Brzozowski's algorithm (co)algebraically

F. Bonchi[1], M.M. Bonsangue[2,3], J.J.M.M. Rutten[3,4], A.M. Silva[3,4,5]

[1] ENS Lyon, Université de Lyon
[2] LIACS - Leiden University
[3] Radboud University Nijmegen
[4] Centrum Wiskunde & Informatica
[5] Universidade do Minho

Abstract. We give a new presentation of Brzozowski's algorithm to minimize finite automata, using elementary facts from universal algebra and coalgebra, and building on earlier work by Arbib and Manes on the duality between reachability and observability. This leads to a simple proof of its correctness and opens the door to further generalizations.

1 Introduction

Brzozowski's algorithm [6] is a somewhat unusual recipe for minimizing finite state automata: starting with a (possibly non-deterministic) automaton, one reverses its transitions, makes it deterministic, takes the part that is reachable, and then repeats all of this once more. The result will be a deterministic automaton that is the minimization of the original one.

Though an elementary description and correctness proof of the algorithm is not very difficult (see for instance [13, Cor. 3.14]), the algorithm comes to most as a bit of a surprise. Here we try to add to its understanding by presenting a proof that is based on a result by Arbib and Manes [2, 3] on the duality between *reachability* and *observability* (the latter is another word for minimality).

We will first present a reformulation of Arbib and Manes' duality result in terms of a bit of elementary algebra and coalgebra. These are the natural mathematical settings for the modelling of reachability and observability, respectively. Ultimately, their duality is due to the fact that the transitions of an automaton X (with input alphabet A) can be modelled both algebraically, as a function of type $X \times A \to X$ and coalgebraically, as a function of type $X \to X^A$. Next, we will derive (the correctness of) Brzozowski's algorithm as a corollary from this duality.

Our reasons for giving this new formulation of Brzozowski's algorithm are the following.

First, the duality between reachability and observability, on which we will base our proof, is in itself a very beautiful result that, unfortunately, it is not very well-known. The original proof by Arbib and Manes uses some category theory that makes it for many difficult to understand. Although our proof of this duality is in essence categorical as well, we have formulated it in elementary terms, using only the notions of sets and functions. As a result, the present paper should be understandable to anyone with a very basic understanding of automata.

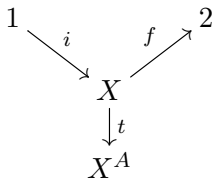
Secondly, Brzozowski's algorithm follows as an immediate corollary of this (newly formulated) duality result. This observation gives a new way of understanding the algorithm and makes a formal proof of its correction very easy.

Thirdly, we expect that our proof of Brozowski's algorithm is easy to generalise. The present paper contains the straightforward generalisation of the algorithm to Moore automata. We mention further applications, to weighted and to probabilistic automata, as future work.

Acknowledgements: Our interest in Brozowski's algorithm was raised by Prakash Panangaden, who asked himself (and several others) how Brzozowski's algorithm could be generalised to probabilistic automata. Jan Rutten and Alexandra Silva gratefully acknowledge several discussions with Prakash on this (and many other) subject(s) .

2 Rechability and observability

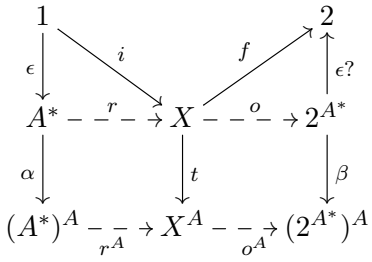
Let $1 = \{0\}$, $2 = \{0, 1\}$ and let A be any set. A deterministic automaton with inputs from A is given by the following data:



(1)

That is: a set X of states; a transition function $t \colon X \to X^A$ mapping each state $x \in X$ to a function $t(x) \colon A \to X$ that sends an input symbol $a \in A$ to a state $t(x)(a)$; an initial state $i \in X$ (formally denoted by a function $i \colon 1 \to X$); and a set of final (or accepting) states given by a function $f \colon X \to 2$, sending a state to 1 if it is final and to 0 if it is not.

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



(2)

in the middle of which we have our automaton X.

On the left, we have the set A^* of all words over A, with the empty word ϵ as initial state and with transition function

$$\alpha : A^* \to (A^*)^A \qquad \alpha(w)(a) = w \cdot a$$

On the right, we have the set 2^{A^*} of all languages over A, with transition function

$$\beta : 2^* \to (2^{A^*})^A \qquad \beta(L)(a) = \{w \in A^* \mid a \cdot w \in L\}$$

and a final state function

$$\epsilon? : 2^{A^*} \to 2$$

that maps a language to 1 if it contains the empty word, and to 0 if it does not.

Horizontally, we have functions r and o that we will introduce next. First we define x_w, for $x \in X$ and $w \in A^*$, inductively by

$$x_\epsilon = x \qquad x_{w \cdot a} = t(x_w)(a)$$

With this notation, we now define

$$r : A^* \to X \qquad r(w) = i_w$$

and

$$o : X \to 2^{A^*} \qquad o(x)(w) = f(x_w)$$

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 send a state x to the language it accepts. That is, switching freely between languages as maps and languages as subsets,

$$o(x) = \{w \in A^* \mid f(x_w) = 1\} \tag{3}$$

We think of $o(x)$ as the semantics or the *behavior* of the state x.

The functions r and o are homomorphisms in the precise sense that they make the triangles and squares of diagram (2) commute. In order to understand the latter, we note that at the bottom of the diagram, we use, for $f : V \to W$, the notation

$$f^A : V^A \to W^A$$

to denote the function defined by $f^A(\phi)(a) = f(\phi(a))$, for $\phi : A \to V$ and $a \in A$.

One can readily see that the function r is uniquely determined by the functions i and t; similarly, the function o is uniquely determined by the functions t and f. In categorical terms, the unique existence of r is a consequence of A^* being an initial algebra of the functor $1 + (A \times -)$; similarly, the unique existence of o rests on the fact that 2^{A^*} is a final coalgebra of the functor $2 \times (-)^A$.

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

Definition 1 (reachability and observability). *A deterministic automaton X is* reachable *if r is surjective. It is* observable *if o is injective.*

Thus X is reachable if all states are reachable from the initial state: for every $x \in X$ there exists a word $w \in A^*$ such that $i_w = x$. And X is observable if different states recognize different languages or, in other words, if they have different observable behavior. We note that an observable automaton is also *minimal*: it does not contain any pair of (language) equivalent states. In what follows, we shall therefore use the words observable and minimal as synonyms.

3 Constructing the reverse of an automaton

Next we 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 minimal.

Our construction will make use the following operation:

$$2^{(-)} : \quad \begin{array}{c} V \\ \downarrow f \\ W \end{array} \quad \mapsto \quad \begin{array}{c} 2^V \\ \uparrow 2^f \\ 2^W \end{array}$$

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

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

(In categorical terms, this is the contravariant powerset functor.)

The main construction: Given the transition function $t\colon X \to X^A$ of our deterministic automaton, we apply, from left to right, the following three transformations:

$$\begin{array}{c|c|c|c} X & X \times A & 2^{X\times A} & (2^X)^A \\ {\scriptstyle t}\downarrow & \downarrow & \uparrow & \uparrow{\scriptstyle 2^t} \\ X^A & X & 2^X & 2^X \end{array}$$

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^X together with a new transition function

$$2^t :\ 2^X \to (2^X)^A \qquad 2^t(S)(a) = \{x \in X \mid t(x)(a) \in S\}$$

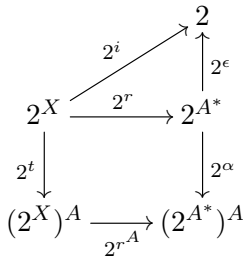
which maps any subset $S \subseteq X$, for any $a \in A$, to the set of all its a-predecessors. Note that our construction does two things at the same time: it reverses the transitions (as we shall see formally later) and yields again a deterministic automaton.

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

$$\begin{array}{c|c} 1 & 2 \\ {\scriptstyle i}\downarrow & \uparrow{\scriptstyle 2^i} \\ X & 2^X \end{array}$$

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

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

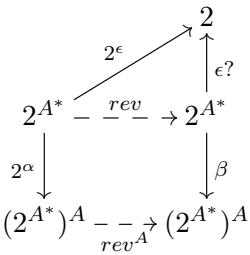


(4)

We note that for any language $L \in 2^{A^*}$, we have $2^\epsilon(L) = \epsilon?(L)$ and, for any $a \in A$,

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

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 $(2^{A^*}, \beta, \epsilon?)$, there exists a unique homomorphism



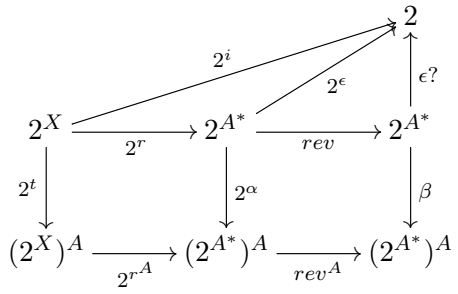
(5)

which sends a language L to its reverse

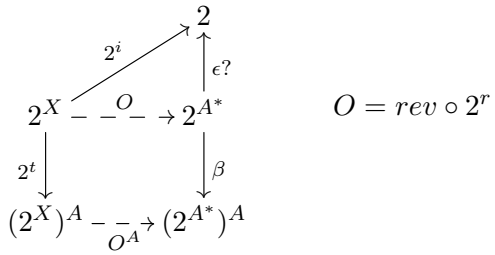
$$rev(L) = \{w \in A^* \mid w^R \in L\}$$

where w^R is the reverse of w.

Combining diagrams (4) and (5) yields the following commuting diagram:



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

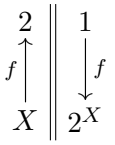


$$O = rev \circ 2^r \tag{6}$$

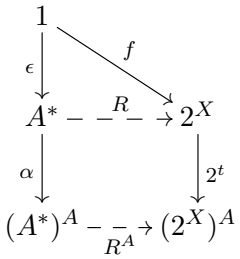
One can easily show that it satisfies, for any $S \subseteq X$,

$$O(S) = \{w^R \in A^* \mid i_w \in S\} \tag{7}$$

Final becomes initial: The following bijective correspondence

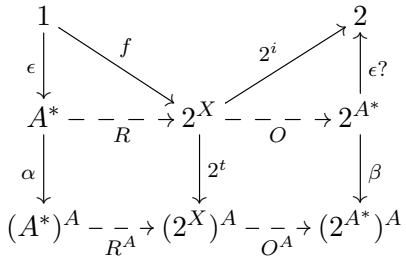


(again an instance of currying) transforms the final state function f of the original automaton X into an initial state function of our new automaton 2^X, which we denote again by f. It will induce, by the universal property of (A^*, ϵ, α), a unique homomorphism as follows:



(8)

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



(9)

where the above diagram is simply the combination of diagrams (8) and (6) above.

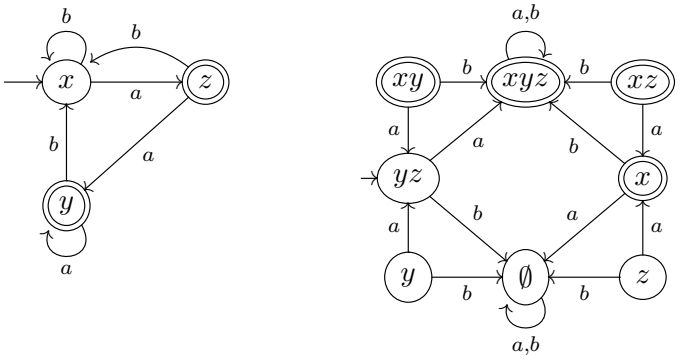
Theorem 2. *If the original automaton X is reachable, that is, if r is surjective, then the new automaton 2^X is observable, that is, O is injective. Furthermore, the language accepted by the initial state f of the new automaton 2^X is the reverse of the language accepted by the initial state i of the new automaton 2^X.*

Proof. As the operation $2^{(-)}$ transforms surjections into injections, reachability of X implies observability of 2^X. The second statement follows from the fact that we have

$$\begin{aligned} O(f) &= \{w \in A^* \mid 2^i(f_w) = 1\} \\ &= \{w^R \in A^* \mid i_w \in f\} \qquad \text{[by identity (7)]} \\ &= rev(\{w \in A^* \mid i_w \in f\}) \\ &= rev(o(i)) \end{aligned}$$

□

Example 3. We 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:



(10)

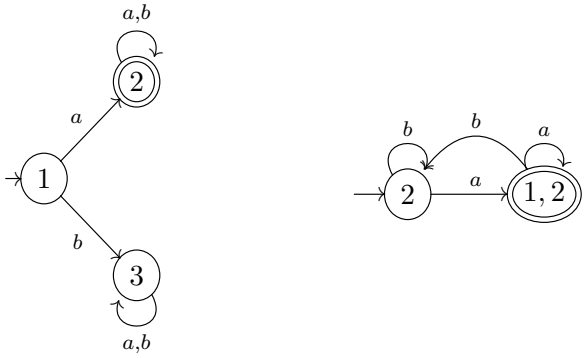
The automaton on the left is reachable (but not observable, since y and z accept the same language $\{a, b\}^*a + 1$). Applying our construction above yields the automaton on the right, which is observable (all the states accept different languages) but not reachable (e.g., the state $\{x, y\}$, denoted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

4 Brzozowski's algorithm

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

Corollary 4. *Applying the above construction to a deterministic and reachable automaton accepting a language L yields a minimal automaton accepting $rev(L)$. Taking of the latter automaton the reachable part, and applying the same procedure again yields a minimal automaton accepting L.*

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



(11)

The automaton on the right is obtained by applying our construction once more. It is the minimization of the automaton we started with. □

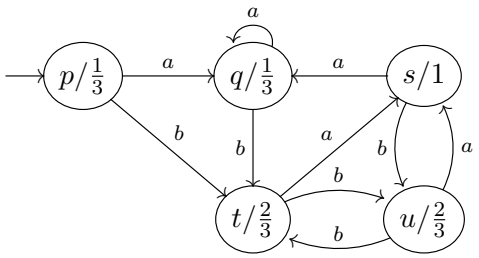
5 Moore automata

Moore automata generalise deterministic automata by allowing outputs in an arbitrary set B, rather than just 2. Formally, a Moore automaton with inputs in A and outputs in B consists of a set of states X, an initial state $i\colon 1 \to X$, a transition function $t\colon X \to X^A$ and an output function $f\colon X \to B$. Moore automata accept functions in B^{A^*} (that is functions $\phi\colon A^* \to B$) instead of languages in 2^{A^*}.

Here is in a nutshell how our story above can be generalised to Moore automata. We can redraw diagram (2) by simply replacing 2 with B. We then define reachability and observability as before. Next we adopt our

procedure of reversing transitions by using (the contra-variant functor) $B^{(-)}$ instead of $2^{(-)}$: for all sets V, $B^V = \{\phi\colon V \to B\}$ and, for all functions $g\colon V \to W$, the function $B^g\colon B^W \to B^V$ maps each $\phi \in B^W$ to $B^g(\phi) = \phi \circ g$. Finally, all the results discussed above will also hold for Moore automata. The next example illustrates the minimization of a Moore automaton.

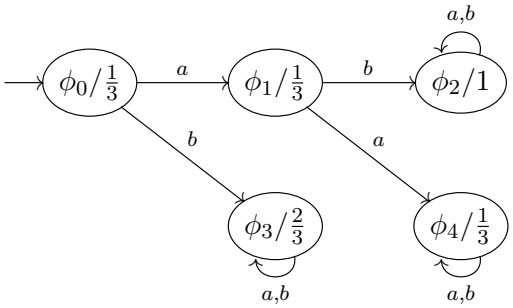
Example 5. We consider the following Moore automaton with inputs in $A = \{a, b\}$ and output in the set of real numbers $\mathbb{R}$. In the picture below, the output value r of a state s is indicated inside the circle by s/r:



(12)

The automaton accepts a function in $\mathbb{R}^{A^*}$ mapping every word w ending with ba to 1, every word ending with b to $\frac{2}{3}$ and every other word to $\frac{1}{3}$. Clearly the automaton is reachable from p. However it is not observable, since, for example, the states p and q accept the the same function.

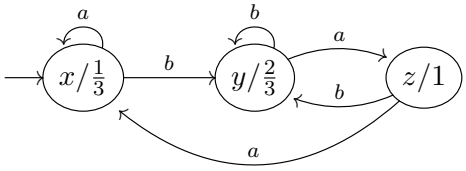
Applying our construction above yields a Moore automaton with $\mathbb{R}^S$ as set of states, where $S = \{p, q, s, t, u\}$ is the set of states of the original automaton. The output value of a state $\phi\colon S \to \mathbb{R}$ is given by $\phi(p)$, where p is the initial state of the original automaton. Further, the output function of the original automaton becomes the new initial state, i.e., the function $\phi_0\colon S \to \mathbb{R}$ mapping p and q to $\frac{1}{3}$, t and u to $\frac{2}{3}$, and u to 1. By the above results, it follows that the automaton is observable. It is not reachable, as it contains infinitely many states, and there are only five states reachable from ϕ_0, as we can see in the picture below.



(13)

We do not spell out the full definition of the above states. As an example, the state ϕ_1 consists of the map assigning p, q and s to $\frac{1}{3}$, and t, u to 1. Note that the function in $\mathbb{R}^{A^*}$ accepted by the state ϕ_0 maps each words $w \in \{a, b\}^*$ to the same value where the reverse word w^R is mapped by the function accepted by the original automaton in Figure 12. More formally, it maps words which begin with ab to 1, words which begin with b to $\frac{2}{3}$, and all other words to $\frac{1}{3}$.

If we repeat the same construction one more time, and take the reachable automaton from the initial state we obtain the minimal Moore automaton equivalent to the one in Figure 12:



(14)

□

6 Discussion

We have given a new description and a new correctness proof of Brzozowski's algorithm [6] for the minimization of deterministic automata. Next we have shown how to generalise the algorithm to deterministic Moore automata.

There are several other, sometimes more efficient minimization algorithms for finite automata [8]. But even though the complexity of Brzozowski's algorithm is double exponential in the worst case (essentially because it applies the contravariant powerset construction twice), the algorithm has been shown to behave rather well in practice, at least for non-deterministic automata [16].

Our proof of correctness of the algorithm is based on the duality between reachability and observability, which is due to Arbib and Manes [2, 3]. Our formulation uses, albeit implicitly, a bit of (standard) universal algebra and coalgebra [12]. In particular we have given a simple proof that the resulting automaton is minimal, where minimality means that the automaton does not contain two different states with the same observable behaviour. Our method can be applied also to automata with infinitely many states.

The duality between reachability and observability has been studied also in other contexts. In [4], this duality is used to establish several analo-

gies between concepts from observational (coalgebraic) and constructor-based (algebraic) specifications. In the present paper, we have highlighted how Brzozowski's algorithm integrates both concepts, for the specific case of deterministic and Moore automata.

Yet another approach, but somewhat similar in spirit to ours, can be found in [5], where a Stone-like duality between automata and their logical characterization is taken as a basis for Brzozowski's algorithm. The precise connection between that approach and the present paper, remains to be better understood. More generally, it is a challenge to try and generalise Brzozowski's algorithm to various other types of coalgebras.

Crucial for our approach was the combined use of both algebra and coalgebra. Notably, it was important to include in the definition of automaton an initial state, which is in essence an algebraic concept. In coalgebra, one typically models automata without initial states. In that respect, so-called well-pointed coalgebras [1], which are coalgebras with a designated initial state, may have some relevance for the further generalization of Brzozowski's algorithm.

A somewhat different notion of nondeterministic Moore automata has recently been introduced in [7]. It basically consists of a nondeterministic automaton with output in a set that comes equipped with a commutative and associative operator. Interesting for our context is their variant of Brzozowski's algorithm for the construction of a minimal deterministic Moore automaton that is equivalent to a given nondeterministic one.

Brzozowski's minimization algorithm has also been combined with Brzozowski's method for deriving deterministic automata from regular expressions in [17]. It would be useful to investigate how such a combination can be generalized to, for example, Kozen's calculus of Kleene algebras with tests [9]. All that is required for our approach is a deterministic Moore automaton accepting the guarded language denoted by the reverse of the input expression.

Brzozowski's algorithm was originally formulated for *nondeterministic* finite automata [6]. Our present approach (as well as that of [5]) takes a *deterministic* automaton as a starting point. Recently [15], we have presented a generalisation of the subset construction for the determinisation of many different types of automata. Examples include Rabin's probabilistic automata [11] and weighted automata [14], to which our method applies in spite of the fact that the resulting deterministic (Moore) automaton is infinite. How to minimize nondeterministic automata directly, without having to introduce an extra determinisation step, is left as future work. Also we would like to combine the results of the present paper

with those of [15] to generalise Brzozowski's algorithm to probabilistic and weighted automata.

References

1. J. Adámek, S. Milius, L. S. Moss, L. Sousa. Well-pointed Coalgebras. Unpublished note.
2. M.A. Arbib and E.G. Manes. Adjoint machines, state-behaviour machines, and duality. *Journal of Pure and Applied Algebra*, 6:313–344, 1975.
3. M.A. Arbib and E.G. Manes. Machines in a category. *Journal of Pure and Applied Algebra*, 19:9–20, 1980.
4. M. Bidoit, R. Hennicker, and A. Kurz. On the duality between observability and reachability. In *Proc. of FoSSaCS 2001*, pages 72–87, 2001.
5. N. Bezhanishvili, P. Panangaden, C. Kupke. Minimization via duality. Unpublished note.
6. J.A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical Theory of Automata*, 12(6):529–561, 1962.
7. G. Castiglione, A. Restivo, and M. Sciortino. Nondeterministic Moore Automata and Brzozowskis Algorithm In *Proc. of CIAA 2011* volume 6807 of *Lecture Notes in Computer Science*, pages 88–99, Springer, 2001.
8. D. Kozen. *Automata and Computability.* Springer-Verlag, 1997.
9. D. Kozen. Kleene algebra with tests. ACM Trans. Program. Lang. Syst., 19:427-443, 1997.
10. P. Panangaden et al. The duality of state and observations. Unpublished note.
11. M.O. Rabin. Probabilistic automata. Information and Control, 6(3):230245, 1963.
12. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. Fundamental Study.
13. J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.
14. M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245270, 1961.
15. A. Silva, F. Bonchi, M.M. Bonsangue and J.J.M.M. Rutten. Generalizing the powerset construction, coalgebraically. In *Proc. of FSTTCS 2010*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs) Series*, pages 272-283, 2010.
16. B.W. Watson. Taxonomies and Toolkits of Regular Language Algorithms. Ph.D thesis, Eindhoven University of Technology, The Netherlands, 1995.
17. B.W. Watson. Directly Constructing Minimal DFAs: Combining Two Algorithms by Brzozowski. In *Proc. of CIAA 2000* volume 2088 of *Lecture Notes in Computer Science*, pages 311–317, Springer, 2001.