Assignment - Brzozowski's Algorithm

Ralph Sarkis
November 16, 2020

Abstract

The main goal of this assignment is to prove the correctness of Brzozowski's algorithm to minimize a DFA. After recalling some notions in automata theory in Section 1, we will introduce (co)algebras and (co)induction in Section 2 and then use this new language to describe Brzozowski's algorithm and its proof of correctness in Section 3. Note that not all results in Section 2 are needed for Section 3 and Section 2 is interesting on its own without any knowledge of automata theory. We strongly encourage you to collaborate and talk to us if you have difficulties, but we ask that you write your own solution.

1 Preliminaries on Automata

There are no questions in this section, only stuff we believe you learned in a first course in theoretical CS. Give it a quick read in order to at least to identify my notation, the most important part is the definition of Brzozowski's algorithm in Section 1.3.

1.1 Deterministic Finite Automata

Definition 1 (DFA)**.** A **deterministic finite automaton** (DFA) M is composed of a **finite** alphabet Σ (a set of symbols), a finite set of **states** Q with a starting state identified by q_0, a **transition function** $\delta : Q \times \Sigma \to Q$ and a subset $F \subseteq Q$ of **accepting** states.

An input for M is a finite word (concatenation of finitely many elements) of Σ, we will denote it $w \in \Sigma^*$. The automaton reads its input letter by letter, starting in state q_0, and changes its state according to δ: $\delta(q, a) = q'$ means that if M is in state q and reads the symbol a, then M ends up in state q'. After reading all of its input M is in some state q and outputs "Accept" if $q \in F$ and "Reject" otherwise.

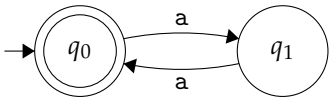
The global behavior of M is described by the subset $L \subseteq \Sigma^*$ of words that M accepts. We denote this subset $L(M)$ and say that $L(M)$ is the **language recognized** by M.

We will denote δ^* the extension of δ to Σ^*, it is defined inductively by

$$\delta^*(x, \varepsilon) = x \quad \text{and} \quad \delta^*(x, a \cdot w) = \delta^*(\delta(x, a), w).$$

Examples 2. Typically, it is more readable to describe DFAs by drawing a graphical representation than by defining each component.

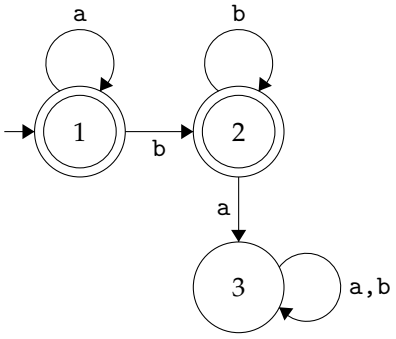
1. Consider the DFA described by $\Sigma = \{\mathtt{a}\}$, $Q = \{q_0, q_1\}$, $\delta = (q_0, \mathtt{a}) \mapsto q_1, (q_1, \mathtt{a}) \mapsto q_0$, $F = \{q_0\}$. It is represented by the following diagram.



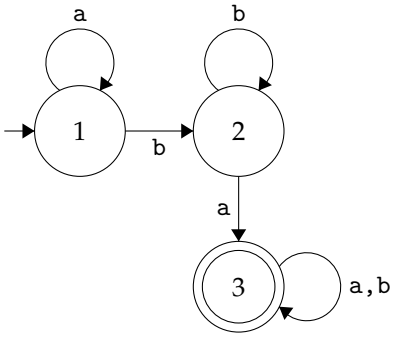
The circles represent the states and accepting states are denoted with a second inner circle. The arrows represent transitions and the labels are the symbols that need to be read for this transition to occur. The smaller arrow with no label designates the starting state.

It is easy to see that the language recognized by this DFA consists of all the words with an even number of a's (i.e.: $L(M) = \{\mathtt{a}^{2n} : n \in \mathbb{N}\}$).

2. The DFA recognizing the language $\{\mathtt{a}^n\mathtt{b}^m \mid n, m \in \mathbb{N}\}$ can be described as follows.



3. For any DFA M on an alphabet Σ, it is very easy to describe a DFA M' that recognizes the complement of $L(M)$, i.e.: such that $L(M') = \Sigma^* \setminus L(M)$. We just invert the roles of accepting and non-accepting states. Here is the DFA recognizing the complement of $\{\mathtt{a}^n\mathtt{b}^m \mid n, m \in \mathbb{N}\}$.



Remark 3. The term **deterministic** means that the transitions are completely determined by the input, that is, you can run the DFA on the same input many many times and it will always end in the same state and output the same thing. Mathematically, determinism comes from the fact that $\delta(q, a)$ takes the value of only one state for any $q \in Q$ and $a \in \Sigma$. If we allow δ to be nondeterministic, we get the definition of an NFA.

1.2 Nondeterminisitic Finite Automata

Definition 4. An **nondeterministic automaton** (NFA) M consists of a finite alphabet Σ, a finite set of states Q with a starting state identified by q_0, a transition function $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ and a subset $F \subseteq Q$ of accepting states.

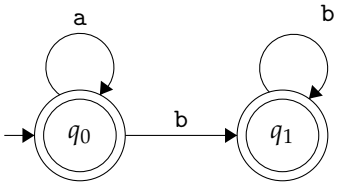
There are two main differences with a DFA. First, an NFA can sometimes make a transition without reading a symbol from the input but rather by reading ε (denoting an empty string). Second, the image of δ is a set of possible states for the transition to end in. Let us see how this affects the behavior of M.

The automaton still reads its input letter by letter starting in state q_0, but now instead of making the only possible transition, it makes all of them at the same time and continues the computation on multiple branches. The output of M is "Accept" if in at least one branch, all the input was read and M is in an accepting state, otherwise M outputs "Reject". The language recognized by M is defined as for DFAs.

Another way to view nondeterminism is to consider that M has access to an all-knowing oracle that will choose which transition to make (out of the possible ones). This oracle always make the choices that lead to accepting the input if possible, hence, M accepts $w \in \Sigma^*$ if and only if there is a sequence of choices that lead to an accepting state after reading all the input.

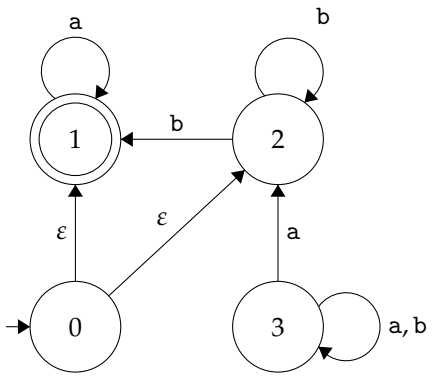
Examples 5. The representation of NFAs is really similar to that of DFAs but now, arrows can be labeled with an ε, multiple arrows coming out of the same state can have the same label, and states can have no arrows coming out with a specific label $a \in \Sigma$ (corresponding to the fact that $\delta(q, \mathtt{a}) = \emptyset$).

1. The NFA recognizing $\{\mathtt{a}^n\mathtt{b}^m \mid n, m \in \mathbb{N}\}$ is simpler than the DFA we drew above.

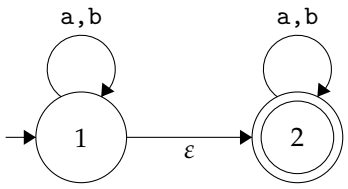


2. For any DFA M, we can easily construct an NFA M' that recognizes the reverse of $L(M)$, i.e.: $L(M') = \{w^{\mathcal{R}} \mid w \in L(M)\}$, where $^{\mathcal{R}}$ denotes the reverse of a word. Indeed, from the representation of a DFA, we can reverse all arrows, make the initial state an accepting state and add a pseudo initial state with ε–transitions to all the old accepting states as shown by the illustration below.

   The reverse of $\{\mathtt{a}^n\mathtt{b}^m \mid n, m \in \mathbb{N}\}$ is $\{\mathtt{b}^n\mathtt{a}^m \mid n, m \in \mathbb{N}\}$ and it is recognized by the following NFA.



3. Finding the complement of an NFA is not as easy as for DFAs. For instance, consider the following NFA on the alphabet $\{\mathtt{a}, \mathtt{b}\}$.



It recognizes all words generated by the alphabet, but if you swap the accepting and non-accepting states, the new NFA will also accept all words.

Remark 6. Since an NFA can have ε–transitions, the extension of δ to Σ^* is defined differently, $\delta^*(x, \varepsilon)$ is the ε–**closure** of x and $\delta^*(x, w)$ is defined inductively. More formally,

$$\delta^*(x, \varepsilon) = \{q \in Q : \exists\{x = x_0, x_2, \ldots, x_n = q\}, \forall 1 \leq i \leq n, x_i \in \delta(x_{i-1}, \varepsilon)\}$$
$$\delta^*(x, a \cdot w) = \bigcup_{q \in \delta(x,a)} \bigcup_{q' \in \delta^*(q,w)} \delta^*(q', \varepsilon).$$

Intuitively, $\delta^*(x, w)$ is the set of words that x can reach while reading the input w, taking into account that the machine can take any ε–transition.

One question that quickly arises is whether there are languages that can be recognized by an NFA, but not by any DFA. The converse is clearly false because any DFA can be written as an NFA where the image of the transition function only contains singletons, i.e.: there is only one possible state for each transition.

Surprisingly, the original question also has a negative answer. In other words, any NFA has a DFA recognizing the same language. To prove this, we describe the powerset construction transforming an NFA into an equivalent DFA.

Proof sketch. Let $M = (\Sigma, Q, q_0, \delta, F)$ be an NFA, we construct an equivalent DFA $M' = (\Sigma, Q', q_0', \delta', F')$ as follows.

- The states of M' are sets of states of M, that is $Q' = \mathcal{P}(Q)$.

- The initial state q_0' is the set of states that can be reached by reading no symbol when running M. Formally, we have

$$q_0' = \delta^*(q_0, \varepsilon).$$

- The transition function will simulate all the possible choices by transitioning between sets of states. For any $S \subseteq Q$ and $a \in \Sigma$,

$$\delta'(S, a) = \bigcup_{q \in S} \delta^*(q, a)$$

- A set of states is accepting if and only if it contains an accepting state.

It is left to show that $L(M) = L(M')$. □

The transformation we just described implies that NFAs are not necessary and we could always work with DFAs. However, observe that the size of the automaton increases exponentially in this procedure, so it is not practical to work with the resulting DFA. Consequently, one could ask whether we could further transform the DFA into a smaller (maybe smallest) but equivalent DFA. This is called DFA minimization and there are several polynomial time algorithms that solve this problem (Hopcroft's and Moore's). Another algorithm by Brzozowski is conceptually much simpler although it runs in exponential time in the worst case.

1.3 Brzozowski's Algorithm

Definition 7. Let M be a DFA, **Brzozowski's algorithm** is the following procedure.

1. Reverse M to obtain an NFA $M^{\mathcal{R}}$ which recognizes the reverse of $L(M)$.
2. Use the powerset construction to obtain a DFA $D(M^{\mathcal{R}})$, still recognizing the reverse of $L(M)$.
3. Discard all unreachable states of $D(M^{\mathcal{R}})$, denote the automaton obtained with N.
4. Apply the same procedure (steps 1 to 3) to N, i.e.: reverse N, determinize the result and discard unreachable states to obtain O.

Proposition 8. *The final automata O is the automaton with the least number of states satisfying $L(O) = L(M)$. Another way to say this: if* rev *denotes the reversing operation,* det *denotes the determinization and* reach *denotes the operation of removing unreachable states, then*

$$\mathsf{reach}(\mathsf{det}(\mathsf{rev}(\mathsf{reach}(\mathsf{det}(\mathsf{rev}(M))))))$$

is the minimal automaton equivalent to M.

Remark 9. This result should be very surprising. Brzozowski's algorithm simply reverses the automaton twice to obtain a minimal form. Moreover, during the procedure, there are two steps inducing an exponential blow-up of the number of states. Indeed, determinization via the powerset construction leads to an automaton with $2^{|Q|}$ states. That

means that in the worst cases, $\mathsf{det}(\mathsf{rev}(\mathsf{reach}(\mathsf{det}(\mathsf{rev}(M)))))$ could have $2^{2^{|Q|}}$ states. This is the reason why Brzozowski's algorithm can take exponential time. Nonetheless, in the end, you still end up with a minimal automaton. Another surprising thing, this algorithm often performs better than the worst case scenario.

This algorithm was first proven correct using complex combinatorial arguments in 1963, but in this assignment, we will give a much simpler proof using coalgebras. The next section is dedicated to introducing coalgebras and coinduction via their duals, algebras and induction.

2 (Co)algebras and (Co)induction

In this section, we will introduce (co)algebras and (co)induction and ask you to fill out some gaps. While the definitions are given for an arbitrary category $\mathbf{C}$, all of the examples and exercises will be done on **Set**.

2.1 Algebras

The term *algebra* has a few different meanings and here we will more precisely consider F–algebras for some endofuntor $F : \mathbf{C} \rightsquigarrow \mathbf{C}$. Nevertheless, all the objects that are referred to as algebras have a common motto: *algebras only care about structure*.

For instance, in a first year algebra course, groups are studied up to isomorphisms (maps that preserve the structure) because all the useful properties of a group are determined completely by how the operation acts on the underlying set. As a concrete example, the groups $\mathbb{Z}_2 \times \mathbb{Z}_3$ and $\mathbb{Z}_6$ are the same group even if their elements have different names. It is a similar situation for rings, vector spaces and a lot more mathematical objects.

Before giving the general definition of an F–algebra, we categorify the definition of a group.

Example 10. Usually, a group is defined as a set G along with an operation $\cdot : G \times G \to G$ satisfying some conditions, namely, associativity, existence of an identity and existence of an inverse for each element. It is then a formal consequence that the identity and inverses are unique.

Therefore, it is equivalent to define a group as a set G with a binary operation $\cdot$, an identity $1 \in G$ and an inverse g^{-1} for all $g \in G$ that satisfy some properties. In order to abide to the categorical mindset, it is better to view the identity as a morphism $1 : \mathbf{1} \to G$ ($\mathbf{1}$ is the final object, i.e.: a singleton) and describe inverses with a morphism $(-)^{-1} : G \to G$. A few additional diagrams have to commute for G to satisfy all axioms of a group, but we leave their construction as an exercise. We conclude that a group can be seen as a morphism

$$[1, (-)^{-1}, \cdot] : \mathbf{1} + G + (G \times G) \to G.$$

This is our first example of an F–algebra, here $F : \mathbf{Set} \rightsquigarrow \mathbf{Set}$ sends a set G to $\mathbf{1} + G + (G \times G)$ and a morphism f to $[\mathrm{id}_{\mathbf{1}}, f, (f, f)]$.

Note that since we have not used the fact that G is a set, this definition gives rise to groups in other categories than **Set** provided they have a final object, products and coproducts.

Exercise 1 (1pt)**.** Draw the additional diagrams that G, $\cdot$, 1 and $(-)^{-1}$ should satisfy to obtain a group.

Definition 11 (F–algebra)**.** Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor, an F–**algebra** is an object $A \in \mathbf{C}_0$ along with a morphism $\alpha : F(A) \to A \in \mathbf{C}_1$ called the **structure map**.

Examples 12.

1. Since a monoid M only has a binary operation and an identity, it can be represented as an algebra $[1, \cdot] : \mathbf{1} + (M \times M) \to M$. Similarly, one can construct algebras that represent rings and vector spaces, but not fields (why?).
2. We will see later that the induction principle we know comes from the algebra $[0, \mathsf{succ}] : \mathbf{1} + \mathbb{N} \to \mathbb{N}$, where $0(*) = 0$ and $\mathsf{succ}(n) = n + 1$.
3. Although we will not use them often, there are algebras in different categories than **Set**. In computer science, we often use induction to reason about lists, we will see that this is because lists are algebras. More precisely, for a type A, the type A^* of lists with elements of type A is has an algebra structure $[\mathsf{nil}, \mathsf{cons}] : 1 + (A \times A^*) \to A^*$ given by $\mathsf{nil}(*) = \varepsilon \in A^*$ (the empty list) and $\mathsf{cons}(a, w) = w \cdot a \in A^*$ (concatenation). The category in which this algebra lives depends on the programming language and types considered.

Remark 13. The components of the structure map (i.e.: 0 and succ in the second example) are often called the **constructors** because they define rules to construct elements of the algebra using other elements as building blocks.

As you might expect F–algebras form a category, denoted $\mathrm{Alg}(F)$, with the following notion of morphism.

Definition 14 (F–algebra homomorphism)**.** Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor and $\alpha : F(A) \to A$ and $\beta : F(B) \to B$ be F–algebras. An F–**algebra homomorphism** from the former to the latter is a morphism $f : A \to B$ that makes this square commute.

$$\begin{array}{ccc} F(A) & \xrightarrow{F(f)} & F(B) \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ A & \xrightarrow[f]{} & B \end{array} \tag{1}$$

This definition also clarifies why we require F to be a functor.

Example 15. Let $F = X \mapsto \mathbf{1} + X + (X \times X)$ be the functor discussed Example 10. An F–algebra homomorphism is represented by the following square.

$$\begin{array}{ccc} \mathbf{1} + G + (G \times G) & \xrightarrow{[\mathrm{id}_{\mathbf{1}}, f, (f,f)]} & \mathbf{1} + H + (H \times H) \\ {\scriptstyle[1_G, (-)^{-1}, \cdot]}\downarrow & & \downarrow{\scriptstyle[1_H, (-)^{-1}, \cdot]} \\ G & \xrightarrow[f]{} & H \end{array} \tag{2}$$

Unwrapped, this says that $f(1_G) = 1_H$, $f(g^{-1}) = f(g)^{-1}$ and $f(g \cdot g') = f(g) \cdot f(g')$ for all $g, g' \in G$, i.e.: if both algebras represent groups as seen in Example 10, it is a group homomorphism.

2.2 Coalgebras

Now that we have a categorical notion of algebra, we can look at its dual.

Definition 16 (F–coalgebra)**.** Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor, an F–**coalgebra** is an object $A \in \mathbf{C}_0$ called the **carrier** along with a morphism $\omega : A \to F(A)$ called the **behavior map**. We will refer to a coalgebra with A, ω or the pair (A, ω).

Examples 17.

1. If F is the identity on **Set**, then an F–coalgebra is just an endomorphism $\omega : A \to A$ and it is sometimes called a **dynamical system**. You can think of the elements of A as states and ω as the transition map for the system.
2. Let $\mathsf{Str}_{\mathbb{N}} : \mathbf{Set} \rightsquigarrow \mathbf{Set} = \mathbb{N} \times (-)$ be the functor sending a set X to $\mathbb{N} \times X$ and a function $f : X \to Y$ to $\mathrm{id}_{\mathbb{N}} \times X : \mathbb{N} \times X \to \mathbb{N} \times Y$. An example of a $\mathsf{Str}_{\mathbb{N}}$–coalgebra is the set $\mathbb{N}^{\mathbb{N}}$ of all infinite sequences (also called **streams**) of natural numbers with the structure map $(\mathsf{head}, \mathsf{tail}) : \mathbb{N}^{\mathbb{N}} \to \mathbb{N} \times \mathbb{N}^{\mathbb{N}}$ given by

$$\mathsf{head} : \mathbb{N}^{\mathbb{N}} \to \mathbb{N} = \sigma \mapsto \sigma(0) \quad \text{and} \quad \mathsf{tail} : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}} = \sigma \mapsto \sigma \circ \mathsf{succ}.$$

   Unsurprisingly, we call $\mathsf{head}(\sigma)$ and $\mathsf{tail}(\sigma)$ the **head** and **tail** of the stream σ respectively.

Exercise 2 (1pt)**.** Denoting $2 = \{0, 1\}$, let $F = 2 \times (-)^A$ send a set X to $2 \times X^A$ and a function $f : X \to Y$ to $\mathrm{id}_2 \times (f \circ -) : 2 \times X^A \to 2 \times Y^A$. When A is finite, show that there is a correspondence between F–coalgebras with a finite carrier and DFAs without initial states.

Remark 18. The components of the behavior map (i.e.: h and t in the second example) are often called **destructors** or **observers** because they decompose elements of the coalgebra.

We define morphisms of F–coalgebras in order to obtain a category $\mathrm{Coalg}(F)$.

Definition 19 (F–coalgebra homomorphism)**.** Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor and $\alpha : A \to F(A)$ and $\beta : B \to F(B)$ be F–coalgebras. An F–**coalgebra homomorphism** from the former to the latter is a morphism $f : A \to B \in \mathbf{C}_1$ that makes (3) commute.

$$\begin{array}{ccc} A & \xrightarrow{\quad f \quad} & B \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ F(A) & \xrightarrow[F(f)]{} & F(B) \end{array} \tag{3}$$

2.3 Induction

Induction is a very well known and prevalent proof principle. In it most common form, it says that for any predicate P on $\mathbb{N}$, if $P(0)$ is true and $P(n) \implies P(n+1)$ is true for any $n \in \mathbb{N}$, then so is $P(n)$ for any $n \in \mathbb{N}$. In this section, we use the power of algebras to generalize this proof principle and give a few examples.

Definition 20 (Initial algebra)**.** Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor, an **initial algebra** is an initial object in the category of F–algebras. Namely, it is an algbera (A, α) such that for any other algebra (B, β), there is a unique $f : A \to B$ making the following square commute.

$$\begin{array}{ccc} F(A) & \xrightarrow{F(f)} & F(B) \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ A & \xrightarrow[f]{} & B \end{array} \tag{4}$$

Example 21. The algebra $(\mathbb{N}, [0, \mathsf{succ}])$ is initial for the functor $\mathbf{1} + (-)$. Indeed, let $[z, s] : \mathbf{1} + X \to X$ be another algebra for this functor, then a map $f : \mathbb{N} \to X$ that makes the following diagram commute must necessarily satisfy $f(0) = z(*)$ and $f(n) = s^n(z(*))$.

$$\begin{array}{ccc} \mathbf{1} + \mathbb{N} & \xrightarrow{[\mathrm{id}_{\mathbf{1}}, f]} & \mathbf{1} + X \\ {\scriptstyle[0,\mathsf{succ}]}\downarrow & & \downarrow{\scriptstyle[z,s]} \\ \mathbb{N} & \xrightarrow[f]{} & X \end{array} \tag{5}$$

This completely determines f and moreover, defining f like this for any $(\mathbf{1} + (-))$-algebra $(X, [z, s])$ yields an algebra homomorphism.

Exercise 3 (2pts)**.** Show that the algebra for lists $[\mathsf{nil}, \mathsf{cons}] : 1 + A \times A^* \to A^*$ is initial for the functor $1 + A \times (-) : \mathbf{Set} \rightsquigarrow \mathbf{Set}$ (you know its action on sets, on morphisms it sends $f : X \to Y$ to $[\mathrm{id}_{\mathbf{1}}, (\mathrm{id}_A, f)]$).

We already know that initial objects are unique up to unique isomorphisms, but Lambek also showed furthermore that initial F–algebras are fixed points of F.

Proposition 22 (Lambek)**.** *Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$, if (A, α) is an initial F–algebra, then $\alpha : F(A) \to A$ is an isomorphism.*

Exercise 4 (1.5pts)**.** Prove Proposition 22. **Hint:** Consider the algebra $F(\alpha) : F^2(A) \to F(A)$.

Initial algebras generalize the inductive reasoning we use with the natural numbers to much more settings. We distinguish two cases where induction is used: inductive definitions and the induction proof principle.

For the former, the general idea is that, given an initial F–algebra (A, α), we can easily define a function $f : A \to B$ by looking at how it acts on constructors. Indeed, with only this data, we can construct an F–algebra structure on B such that the unique homomorphism $! : A \to B$ acts exactly like f.

Example 23 (Inductive definition)**.** Recall that $(A^*, [\mathsf{nil}, \mathsf{cons}])$ is the initial $(1 + A \times (-))$-algebra. We would like to define the function $\mathsf{len} : A^* \to \mathbb{N}$ that computes the length of a list. Intuitively, it satisfies the equations

$$\mathsf{len}(\mathsf{nil}) = 0 \qquad \mathsf{len}(\mathsf{cons}(a, l)) = 1 + \mathsf{len}(l).$$

Then, if we construct the $(1 + A \times (-))$-algebra $[z, s] : 1 + A \times \mathbb{N} \to \mathbb{N}$ defined by $z(*) = 0$ and $s(a, n) = 1 + n$, we can verify that the unique algebra homomorphism $! : A^* \to \mathbb{N}$ is the function len because both make the following diagram commute.

$$\begin{array}{ccc} 1 + A \times A^* & \xrightarrow{1+\mathrm{id}_A \times !} & 1 + A \times \mathbb{N} \\ {\scriptstyle [\mathsf{nil},\mathsf{cons}]}\downarrow & & \downarrow{\scriptstyle [z,s]} \\ A^* & \xrightarrow[!=\mathsf{len}]{} & \mathbb{N} \end{array} \tag{6}$$

Exercise 5 (1pt)**.** Use the initiality of $\mathbb{N}$ for the functor $1 + (-)$ to define the function $n \mapsto 2^n$.

Generalizing proofs by induction in this context is more involved and we will need the definition of F–congruences. While F–algebra homomorphisms are maps between algebras that preserve the structure, an F–congruence is a relation between two algebras that preserves the structure.

Definition 24. Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor and (A, α) and (B, β) be F–algebras, a relation $R \subseteq A \times B$ is an **F-congruence** if there is a structure map $\gamma : F(R) \to R$ such that the projections $\pi_1 : R \to A$ and $\pi_2 : R \to B$ are algebra homomorphisms making this diagram commute.

$$\begin{array}{ccccc} F(A) & \xleftarrow{F\pi_1} & F(R) & \xrightarrow{F\pi_2} & F(B) \\ {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \gamma} & & \downarrow{\scriptstyle \beta} \\ A & \xleftarrow[\pi_1]{} & R & \xrightarrow[\pi_2]{} & B \end{array} \tag{7}$$

Example 25. If F is the identity functor, then for any algebras (A, α) and (B, β) and any relation $R \subseteq A \times B$, $\gamma = (\alpha \circ \pi_1, \beta \circ \pi_2)$ is a structure map making R into an F–congruence.

Exercise 6 (2pts)**.** Let $F = \mathbf{1} + (-)$, we have already seen that $\mathbb{N}$ is an initial F–algebra.

(a) Give a necessary and sufficient condition for $R \subseteq \mathbb{N} \times \mathbb{N}$ to be an F–congruence.

(b) Conclude that for any F–congruence $R \subseteq \mathbb{N} \times \mathbb{N}$, $\forall n \in \mathbb{N}, (n, n) \in \mathbb{R}$.

The next theorem generalizes the previous exercise.

Theorem 26 (General induction)**.** *Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor and (A, α) be an initial F–algebra, if $R \subseteq A \times A$ is an F–congruence, then it is reflexive, that is $(a, a) \in R$ for all $a \in A$.*

Exercise 7 (1.5pts)**.** Prove Theorem 26.

Example 27 (Induction in $\mathbb{N}$)**.** We will see how the induction principle in Theorem 26 implies the usual induction principle. Let P be a predicate on $\mathbb{N}$ that satisfies $0 \in P$ and $n \in P \implies n + 1 \in P$. One can show that $P \times P \subseteq \mathbb{N} \times \mathbb{N}$ is an F–congruence and by general induction, $(n, n) \in P \times P$ for all $n \in \mathbb{N}$, i.e.: $\forall n \in \mathbb{N}, n \in P$.

Although going through all these abstractions and definitions seems like a really convoluted way to prove the induction principle, it lead us to two new concepts. First, we can now use inductive reasoning on all sorts of algebras even if they are in no way similar to $\mathbb{N}$. Second, we obtained an easy access to the dual of induction which we present in the following section.

2.4 Coinduction

Definition 28 (Final coalgebra)**.** Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor, a **final coalgebra** is a final object in the category of F–coalgebras. Namely, it is a coalgbera (A, ω) such that for any other coalgebra (B, ψ), there is a unique morphism $f : B \to A$ making the following square commute.

$$\begin{array}{ccc} B & \xrightarrow{\quad f \quad} & A \\ {\scriptstyle\psi}\downarrow & & \downarrow{\scriptstyle\omega} \\ F(B) & \xrightarrow[F(f)]{} & F(A) \end{array} \tag{8}$$

Since final coalgebras are unique up to unique homomorphism, we will refer to **the** final coalgebra.

Example 29. The $\mathsf{Str}_{\mathbb{N}}$–coalgebra $(\mathsf{head}, \mathsf{tail}) : \mathbb{N}^{\mathbb{N}} \to \mathbb{N} \times \mathbb{N}^{\mathbb{N}}$ is final. That is, for any $\mathsf{Str}_{\mathbb{N}}$–coalgebra, $(h, t) : X \to \mathbb{N} \times X$, there is unique morphism $! : X \to \mathbb{N}^{\mathbb{N}}$ making (9) commute.

$$\begin{array}{ccc} X & \xrightarrow{\quad ! \quad} & \mathbb{N}^{\mathbb{N}} \\ {\scriptstyle(h,t)}\downarrow & & \downarrow{\scriptstyle(\mathsf{head},\mathsf{tail})} \\ \mathbb{N} \times X & \xrightarrow[\mathrm{id}_N \times !]{} & \mathbb{N} \times \mathbb{N}^{\mathbb{N}} \end{array} \tag{9}$$

The equation corresponding to (9) is $(\mathsf{head}, \mathsf{tail}) \circ ! = (\mathrm{id}_{\mathbb{N}} \times !) \circ (h, t)$. We can decompose it into $\mathsf{head} \circ ! = h$ and $\mathsf{tail} \circ ! = ! \circ t$. The first equation tells us that $!(x)$ starts with the number $h(x)$ and the second equation tells us that the tail of $!(x)$ is the stream corresponding to $t(x)$ (via $!$). In short, we have $!(x) = h(x) \cdot !(t(x))$. If we further decompose the tail, we obtain

$$!(x) = h(x) \cdot h(t(x)) \cdot h(t(t(x))) \cdots h(t^n(x)) \cdots .$$

This should convince you that the only suitable choice for $!$ is $x \mapsto (n \mapsto h(t^n(x)))$.

Exercise 8 (3pts)**.** Let $F = 2 \times (-)^A$ with A finite and consider the F–coalgebra

$$(\varepsilon?, \omega) : 2^{A^*} \to 2 \times (2^{A^*})^A,$$

where for a language $L \subseteq A^*$ (ε denotes the empty string),

$$\varepsilon?(L) = \begin{cases} 1 & \varepsilon \in L \\ 0 & \text{o/w} \end{cases}, \qquad \omega(L) = a \mapsto L_a = \{w \in A^* \mid a \cdot w \in L\}.$$

The language $L_a = \omega(L)(a)$ is sometimes called the left a–derivative of L. Given a DFA M corresponding to the coalgebra $[f, \delta] : Q \to 2 \times Q^A$, show that the function

$$o : Q \to 2^{A^*} = q \mapsto \{w \in A^* \mid M \text{ accepts } w \text{ when starting in state } q\}$$

is the only map making (10) commute.

$$\begin{array}{ccc} & & 2 \\ & \overset{f}{\nearrow} & \uparrow{\scriptstyle \varepsilon?} \\ X & \xrightarrow{o} & 2^{A^*} \\ {\scriptstyle \delta}\downarrow & & \downarrow{\scriptstyle \omega} \\ X^A & \xrightarrow[o^A]{} & (2^{A^*})^A \end{array} \tag{10}$$

Remark 30. Extending your proof to X not necessarily finite, you could obtain the fact that $(2^{A^*}, (\varepsilon?, \omega))$ is the final $2 \times (-)^A$ coalgebra.

Proposition 31. *If $\omega : A \to F(A)$ is a final F–coalgebra, then ω is an isomorphism.*

Exercise 9 (0.5pts)**.** Prove Proposition 31.

Final coalgebras lets us use **coinductive definitions**. You will see that they are quite similar to inductive definitions you are used to. In fact, as the name suggests, they are dual to each other, but we will not make this formal here.

Examples 32 (Coinductive definitions)**.** Fix some set A and consider the functor $\mathsf{Str}_A = A \times (-)$, similarly to $\mathsf{Str}_{\mathbb{N}}$, the set $A^{\mathbb{N}}$ of streams in A is the final Str_A–coalgebra with the behavior map $(\mathsf{head}, \mathsf{tail})$ as defined in Example 17. We will define three different maps using the finality of $A^{\mathbb{N}}$.

1. The function $\mathsf{even} : A^{\mathbb{N}} \to A^{\mathbb{N}}$ takes a stream $\sigma = (\sigma(0), \sigma(1), \dots)$ and maps it to the stream of elements of σ at even positions, namely $\mathsf{even}(\sigma) = (\sigma(0), \sigma(2), \dots)$. To define it coinductively, we need to describe how destructors act on it. It is easy to verify that

   $$\mathsf{head}(\mathsf{even}(\sigma)) = \mathsf{head}(\sigma) \quad \text{and} \quad \mathsf{tail}(\mathsf{even}(\sigma)) = \mathsf{even}(\mathsf{tail}(\mathsf{tail}(\sigma))).$$

   Hence, if we define a new Str_A–coalgebra on $A^{\mathbb{N}}$ by $(h, t) = (\mathsf{head}, \mathsf{tail}^2 = \mathsf{tail} \circ \mathsf{tail})$, then we conclude by finality and commutativity of the following diagram that $! : A^{\mathbb{N}} \to A^{\mathbb{N}}$ is the function even.

   $$\begin{array}{ccc} A^{\mathbb{N}} & \xrightarrow{!} & A^{\mathbb{N}} \\ {\scriptstyle (\mathsf{head},\mathsf{tail}^2)}\downarrow & & \downarrow{\scriptstyle (\mathsf{head},\mathsf{tail})} \\ A \times A^{\mathbb{N}} & \xrightarrow[\mathrm{id}_A \times !]{} & A \times A^{\mathbb{N}} \end{array} \tag{11}$$

2. The operation of merging two streams is described by the function $\mathsf{merge} : A^{\mathbb{N}} \times A^{\mathbb{N}} \to A^{\mathbb{N}}$ mapping (σ, τ) to $(\sigma(0), \tau(0), \sigma(1), \tau(1), \dots)$. Observe that destructors act as follows:

   $$\mathsf{head}(\mathsf{merge}(\sigma, \tau)) = \mathsf{head}(\sigma) \quad \text{and} \quad \mathsf{tail}(\mathsf{merge}(\sigma, \tau)) = \mathsf{merge}(\tau, \mathsf{tail}(\sigma)).$$

The existence of merge is then proven with finality of $A^{\mathbb{N}}$ and the following coalgebra behavior map (where π_1 and π_2 are the projections):

$$(\mathsf{head} \circ \pi_1, \pi_2, \mathsf{tail} \circ \pi_1) : A^{\mathbb{N}} \times A^{\mathbb{N}} \to A \times A^{\mathbb{N}} \times A^{\mathbb{N}}.$$

Exercise 10 (1pts)**.** Similarly to the first item, show that the function $\mathsf{odd} : A^{\mathbb{N}} \to A^{\mathbb{N}}$ mapping $\sigma = (\sigma(0), \sigma(1), \dots)$ to $\mathsf{odd}(\sigma) = (\sigma(1), \sigma(3), \dots)$ can be defined coinductively.

There is also a dual to the induction proof principle for which we need to define bisimulation.

Definition 33 (F–bisimulation)**.** Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor and (A, ω) and (B, ψ) be F–coalgebras, a relation $R \subseteq A \times B$ is an **F-bisimulation** if there is a behavior map $\gamma : R \to F(R)$ such that the projections $\pi_1 : R \to A$ and $\pi_2 : R \to B$ are coalgebra homomorphisms making this diagram commute.

$$\begin{array}{ccccc} A & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & B \\ {\scriptstyle\omega}\downarrow & & \downarrow{\scriptstyle\gamma} & & \downarrow{\scriptstyle\psi} \\ F(A) & \xleftarrow[F\pi_1]{} & F(R) & \xrightarrow[F\pi_2]{} & F(B) \end{array} \tag{12}$$

Theorem 34 (Coinductive proof principle)**.** *Let $F : \mathbf{C} \rightsquigarrow \mathbf{C}$ be a functor and (A, ω) be the final F–coalgebra, if $R \subseteq A \times A$ is an F–bisimulation, then it is contained in the diagonal relation, that is $(a, a') \in R$ implies $a = a'$.*

Exercise 11 (0.5pts)**.** Prove Theorem 34.

Example 35. We will use coinduction to prove that $\mathsf{odd}(\mathsf{merge}(\sigma, \tau)) = \tau$. By the previous theorem, it is enough to show that $\mathcal{R} = \{(\mathsf{odd}(\mathsf{merge}(\sigma, \tau)), \tau) \mid \sigma, \tau \in A^{\mathbb{N}}\}$ is an F–bisimulation. We claim that $\gamma = (x, y) \mapsto (\mathsf{head}(x), (\mathsf{tail}(x), \mathsf{tail}(y)))$ makes the following diagram commute.

$$\begin{array}{ccccc} A^{\mathbb{N}} & \xleftarrow{\pi_1} & \mathcal{R} & \xrightarrow{\pi_2} & A^{\mathbb{N}} \\ {\scriptstyle(\mathsf{head},\mathsf{tail})}\downarrow & & \downarrow{\scriptstyle\gamma} & & \downarrow{\scriptstyle(\mathsf{head},\mathsf{tail})} \\ A \times A^{\mathbb{N}} & \xleftarrow[\mathrm{id}_A \times \pi_1]{} & A \times \mathcal{R} & \xrightarrow[\mathrm{id}_A \times \pi_2]{} & A \times A^{\mathbb{N}} \end{array} \tag{13}$$

To prove our claim, first note that

$$\begin{aligned} \mathsf{head}(\mathsf{tail}(\mathsf{merge}(\sigma, \tau))) &= \mathsf{head}(\mathsf{merge}(\tau, \mathsf{tail}(\sigma))) \\ &= \mathsf{head}(\tau), \end{aligned}$$

so if we can show that $(\mathsf{tail}(x), \mathsf{tail}(y)) \in \mathcal{R}$ for any $(x, y) \in \mathcal{R}$, then we would conclude that the diagram commutes. This last part follows from the derivation

$$\begin{aligned} \mathsf{tail}(\mathsf{odd}(\mathsf{merge}(\sigma, \tau))) &= \mathsf{odd}(\mathsf{tail}(\mathsf{tail}(\mathsf{merge}(\sigma, \tau)))) \\ &= \mathsf{odd}(\mathsf{tail}(\mathsf{merge}(\tau, \mathsf{tail}(\sigma)))) \\ &= \mathsf{odd}(\mathsf{merge}(\mathsf{tail}(\sigma), \mathsf{tail}(\tau))). \end{aligned}$$

Indeed, we obtain

$$(\mathsf{tail}(\mathsf{odd}(\mathsf{merge}(\sigma,\tau),\mathsf{tail}(\tau)) = (\mathsf{odd}(\mathsf{merge}(\mathsf{tail}(\sigma),\mathsf{tail}(\tau))),\mathsf{tail}(\tau)) \in \mathcal{R}.$$

Exercise 12 (3pts). (a) Given $f : A \to A$ coinductively define $\mathsf{map}_f : A^{\mathbb{N}} \to A^{\mathbb{N}}$ such that $\mathsf{map}_f(\sigma) = (f(\sigma(0)), f(\sigma(1)), \dots)$.

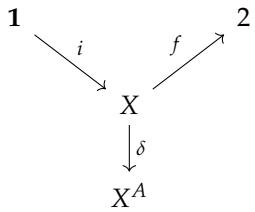
(b) Show by coinduction that for any $\sigma \in A^{\mathbb{N}}$, $\mathsf{even}(\mathsf{map}_f(\sigma)) = \mathsf{map}_f(\mathsf{even}(\sigma))$.

3 Brzozowski's Algorithm Coalgebraically

In this section we follow the proof given in this paper (answers are in there).

3.1 Reachability and Observability

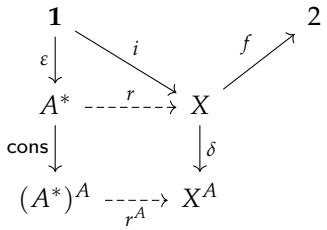
Let A be an alphabet, and (X, i, δ, F) be deterministic automaton, where X is the set of states, $\delta : X \times A \to X$ is the transition function, $i \in X$ is the initial state and F is the set of accepting states. For our purposes, we will view i as a function $i : \mathbf{1} \to X$, F as a function $f : X \to 2 = \{0, 1\}$ and δ will also denote the curried version $\delta : X \to X^A$. This leads to the following simple representation of the automaton.



(14)

In the light of the previous sections, we decompose this automaton into an algebra and a coalgebra.

First, we have the algebra $[i, \delta] : \mathbf{1} + (A \times X) \to X$ for the functor $\mathbf{1} + A \times (-)$. Recall from Exercise 3 that $(A^*, [\mathsf{nil} := \varepsilon, \mathsf{cons}])$ is initial for the functor $1 + (A \times -)$, so after currying cons, we obtain a unique morphism $r : A^* \to X$ that makes the following diagram commute.



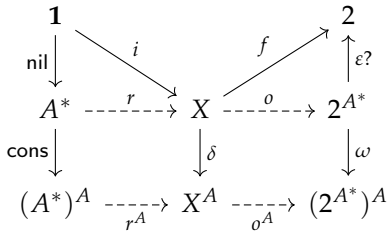
(15)

Explicitly, the commutativity of the left half of diagram 15 amounts to

$$r(\varepsilon) = i \quad \text{and} \quad r(\mathsf{cons}(w, a)) = \delta(r(w), (a)), \forall a \in A$$

We infer that for any word $w \in A^*$, $r(w)$ is the state reached by the automaton after starting in state i and reading input w.

Second, we have the coalgebra $(f, \delta) : X \to 2 \times X^A$. Recall from diagram 10 that $(2^{A^*}, (\varepsilon?, \omega))$ is final for the functor $2 \times (-)^A$. Thus, we obtain a unique morphism $o : X \to 2^{A^*}$ that completes diagram 15.



(16)

We recall from Exercise 8 that for any $x \in X$, $o(x)$ is the language in accepted by the automaton if started on state x.

Definition 36. In the setting above, the automaton (X, i, δ, f) is said to be

1. **reachable** if r is surjective,
2. **observable** if o is injective, and
3. **minimal** if it is both reachable and observable.

Exercise 13 (1pts)**.** (a) Describe, in automata theoretic terms, what reachability and observability mean.

(b) Informally explain why an automaton has a minimal number of states if and only if it is both reachable and observable.

3.2 Reversing an Automaton

We will use our new method of representing automata to give the construction of an automata which recognizes the reverse language. Morally, our procedure does the same thing as the original reversing algorithm and the powerset construction at the same time.

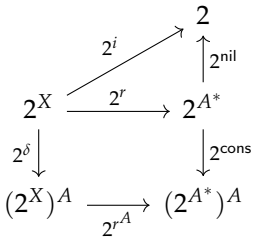
In the sequel, let $2^{(-)}$ denote the contravariant powerset functor, namely, it sends X to 2^X and $f : X \to Y$ to $2^f : 2^Y \to 2^X = S \mapsto \{x \in X : f(x) \in S\}$.

The reversed powerset construction goes like this. Given a transition function $\delta : X \to X^A$, we can uncurry it to get $\delta : X \times A \to X$, then apply $2^{(-)}$ to obtain $2^\delta : 2^X \to 2^{X \times A}$ and finally curry the elements in the codomain to obtain $2^\delta : 2^X \to (2^X)^A$. This is now a transition function for an automata whose states are set of states of the original automata.

Exercise 14 (3pts)**.** (a) Describe the action of 2^δ.

(b) Continue applying $2^{(-)}$ to the L.H.S. of diagram (16) to obtain (17). Briefly describe

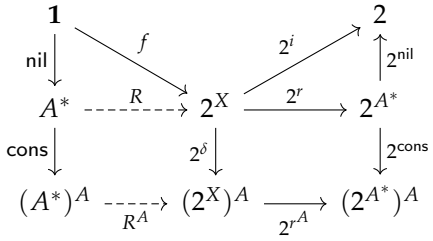
what each depicted morphism does.



(17)

Warning: For cons, you will need to use the same trick as for δ.

By currying $f : X \to 2$ to $f : \mathbf{1} \to 2^X$, we obtain an $\mathbf{1} + A \times (-)$-algebra structure on 2^X which, by initiality of A^*, gives the following diagram.



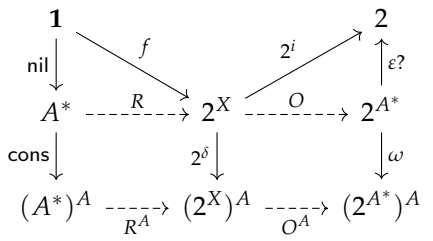
(18)

This is very close to what we are looking for:

- We have an automaton whose states are 2^X,
- its initial state is the set of states which contain at least one final state of the original automaton,
- its final states are the set of states which contain the initial state of the original automaton, and
- the morphism R tells us which states are reachable.

However, the R.H.S. of (18) is not excatly what we need to talk about observability of this automaton. You can see that $2^{\mathsf{nil}} = \varepsilon?$, but 2^{cons} is a kind of reversed version of ω, the concatenation is done in the opposite way.

Exercise 15 (1pts)**.** Describe the unique way to complete (18) into (19).



(19)

In conclusion, we have an automaton $(2^X, f, 2^\delta, 2^i)$ which is reachable if R is surjective and observable if O is injective. Moreover, we can show two imporant properties.

Exercise 16 (4pts)**.** (a) If (X, i, δ, f) recognizes the language L, then $(2^X, f, 2^\delta, 2^i)$ recognizes the reverse of L.

(b) If (X, i, δ, f) is reachable, then $(2^X, f, 2^\delta, 2^i)$ is observable.

3.3 Correctness of the Algorithm

Exercise 17 (1pt)**.** Show that Brzozowski's algorithm is correct. Namely, if (X, i, δ, f) recognizes a language L, then

- applying the (new) reverse construction,
- keeping only the reachable states,
- applying the reverse construction again, and
- keeping only the reachable states

yields an automaton which is minimal and recognizes the language L.