once. The applications of $\underline{1}$ and $\underline{2}$ skip it. c_1 is also 2 as each application of $\underline{1}$ flips x_1 once and $\underline{0}$ and $\underline{2}$ skip it. $c_2 = 2$ - $\underline{2}$ flips it once, $\underline{1}^2$ skips it, and exactly one of the two applications of $\underline{0}$ flip it.

Then $c_{i-2} = c_{i-3} = 2$ by induction, so

$$c_i = \lfloor (c_{i-3}/2) \rfloor + \lfloor (c_{i-2}/2) \rfloor = 1 + 1 = 2$$

□

This means that $\mathcal{S}(A_2^3)$ is already a group ($\underline{0}^{-1} = \underline{0}\underline{1}^2\underline{2}$ and so on). Further, since $\underline{2}$ is expressible in terms of $\underline{0}$ and $\underline{1}$, we have that $\mathcal{S}(A_2^3) = \{\underline{0}^i\underline{1}^j \mid i, j \in \mathbb{Z}\}$, giving us a concise data structure to represent elements of $\mathcal{S}(A_2^3)$.

Derivatives

We introduce the notion of so-called derivatives. They're named as such because they resemble normal calculus derivatives in many ways, though that's currently beyond my knowledge.

If f_u is the transduction corresponding to some state state u of a transducer, then $\partial_b(f_u) = f_v$ if state u transitions to state v on input b. Each transducer has a corresponding derivative table, which is effectively an adjacency list representation of the transducer.

For example, A_2^3 has the following representation:

- $\partial_0(\underline{0}) = \underline{2}$
- $\partial_1(\underline{0}) = \underline{1}$
- $\partial_b(\underline{2}) = \underline{1}$ for any b
- $\partial_b(\underline{1}) = \underline{0}$ for any b
- $\partial_0(\underline{0}^i) = \underline{1}^{\lfloor i/2 \rfloor}\underline{2}^{\lceil i/2 \rceil}$
- $\partial_1(\underline{0}^i) = \underline{1}^{\lceil i/2 \rceil}\underline{2}^{\lfloor i/2 \rfloor}$
- $\partial_b(\underline{1}^i) = \underline{0}^i$ for any b
- $\partial_b(\underline{2}^i) = \underline{1}^i$ for any b
- $\partial_0(\underline{0}^i\underline{1}^j\underline{2}^k) = \underline{0}^j\underline{1}^{\lfloor i/2 \rfloor + k}\underline{2}^{\lceil i/2 \rceil}$
- $\partial_1(\underline{0}^i\underline{1}^j\underline{2}^k) = \underline{0}^j\underline{1}^{\lceil i/2 \rceil + k}\underline{2}^{\lfloor i/2 \rfloor}$

We can extend this notion of differentiation to differentiating with respect to arbitrary strings, rather than single bits (simply iterate the differentiation). Then, for $x, y \in \mathbf{1}^*$, $f \in \mathcal{S}$

$$f(xy) = f(x)\partial_x f(y)$$

A_2^3 Orbit Checking is in NP

Call $f \in \mathcal{S}(A_2^3)$ *even* if f leaves the first bit of its input unchanged and *odd* otherwise.

It then suffices, given i as the index into the orbit as a certificate, $f \in \mathcal{S}(A_2^3)$ and strings x, y, to simply iterate through x, differentiating f as we go and asserting that $y_i = \partial_z f(x_i)$, where z is the prefix of x so far.

Parity checking f can be done in polynomial time - simply check the parity of $\underline{0}$ in f. Additionally, differentiation can be performed in polynomial time, as it's a couple of additions and a couple bit shifts. Since the value of i is at most 2^n where n is the input length, the length of i is polynomial with respect to x and y.

A_2^3 Orbit Checking is in P

The A_2^3 Orbit Relation is Rational

This is hard - need to write that vector reduction thing.

A New Class of Transducers - 1-Toggle-1-Split

1-Tog-1-Split Orbit Checking in P?

1-Tog-1-Split Orbit Relation Rational?

1-Tog Orbit Relation Rational?

TODO: Automate making the orbit automaton?