15-354: CDM	**K. Sutner**
Assignment 6	Due: **November 3, 2023, 24:00**.

1. Equational Theories (30)

Background

Suppose we have a language of first-order logic that has only function symbols (plus equality). In this case the only atomic formulae look like

$$f(x, g(x, y)) = h(y, x)$$

By an equational theory we mean a set Γ of equations of this kind (where one should think of all the variables as being universqally quantified). Equational theories are hugely important in algebra. By a model of Γ we mean any first-order structure that satisfies all the axioms in Γ.

Task

A. Suppose we have only one unary function symbol f and a single axiom of the form $\gamma_n \equiv f^n(x) = x$, $n \geq 0$. Find all the models of γ_n up to isomorphism.

B. Find a way to express groups as an equational theory.

C. Wurzelbrunft thinks he has found an exceedingly clever equational theory that has only infinite models. What do you say?

D. Show how to define a product operation on the models of an equational theory so that, for all models M_1 and M_2, the product $M_1 \times M_2$ is another model (that depends on both M_1 and M_2)

Comment For part (B), you need to specify the language as well as the axioms. For part (C), recall that all our first-order structures are required to have at least one element. Lastly, in (D), the product $M_1 \times M_2$ has to depend on both M_1 and M_2, and it has to be useful.

Solution: Equational Theories

Part A: Unary Function

Think of the models as digraphs with out-degree 1. For $n > 0$, each connected component is a cycle of length k where k divides n. There can be arbitrarily many of these components.

For $n = 0$ the axiom reads $x = x$ and has no impact. In this case, any digraph with out-degree 1 is a model. The connected components now look like one of the following: cycles with trees attached to them (the transients of the f orbits), or a copy of $\mathbb{N}$, or a copy of $\mathbb{Z}$.

Part B: Groups

Three functions of arity (2,1,0), say, f, $\lambda x.x^{-1}$, 1. Axioms $f(f(x, y), z) = f(x, f(y, z))$, $f(x, x^{-1}) = f(x^{-1}, x) = 1$, $f(x, 1) = f(1, x) = x$.

Part C: Wurzelbrunft

No, this cannot work. There is always a model with just one element, typically a banana. All function are constant.

Part D: Product

For simplicity, suppose there is only one binary function symbol f. Given models $M_i = \langle M_i; F_i \rangle$, define a new structure with carrier set $M = M_1 \times M_2$ and function $F((x_1, x_2), (y_1, y_2)) = (F_1(x_1, y_1), F_2(x_2, y_2))$.

It is easy to check that all the axioms are satisfied in the product.

2. Arithmetic Transducers (30)

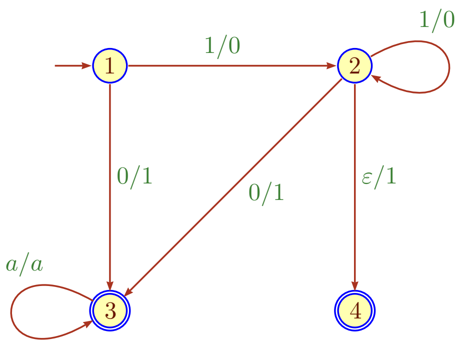
Background

In the following we will always use reverse binary representations of natural numbers, so the value of a string $x = x_0 x_1 \dots x_{n-1} \in \mathbf{2}^\star$ is $\mathsf{val}(x) = \sum_{i<n} x_i 2^i$. We do not allow trailing zeros, except for the string '0' with value $0 : \mathbb{N}$, so numbers are represented by the regular language $\mathcal{N} = \{0\} \cup \{0,1\}^\star 1$. Note that, restricted to $\mathcal{N}$, val is a bijection.

Now suppose we have some transducer $\mathcal{T}$ defining a transduction $\tau \subseteq \mathcal{N} \times \mathcal{N}$. We say that $\mathcal{T}$ implements an arithmetic function $f : \mathbb{N} \to \mathbb{N}$ if

$$\tau = \{ (\mathsf{val}(x), f(\mathsf{val}(x)) \mid x \in \mathbb{N} \}$$

For example, the following transducer implements the successor function:



Task

A. Construct a transducer that implements the function $n \mapsto n + 2$ and prove correctness.

B. Construct a transducer that implements the function $n \mapsto n + 3$ and prove correctness.

C. Construct a transducer that implements the function $n \mapsto 3n + 2$ and prove correctness.

Comment

"Construct" here means: build it explicitly, don't just argue that it must exist. The machines only require 5/7/6 states (assuming I did not screw up somewhere). It may help to assume initially that ε for zero and trailing zeros are allowed, and then refine the machines so that they handle the additional constraints.

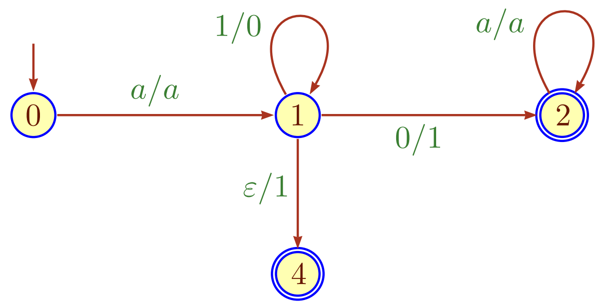
Solution: Arithmetic Transducers

Part A: Plus 2

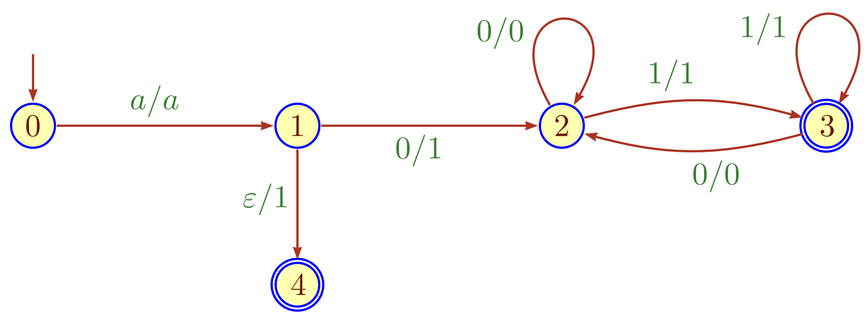
It is a good idea to think about correctness first: how do we design a machine that performs certain arithmetical operations? Let's focus on adding 2 and not worry about the details of the numeration system initially. In terms of bit-patterns, adding 2 means the following:

$$\begin{array}{lcl} a\,1^k & \rightsquigarrow & a\,0^k\,1 \\ a\,1^k\,0\,x & \rightsquigarrow & a\,0^k\,1\,x \\ a\,0\,x & \rightsquigarrow & a\,1\,x \end{array}$$

Here $k \geq 0$, $a \in \mathbf{2}$ and $x \in \mathbf{2}^\star$. The first rule increases the length of the string by 1, the other two are length-preserving. Suffix x can be handled by a copy-loop state (like state 3 above), and switching 1s to 0s by a flip-loop state (like state 2 above). Since there are only finitely many cases to check, one can verify that the following machine really works.



Alas, there is a glitch: this machine does not handle trailing zeros because of the copy state. Except for the $0 \rightsquigarrow 01$ complication, all transitions to a final state must have a 1 in the upper track. This can be enforced with a little surgery:

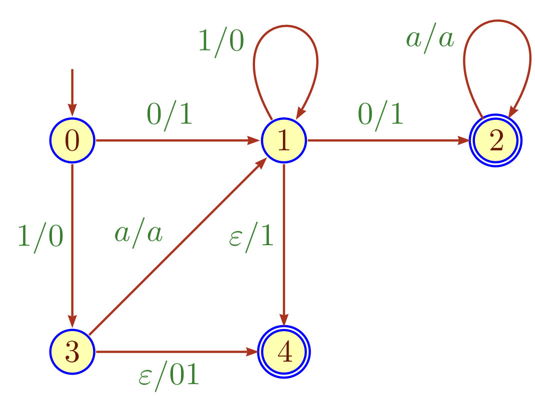


Part B: Plus 3

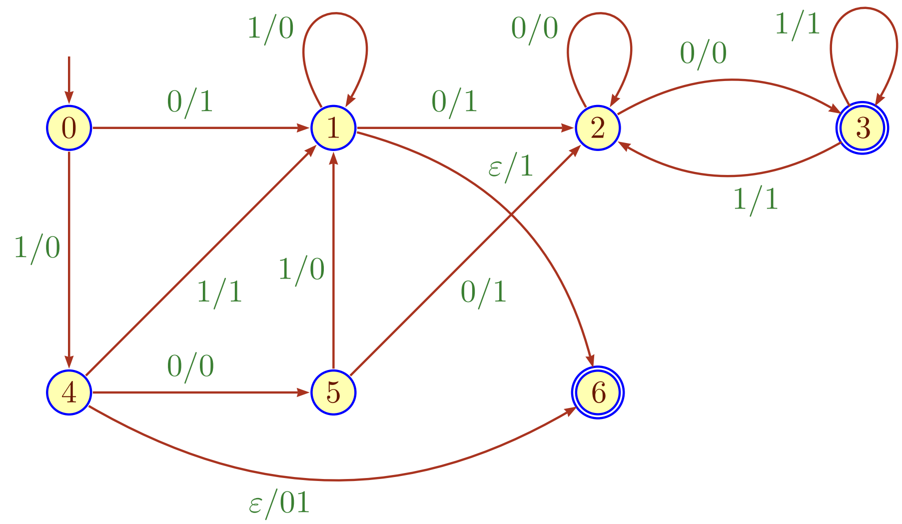
This time, the rules are slightly more complicated, we have to distinguish based on the first two input bits.

0	$\rightsquigarrow$	$1\,1$	1	$\rightsquigarrow$	$0\,0\,1$
$0\,0\,x$	$\rightsquigarrow$	$1\,1\,x$			
$0\,1\,1^k$	$\rightsquigarrow$	$1\,0\,0^k\,1$	$0\,1\,1^k\,0\,x$	$\rightsquigarrow$	$1\,0\,0^k\,1\,x$
$1\,a\,1^k$	$\rightsquigarrow$	$0\,a\,0^k\,1$	$1\,a\,1^k\,0\,x$	$\rightsquigarrow$	$0\,a\,0^k\,1\,x$

This leads to the following machine, which ignores the trailing zeros rule.

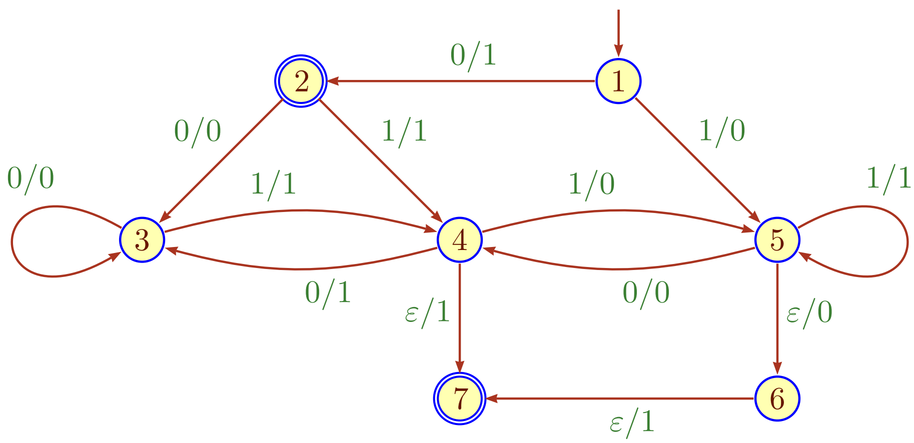


More fumbling produces a machine fully compliant with $\mathcal{N}$:

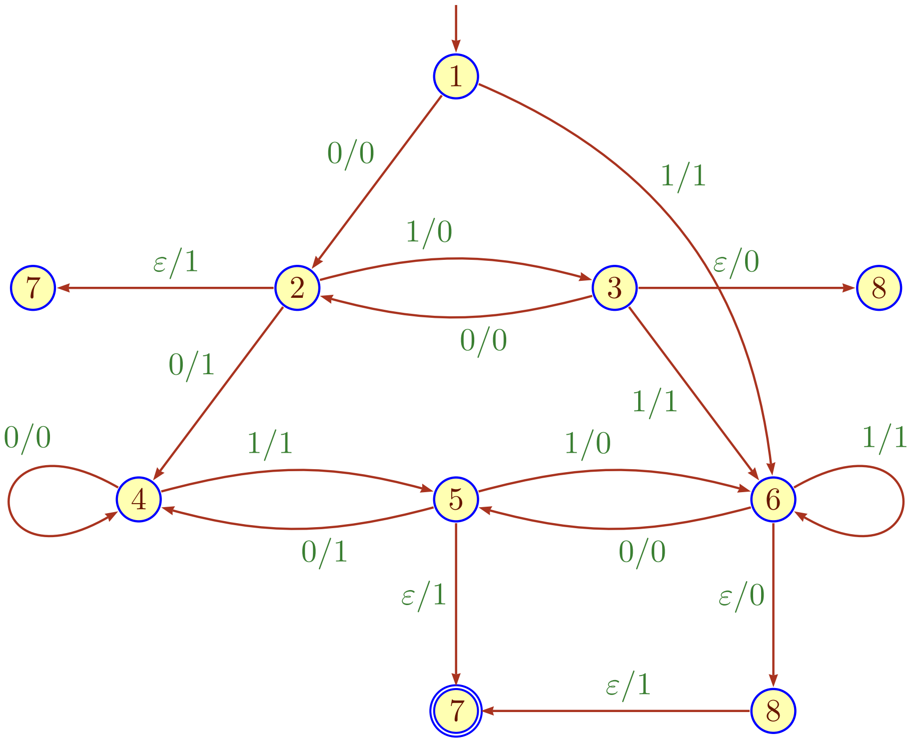


Part C: $3n + 2$

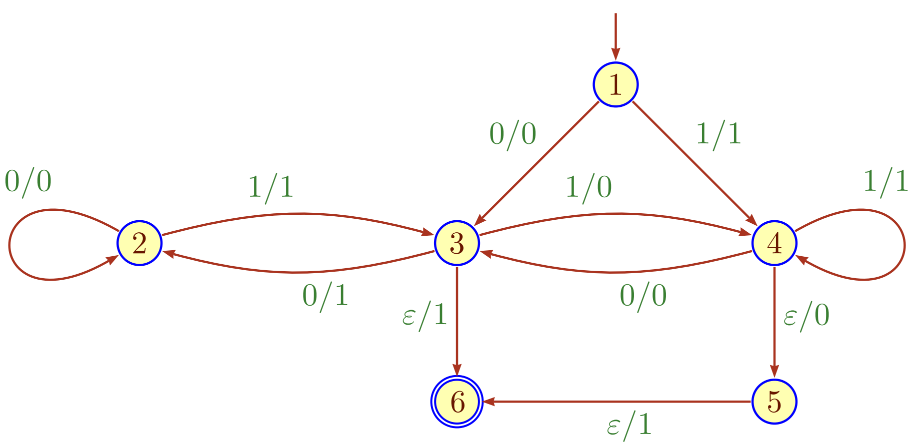
In class we mentioned a transducer implementing the Collatz function. It is easy to modifiy this machine to get a transducer T_{3x+1} that computes the arithmetical map $x \mapsto 3x + 1$ for all x (adjust the even branch).



Given T_{3x+1}, one needs to figure out how to increase the output by 1. We have to follow the computations in T_{3x+1}, starting at the initial state, and change the output behavior accordingly. The difficulty is to propagate the carry properly. With a bit of fumbling, this produces the following machine T_{3x+2}:



This is a bit messy, but careful inspection shows that we can merge states 2 and 5, as well as 3 and 6. This produces a better machine T_{3x+2} that is actually less complicated than T_{3x+1}.



3. Reversibility of ECA (40)

Background

Suppose $\rho : \mathbf{2}^3 \to \mathbf{2}$ is the local map of an elementary cellular automaton (i.e., a ternary Boolean function). We have seen how to construct from ρ a synchronous transducer $\mathcal{A}_{\twoheadrightarrow,x,y}$ that checks whether a finite bit sequence x evolves to y in one step under fixed boundary conditions. Naturally, there is a similar machine for cyclic boundary conditions, though things are a bit messier than in the fixed case.

Reversibility of a cellular automaton is expressed by the first-order formula

$$\mathsf{inj} \equiv \forall\, x, y, z\, (x \twoheadrightarrow z \wedge y \twoheadrightarrow z \Rightarrow x = y)$$

It is slightly easier to work with irreversibility, expressed by the negation $\mathsf{ninj} = \neg\mathsf{inj}$.

Task

A. Construct a synchronous 2-track transducer $\mathcal{B}_{\twoheadrightarrow}$ that checks whether a finite bit sequence x evolves to y in one step under cyclic boundary conditions.

B. Then build a synchronous 3-track transducer $\mathcal{A}$ that accepts the language defined by the matrix of ninj.

C. What does $\mathcal{A}$ have to do with injectivity of the global map on $\mathbf{2}^n$?

D. Explain how one can directly construct a 2-track transducer $\mathcal{A}'$ that still can be used to check ninj. This machine should be of the form $\mathcal{A}' = \mathcal{A}_0 \times \mathcal{U}$ where $\mathcal{U}$ is the un-equal transducer on 2 tracks.

Comment

For part (A), nondeterminism is critical (see the construction for the fixed boundary condition case). To avoid a silly edge case, let's assume that all words are non-empty.

Solution: Reversibility of ECA

Part A: Cyclic Boundary

The states are of the form $(xyz, st) \in \mathbf{2}^3 \times \mathbf{2}^2$ where s and t are used to nondeterministically guess the last and first bit, respectively.

$$\begin{aligned} \bot &\xrightarrow{a/e} stz, st & e &= \rho(s,t,z) \\ xyz, st &\xrightarrow{z/e} yzu, st & e &= \rho(y,z,u) \\ xys, st &\xrightarrow{s/e} \top & e &= \rho(y,s,t) \end{aligned}$$

Part B: 3-Track

We get a product machine

$$\mathcal{A} = \mathcal{A}_{\twoheadrightarrow,x,z} \times \mathcal{A}_{\twoheadrightarrow,y,z} \times \mathcal{U}_{x,y}$$

with attachments as indicated. Note that the standard $\mathcal{A}$ product machine construction will essentially produce 2 copies of $\mathcal{A}_{\twoheadrightarrow,x,z} \times \mathcal{A}_{\twoheadrightarrow,y,z}$, joined by transitions that verify $x \neq y$ (with initial state in the first copy, and final state in the second).

Part C: Length n

Clearly, $\mathcal{A}$ accepts some string of length n iff the global map fails to be injective on $\mathbf{2}^n$. This can be tested for example by computing the product between $\mathcal{A}$ and the "all strings of length n" machine (more algebraic solutions are better).

Part D: 2-Track

The standard decision procedure would eliminate the z track by removing the z-labels: we are guessing the z such that $x \twoheadrightarrow z$ and $y \twoheadrightarrow z$. This can be done by a constructing a product automaton $\mathcal{A}_0$ of the de Bruijn automaton in class: states are $\mathbf{2}^2 \times \mathbf{2}^2$ and transitions are

$$(a_1 a_2, b_1 b_2) \xrightarrow{a_3 : b_3} (a_2 a_3, b_2 b_3)$$

provided that $\rho(a_1, a_2, a_3) = \rho(b_1, b_2, b_3)$

One can then use the standard un-equal construction, two copies of $\mathcal{A}_0$, to produce $\mathcal{A}$.