Notes on understanding Brzozowski's algorithm

Prakash Panangaden

29th September 2015

1 Introduction

In the 1960's Brzozowski invented a remarkable algorithm for minimizing DFAs [Brz62, Brz64] These notes are a watered-down version of a paper by 6 authors [BBH+14] on a categorical way of understanding the original algorithm. Our paper uses coalgebras and algebras and emphasizes the rôle of duality. In these notes I will give a presentation that is suitable for undergraduates; it does not use any category theory, just simple facts about sets but the essential ideas are lurking just beneath the surface.

First I will find it useful to introduce some notation. Suppose that I have sets X, Y, Z I can consider functions $f : X \times Y \longrightarrow Z$. This is a function that takes as input two arguments, one from X and one from Y and produces a result in Z. Any such function can be viewed as a function from X to *the set of functions from* Y to Z. I write the latter set as Z^Y. Thus I have a bijection between $[X \times Y \longrightarrow Z]$ and $[X \longrightarrow Z^Y]$. I will switch back and forth between whichever form is more convenient for me. For example, the transition function of a DFA is $\delta : S \times \Sigma \longrightarrow S$. I can equally well write this as $\delta : S \longrightarrow S^\Sigma$. Another piece of notation, which seems silly at first, but is very useful. Instead of naming a specific element in a set, for example I say that I have a special "start" state $s_0 \in S$, I say I have a function from any one-element set to S. I call our one-element set $\mathbf{1}$ (I absolutely do not care what the one element is, it could be 0 or $\star$ or $\bullet$). It just allows me to express everything in terms of diagrams of functions. Finally, instead of naming a subset of a set, for example $F \subseteq S$, I can define a function from S to a *two-element* set, say $\{0, 1\}$, I call it $\mathbf{2}$. The members of F are mapped to 1 and the elements of S that are not in F are mapped to 0. I will write $i : \mathbf{1} \longrightarrow S$ instead of saying $s_0 \in S$ and I will write $f : S \longrightarrow \mathbf{2}$ instead of saying there is a subset F of S.

2 Automata

Let $\mathbf{1} = \{0\}$, $\mathbf{2} = \{0,1\}$ and let Σ be any set. A deterministic automaton with inputs from Σ is given by the following data:

$$\begin{tikzcd}
\mathbf{1} \arrow[dr, "i"] & & \mathbf{2} \\
 & S \arrow[ur, "f"] \arrow[d, "\delta"] & \\
 & S^{\Sigma} &
\end{tikzcd} \tag{1}$$

That is, a set S of states, a transition function $\delta\colon S \longrightarrow S^{\Sigma}$ mapping each state $s \in S$ to a function $\delta(s)\colon \Sigma \longrightarrow S$ that sends an input symbol $a \in \Sigma$ to a state $\delta(s)(a)$, an initial state $s_0 \in S$ (formally denoted by a function $i\colon \mathbf{1} \longrightarrow S$), and a set of final (or accepting) states given by a function $f\colon S \longrightarrow \mathbf{2}$, sending a state to 1 if it is final and to 0 if it is not.

I introduce *reachability* and *observability* of deterministic automata by means of the following diagram:

$$\begin{tikzcd}
\mathbf{1} \arrow[d, "\varepsilon"'] \arrow[drr, "i"] & & & & \mathbf{2} \\
\Sigma^* \arrow[rr, dashed, "r"] \arrow[d, "\alpha"'] & & S \arrow[urr, "f"] \arrow[rr, dashed, "o"] \arrow[d, "\delta"] & & 2^{\Sigma^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(\Sigma^*)^{\Sigma} \arrow[rr, dashed, "r^{\Sigma}"'] & & S^{\Sigma} \arrow[rr, dashed, "o^{\Sigma}"'] & & (2^{\Sigma^*})^{\Sigma}
\end{tikzcd} \tag{2}$$

in the middle of which I have our automaton (S, δ, i, f). Don't be alarmed by this picture; I will explain its components.

Now I am going to introduce two automata that are deterministic but not finite at all. They are very special and will have what are called "universal" properties. On the left, I have the set Σ^* of all words over Σ. I view this as an automaton with the empty word ε as initial state and with transition function

$$\alpha\colon \Sigma^* \longrightarrow (\Sigma^*)^{\Sigma} \qquad \alpha(w)(a) = w \cdot a.$$

On the right, I have the set 2^{Σ^*} of all languages over Σ, also viewed as an automaton. The transition function of this automaton is

$$\beta\colon 2^{\Sigma^*} \longrightarrow (2^{\Sigma^*})^{\Sigma} \qquad \beta(L)(a) = \{w \in \Sigma^* \mid a \cdot w \in L\} \tag{3}$$

where $\beta(L)(a)$ is the so-called *(left) a-derivative of the language L*; and a final state function

$$\varepsilon?\colon 2^{\Sigma^*} \longrightarrow 2$$

that maps a language to 1 if it contains the empty word, and to 0 if it does not. Notice that in the automaton on the left I do not care about a final state, only the initial state matters; this makes sense from the point of view of reachability, I only care where I can get to from the initial state. Similarly for the automaton on the right, I do not care about the initial state, only about the final state.

Horizontally, I have functions r and o that I will introduce next. First recall I defined δ^* inductively. I will use a slightly different notation here for the same thing: $x_w \in S$, for $x \in S$ and $w \in \Sigma^*$, inductively by

$$x_\varepsilon = x \qquad x_{w\cdot a} = \delta(x_w)(a)$$

i.e., x_w is the state reached from x by inputting (all the letters of) the word w. In other words x_w is what I called $\delta(x, w)$ before. With this notation, I now define

$$r\colon \Sigma^* \longrightarrow X \qquad r(w) = i_w$$

and

$$o\colon X \longrightarrow 2^{\Sigma^*} \qquad o(x)(w) = f(x_w)$$

Thus r sends a word w to the state i_w that is reached from the initial state $i \in X$ by inputting all the letters of the word w; and o sends a state x to the language it accepts. That is, switching freely between languages as maps and languages as subsets,

$$o(x) = \{w \in \Sigma^* \mid f(x_w) = 1\} \tag{4}$$

I think of $o(x)$ as the *behavior* of the state x.

Now I need to explain the bottom row and I have to motivate why we are drawing these schematic diagrams with a lot of functions in them. Before I explain the bottom row I will discuss the diagram. If you look at the picture you see that there are paths that you get by following the arrows.

Each arrow is labelled by a function. *A path is labelled by the composition of the functions* labelling the arrows in the path. Now the main point: it **may** happen that for any two vertices the composed functions that you get along *any two different paths are equal*; if that is the case I say that the diagram *commutes*. If you have a commuting diagram you have written a whole slew of equations in a compact and visually appealing way. More importantly, you can "paste" commuting diagrams together and prove new equations. A lot of equational reasoning can be done very simply this way: it has come to be called "diagram chasing." We will do a bit of diagram chasing in these notes. It is very important to understand that just drawing a picture does not prove anything; you still have to prove that the diagram commutes. The diagram states a lot of equations; by itself it does not prove anything.

At the bottom of the diagram 2, I use the following notation. Suppose I have a function $f\colon V \longrightarrow W$, I write

$$f^A\colon V^A \longrightarrow W^A$$

to denote the function defined by $f^A(\phi)(a) = f(\phi(a))$, for $\phi\colon A \longrightarrow V$ and $a \in A$. Now I claim that the diagram 2 *does commute*. I call the functions r and o *homomorphisms* to signify that they make the triangles and squares of diagram (2) commute.

Now let me try to convince you that the diagram commutes. I will only look at the left hand triangle and square. I define r inductively: first I *define* $r(\varepsilon) = i$[1]. This makes the triangle commute. Then I set $r(w \cdot a) = \delta(r(w))(a)$. Let us verify that the square commutes. Now $r^{\Sigma}(\alpha(w))(a) = r(\alpha(w)(a)) = r(w \cdot a)$; this is the path obtained by going down and then right. Going right gives $r(w)$ and then going down gives $\delta(r(w))(a)$; but this is exactly what I defined $r(w \cdot a)$ to be. Similarly we can check that the other triangle and square commutes.

One can readily see that the function r is uniquely determined by the functions i and δ; similarly, the function o is uniquely determined by the functions δ and f.

Having explained diagram (2), I can now give the following definition.

Definition 1 (reachability, observability, minimality)**.** A deterministic automaton (S, δ, i, f) is *reachable* if r is surjective, it is *observable* if o is injective, and it is *minimal* if it is both reachable and observable.

[1] I am using i for the start state as well as for the function that designates the start state.

Thus (S, δ, i, f) is reachable if all states are reachable from the initial state, that is, for every $s \in S$ there exists a word $w \in \Sigma^*$ such that $i_w = s$; and (S, δ, i, f) is observable if different states recognize different languages, in other words, if they have different observable behavior. This explains the use of the word "observable", namely, an automaton is observable if its states can be unambiguously identified with their observable behaviour. Note that our definition of a minimal automaton coincides with the standard one.

3 Constructing the reverse of an automaton

Next I show that by reversing the transitions, and by swapping the initial and final states of a deterministic automaton, one obtains a new automaton accepting the reversed language. By construction, this automaton will again be deterministic. Moreover, if the original automaton is reachable, the resulting one is observable.

Our construction will make use of the following operation:

$$\begin{tikzcd} & V \arrow[d, "f"'] & & 2^V \\ 2^{(-)} : & W & \mapsto & 2^W \arrow[u, "2^f"'] \end{tikzcd}$$

which is defined, for a set V, by $2^V = \{S \mid S \subseteq V\}$ and, for $f \colon V \longrightarrow W$ and $S \subseteq W$, by

$$2^f \; : \; 2^W \longrightarrow 2^V \qquad 2^f(S) = \{v \in V \mid \; f(v) \in S\}$$

The main construction Given the transition function $\delta \colon S \longrightarrow S^\Sigma$ of our deterministic automaton, I apply, from left to right, the following three transformations:

$$\begin{tikzcd} X \arrow[d, "\delta"'] & \| & S \times \Sigma \arrow[d] & | & 2^{S\times\Sigma} & \| & (2^S)^\Sigma \\ S^\Sigma & \| & S & | & 2^S \arrow[u] & \| & 2^S \arrow[u, "2^\delta"'] \end{tikzcd}$$

The single, vertical line in the middle corresponds to an application of the operation $2^{(-)}$ introduced above. The double lines, on the left and on the right, indicate isomorphisms that are based on the operations of *currying* and *uncurrying*. The end result consists of a new set of states 2^S together with a new transition function (which I denote by 2^δ)

$$2^\delta : \; 2^S \to (2^S)^\Sigma \qquad 2^\delta(X)(a) = \{x \in S \mid \delta(s)(a) \in X\}$$

which maps any subset $X \subseteq S$, for any $a \in A$, to the set of all its a-predecessors.

Note that our construction does two things at the same time: it reverses the transitions and yields a deterministic automaton.

Initial becomes final Applying the operation $2^{(-)}$ to the initial state (function) of our automaton S gives

$$\begin{tikzcd} 1 \arrow[d, "i"'] & 2 \\ S & 2^S \arrow[u, "2^i"'] \end{tikzcd}$$

(where I write 2 for 2^1), by which I have transformed the initial state i into a final state function 2^i for the new automaton 2^S. I note that according to this new function 2^i, a subset $X \subseteq S$ is final (that is, is mapped to 1) precisely when $i \in X$.

Reachable becomes observable Next I apply the above construction(s) to the entire left hand-side of diagram (2), that is, to both δ and i and to α and ε, as well as to the functions r and r^Σ. This yields the following commuting diagram:

$$\begin{tikzcd} & 2 \\ 2^S \arrow[ur, "2^i"] \arrow[r, "2^r"] \arrow[d, "2^t"'] & 2^{\Sigma^*} \arrow[u, "2^\varepsilon"'] \arrow[d, "2^\alpha"] \\ (2^S)^\Sigma \arrow[r, "2^{r^\Sigma}"'] & (2^{\Sigma^*})^\Sigma \end{tikzcd} \qquad (5)$$

Note that for any language $L \in 2^{\Sigma^*}$, I have $2^\varepsilon(L) = \varepsilon?(L)$ and, for any $a \in A$,

$$2^\alpha(L)(a) = \{w \in \Sigma^* \mid w \cdot a \in L\}$$

The latter resembles the definition of $\beta(L)(a)$ but it is different in that it uses $w \cdot a$ instead of $a \cdot w$. By the universal property (of finality) of the triple $(2^{\Sigma^*}, \beta, \varepsilon?)$, there exists a unique homomorphism $rev\colon 2^{\Sigma^*} \longrightarrow 2^{\Sigma^*}$ as shown here

$$\begin{tikzcd}
 & 2 \\
2^{\Sigma^*} \arrow[ur, "2^{\varepsilon}"] \arrow[r, dashed, "rev"] \arrow[d, "2^{\alpha}"'] & 2^{\Sigma^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(2^{\Sigma^*})^{\Sigma} \arrow[r, dashed, "rev^{\Sigma}"'] & (2^{\Sigma^*})^{\Sigma}
\end{tikzcd} \tag{6}$$

which sends a language L to its reverse

$$rev(L) = \{w \in \Sigma^* \mid w^R \in L\}$$

where w^R is the reverse of w.

Combining diagrams (5) and (6) yields the following commuting diagram:

$$\begin{tikzcd}
 & & 2 \\
2^S \arrow[urr, "2^i"] \arrow[r, "2^r"'] \arrow[d, "2^t"'] & 2^{\Sigma^*} \arrow[ur, "2^{\varepsilon}"] \arrow[r, "rev"'] \arrow[d, "2^{\alpha}"] & 2^{\Sigma^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(2^S)^{\Sigma} \arrow[r, "2^{r^{\Sigma}}"'] & (2^{\Sigma^*})^{\Sigma} \arrow[r, "rev^{\Sigma}"'] & (2^{\Sigma^*})^{\Sigma}
\end{tikzcd}$$

Thus we see that the composition of *rev* and 2^r (is the unique function that) makes the following diagram commute:

$$\begin{tikzcd}
 & 2 \\
2^S \arrow[ur, "2^i"] \arrow[r, dashed, "O"] \arrow[d, "2^t"'] & 2^{\Sigma^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(2^S)^{\Sigma} \arrow[r, dashed, "O^{\Sigma}"'] & (2^{\Sigma^*})^{\Sigma}
\end{tikzcd} \qquad O = rev \circ 2^r \tag{7}$$

One can easily show that it satisfies, for any $S \subseteq X$,

$$O(S) = \{w^R \in \Sigma^* \mid i_w \in S\} \tag{8}$$

Final becomes initial The following bijective correspondence

$$\begin{tikzcd} 2 & 1 \arrow[d, "f"] \\ X \arrow[u, "f"] & 2^S \end{tikzcd}$$

(again an instance of currying) transforms the final state function f of the original automaton X into an initial state function of our new automaton 2^S, which I denote again by f. It will induce, by the universal property of $(\Sigma^*, \varepsilon, \alpha)$, a unique homomorphism R as follows:

$$\begin{tikzcd} 1 \arrow[d, "\varepsilon"'] \arrow[dr, "f"] & \\ \Sigma^* \arrow[r, dashed, "R"] \arrow[d, "\alpha"'] & 2^S \arrow[d, "2^t"] \\ (\Sigma^*)^\Sigma \arrow[r, dashed, "R^\Sigma"'] & (2^S)^\Sigma \end{tikzcd} \tag{9}$$

Putting everything together By now, I have obtained the following, new deterministic automaton:

$$\begin{tikzcd} 1 \arrow[d, "\varepsilon"'] \arrow[dr, "f"] & & 2 \\ \Sigma^* \arrow[r, dashed, "R"'] \arrow[d, "\alpha"'] & 2^S \arrow[ur, "2^i"] \arrow[r, dashed, "O"'] \arrow[d, "2^t"] & 2^{\Sigma^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\ (\Sigma^*)^\Sigma \arrow[r, dashed, "R^\Sigma"'] & (2^S)^\Sigma \arrow[r, dashed, "O^\Sigma"'] & (2^{\Sigma^*})^\Sigma \end{tikzcd} \tag{10}$$

where the above diagram is simply the combination of diagrams (9) and (7) above.

Theorem 2. Let (S, δ, i, f) be a deterministic automaton and let $(2^S, 2^\delta, f, 2^i)$ be the reversed deterministic automaton constructed as above.

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

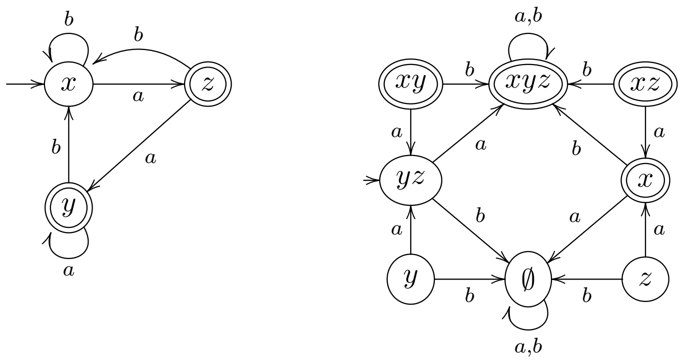
2. If (S, δ, i, f) accepts the language L, then $(2^S, 2^\delta, f, 2^i)$ accepts $rev(L)$.

Proof As the operation $2^{(-)}$ transforms surjections into injections (and since *rev* is a bijection), reachability of (S, δ, i, f) implies observability of $(2^S, 2^\delta, f, 2^i)$. The second statement follows from the fact that we have

$$\begin{aligned} O(f) &= \{w \in \Sigma^* \mid 2^i(f_w) = 1\} \\ &= \{w^R \in \Sigma^* \mid i_w \in f\} \qquad \text{[by identity (8)]} \\ &= rev(\{w \in \Sigma^* \mid i_w \in f\}) \\ &= rev(o(i)) \end{aligned}$$

■

I consider the following two automata. In the picture below, an arrow points to the initial state and a double circle indicates that a state is final:



(11)

The automaton on the left is reachable (but not observable, since y and z accept the same language $\{a, b\}^*a + 1$). Applying our construction above yields the automaton on the right, which is observable (all the states accept different languages) but not reachable (e.g., the state $\{x, y\}$, denoted by xy, is not reachable from the initial state $\{y, z\}$). Furthermore, the language accepted by the automaton on the right, $a\{a, b\}^*$, is the reverse of the language accepted by the automaton on the left, which is $\{a, b\}^*a$.

4 Brzozowski's algorithm

As an immediate consequence, I obtain the following version of Brzozowski's algorithm.

Corollary 3. Given a deterministic automaton accepting a language L,

1. reverse and determinize it (as in the construction above),
2. take its reachable part,
3. reverse and determinize it (as in the construction above),

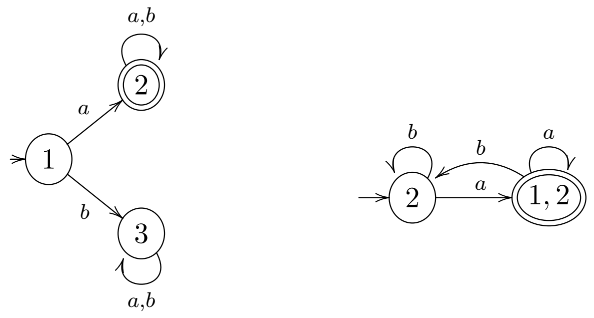
4. take its reachable part.

The resulting automaton is the minimal automaton accepting L.

Proof The automaton obtained after steps (1) and (2) is reachable and accepts $rev(L)$. After step (3), the automaton is observable and accepts $rev(rev(L)) = L$. Finally, taking reachability in step (4) yields a minimal automaton accepting L. ∎

Note that in Corollary 3, if the original automaton is already reachable, then the automaton obtained after step (2) is a minimal automaton accepting $rev(L)$.

We saw that applying our construction (step (1)) to the left automaton in (11) resulted in the automaton on the right in (11). By taking the reachable part of the latter (step (2)), we obtain the automaton depicted below on the left (where $1 = \{y, z\}$, $2 = \{x, y, z\}$ and $3 = \emptyset$):



(12)

The automaton on the right in (12) is obtained by, once more, reversing-determinizing (step (3)) and taking the reachable part (step (4)). It is the minimization of the automaton we started with.

References

[BBH+14] Filippo Bonchi, Marcello M. Bonsangue, Helle Hvid Hansen, Prakash Panangaden, Jan Rutten, and Alexandra Silva. Algebra-coalgebra duality in Brzozowski's minimization algorithm. *ACM Transactions on Computational Logic*, 2014.

[Brz62] Janusz A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In J. Fox, editor, *Proceedings* *of the Symposium on Mathematical Theory of Automata*, number 12 in MRI Symposia Series, pages 529–561. Polytechnic Press of the Polytechnic Institute of Brooklyn, April 1962. Book appeared in 1963.

[Brz64] Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964.