García Gómez, P.; López Rodríguez, D.; Vázquez-De-Parga Andrade, M. (2014). Efficient deterministic finite automata split-minimization derived from Brzozowski's algorithm. International Journal of Foundations of Computer Science. 25(6):679-696. doi:10.1142/S0129054114500282.

International Journal of Foundations of Computer Science

EFFICIENT DETERMINISTIC FINITE AUTOMATA SPLIT-MINIMIZATION DERIVED FROM BRZOZOWSKI'S ALGORITHM

Pedro García, Damián López and Manuel Vázquez de Parga
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Camino de Vera s/n, Valencia 46022, SPAIN
{*pgarcia,dlopez,mvazquez*}*@dsic.upv.es*

Received (Day Month Year)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

Minimization of deterministic finite automata is a classic problem in Computer Science which is still studied nowadays. In this paper, we relate the different split-minimization methods proposed to date, or to be proposed, and the algorithm due to Brzozowski which has been usually set aside in any classification of DFA minimization algorithms. In our work, we first propose a polynomial minimization method derived from a paper by Champarnaud et al. We also show how the consideration of some efficiency improvements on this algorithm lead to obtain an algorithm similar to Hopcroft's classic algorithm. The results obtained lead us to propose a characterization of the set of possible splitters.

Keywords: DFA minimization; Brzozowski's algorithm; Hopcroft's algorithm

1. Introduction

Many computer applications, from text processing or image analysis to linguistics among others, consider the computation of minimal automata in order to obtain efficient solutions. The problem of automata minimization is a classic issue in Computer Science, which, still nowadays, arouses interest.

The minimization of deterministic finite automata is based on the computation of the coarsest equivalence relation which fulfills that any pair of equivalent states p and q have the same final/non-final status, and, for any given symbol, the states reached from p and q with that symbol are also equivalent. The computation of such relation is, in fact, the computation of the Nerode's equivalence relation for the language accepted by the automaton to be minimized.

The methods used to compute the above mentioned relationship usually follow one of two different approaches. On the one hand, some methods check every pair of states to test if they are equivalent or not [1, 2]. On the other hand, some other methods iteratively refine an initial partition of the set of states into final and not final states [3, 4, 5]. Among these algorithms, the algorithm by Hopcroft is of special interest, because it is the one with the best time complexity ($\mathcal{O}(kn \log n)$, where

n stands for the number of states of the input automaton and k denotes the size of the alphabet).

The minimization algorithm proposed by Brzozowski [6] is usually set appart from the rest [7, 8]. Despite its worst-case exponential time complexity, the method has a good average behaviour in the practice. Furthermore, it is a very concise and elegant algorithm based on two well-known constructions on automata, which makes its implementation very straightforward. Essentially, the algorithm computes the automaton $D(R(D(R(A))))$, where $D(A)$ denotes the determinization of A by the well-known subset construction and $R(A)$ is the reverse automaton of A. Recently, Brzozowski and Tamm proposed a general minimization by double reversal framework [9], thus, the original algorithm by Brzozowski, as well as other recent work [10], became instances within such framework.

The paper by Champarnaud et al. [7] can be seen as a first attempt to relate Brzozowski's algorithm, when applied to DFA, with other minimization methods. In order to obtain the minimal DFA equivalent to an automaton A, the algorithm proposed by Champarnaud et al. considers the states of the automaton $D(R(A))$ (which are in fact named by subsets of the states of A) in order to refine the initial partition. Thus, the algorithm proposed modifies the double reversal minimization method by substituting the second determinization of the algorithm by the computation of an equivalence relation that states any two states of A as equivalent when, for each state P of $D(R(A))$, either both states or none of them are in P. This, exponential in the worst case algorithm, is interesting because relates Brzozowski algorithm with minimization algorithms by splitting such as those by Hopcroft or Moore [11].

In this paper we relate the algorithm by Brozozowski with any split-minimization method. We first propose an improvement of the algorithm due to Champarnaud et al. based in the following fact. Given an automaton A, any succession of effective partitions of the states in A that leads to the Nerode's equivalence has a number of partitions bounded by n, the number of states of A. This implies that not every state in $D(R(A))$ (potentially 2^n states) is necessary to refine the initial partition. Our first algorithm (quadratic) is modified in order to improve its complexity, that becomes $\mathcal{O}(n \log n)$, similar to Hopcroft's complexity. The study concludes with a characterization of the whole set of valid splitters to be used by any split-minimization method.

2. Notation and definitions

Let Σ be a finite alphabet and let Σ^* be the free monoid generated by Σ with concatenation as the internal operation and the empty string λ as neutral element. For any given $x \in \Sigma^*$, we will denote x^r the reverse of x. Let us denote the size of a set Q with $|Q|$. Let us also denote by 2^Q the power set of Q.

A *finite automaton* is a 5-tuple $A = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of states, Σ is an alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of

final states and $\delta : Q \times \Sigma \to 2^Q$ is the transition function. Let us note that the transition function can also be seen as $\delta \subseteq (Q \times \Sigma \times Q)$. The transition function can be extended in a natural way to $2^Q \times \Sigma^*$.

Given an automaton A, we say it is accessible if, for each $q \in Q$, there exists a string x such that $q \in \delta(I, x)$. The right language of a state q, denoted by R_q^A or R_q when no confusion is possible, is defined as $R_q = \{x \in \Sigma^* \ : \ \delta(q, x) \cap F \neq \emptyset\}$ and the language accepted by the automaton as $L(A) = \bigcup_{q \in I} R_q$. For any automaton A, any two states p and q are defined to be equivalent according the relation $\equiv_A$ if and only if they are such that $R_p^A = R_q^A$.

An automaton is called *deterministic* (DFA) if, for every state q and every symbol a, $|\delta(q, a)|$ is at most one, and it has only one initial state usually denoted by q_0. A DFA is said to be complete whenever $|\delta(q, a)|$ is just one. In the following we will consider only complete and accessible DFA.

Given any language L, we will denote the reverse language by L^r. Given a finite automaton $A = (Q, \Sigma, \delta, I, F)$ that accepts a language L, the reverse automaton is defined as the automaton $R(A) = (Q, \Sigma, \delta_r, F, I)$, where $q \in \delta_r(p, a)$ if and only if $p \in \delta(q, a)$. Note that $L(R(A)) = L(A)^r$. For any automaton $A = (Q, \Sigma, \delta, I, F)$ it is known that the automaton $A' = (2^Q, \Sigma, \delta', I, F')$, where $F' = \{P \in 2^Q \ : \ P \cap F \neq \emptyset\}$ and $\delta'(P, a) = \cup_{p \in P} \delta(p, a)$ is a DFA equivalent to A. Let us denote the accesible version of A by $D(A)$. Whenever we will refer to a state of $D(A)$, we will usually do to the subset P of states of A that names it. For the sake of clarity, we will reduce the parenthesis to denote the composition of reverse and determinization operations, thus, for instance, we will use $DR(A)$ instead of $D(R(A))$.

A *partition* of a set Q is a set $\{P_1, P_2, \ldots, P_k\}$ of pairwise disjoint non-empty subsets of Q such that $Q = \cup_{1 \leq i \leq k} P_i$. We will refer to those subsets as *blocks*, and we will denote with $B(p, \pi)$ the block of π which contains p. A partition π_1 is refined by π_2 (π_1 is coarser than π_2) if each class in π_2 is contained in some class in π_1. We will denote this $\pi_2 \leq \pi_1$.

Let π_1 and π_2 be two partitions of Q, we will denote with $\pi_1 \wedge \pi_2$ the coarsest partition which refines both π_1 and π_2. The classes of this partition are the non empty sets in $P_1 \cap P_2$, where $P_1 \in \pi_1$ and $P_2 \in \pi_2$. In order to reduce the notation, for any $P \subseteq Q$, we will denote the complementary of P in Q by $\overline{P_Q}$, or $\overline{P}$ whenever this omission do not lead to confusion.

Given a complete DFA $A = (Q, \Sigma, \delta, q_0, F)$, let $P, R \subset Q$ and $a \in \Sigma$. Let us refer to (P, a) as a *splitter* and also denote by $(P, a)|R$ the *split* of the set R into the sets $R' = \delta^{-1}(P, a) \cap R$ and $R'' = R - R'$. It is interesting to be noted here that $(P, a)|R = (\overline{P}, a)|R$. Whenever $\delta^{-1}(P, a) \cap R = \emptyset$ or $\delta^{-1}(P, a) \cap R = R$ we will say that (P, a) does not split R and we will denote it by $(P, a)|R = R$.

3. Brzozowski's algorithm

The algorithm proposed by Brzozowski [6] computes the minimal DFA equivalent to any non-deterministic automaton $A = (Q, \Sigma, \delta, I, F)$. The process consists

in computing the automaton $DRDR(A)$. Following result is the key to prove the correctness of this algorithm.

Proposition 1 (Brzozowski) *Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$ that accepts a language L, then $DR(A)$ is the minimal DFA that accepts the language L^r.*

Let us point out briefly in other terms the reason why the second determinization of Brzozowski's algorithm effectively computes the minimum DFA. Please note that, the right language of every state P in $RDR(A)$ ($P \subseteq Q$) contains the strings that are in every right language R_q^A, where $q \in P$, and such that they are not in the right languages of states in Q not in P. Note that this implies that, for any pair of states P and P' of the automaton $RDR(A)$, the right languages $R_P^{RDR(A)}$ and $R_{P'}^{RDR(A)}$ are disjoint.

Taking this into account, the second determinization of Brzozowski's algorithm can be seen as a method to relate each state $p \in Q$ with a set of states $\mathscr{P} = \{P_1, P_2, \ldots, P_n\}$ of $RDR(A)$, in such a way that the union of the different $R_{P_i}^{RDR(A)}$, with $1 \leq 1 \leq n$, equals R_q^A. A consequence of this is that the state p is included in every $P_i \in \mathscr{P}$, and that p is not included in any other state P' of $RDR(A)$, $P' \notin \mathscr{P}$. Note that every pair of equivalent states according $\equiv_A$ (states with the same right language) will be related with the same set of states of $RDR(A)$.

In [12], and in the setting of Universal Algebra, Courcelle et al. formultate Brzozowski's algorithm, along with other operations. The algebraic framework used in that work facilitates to extend the results from words to trees.

It is worth to be noted that, when the input automaton A is non-deterministic, the computation of the classes of the relation $\equiv_A$ leads to a (non-deterministic) automaton which is a partial reduction of A. Nevertheless, when A is deterministic, the right language of the states are quotients of $L(A)$ with respect to the strings in Σ^*, and the computation of $\equiv_A$ leads to the minimum DFA for the language.

In the work by Champarnaud et al. [7] the authors use the set of states of this automaton to propose their DFA minimization algorithm. The correctness of the method is based in Proposition 2.

Proposition 2 (Champarnaud et al [7], Proposition 8)
Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts a language L and let $\mathscr{R}$ the set of states of $DR(A)$. For each pair of states $p, q \in Q$, $R_p^A = R_q^A$ if and only if, for all $P \in \mathscr{R}$, it is fulfilled that $p \in P \Leftrightarrow q \in P$.

Let us relate Proposition 2 with our comment on the second determinization of Brzozowski's minimization algorithm. Thus, note that for a given automaton A, any two states of A such that $p \equiv_A q$ are related with the same set of states of $RDR(A)$, that is, for every state $P \in DR(A)$, the state p is in P if and only if q is also in P.

Taking into account that the target is to detect the equivalent states (and therefore minimize the input automaton), there are several ways to compute this: Brzo-

zowski's algorithm is one of them; the split operation used by Champarnaud et al. obtain the same result. Other authors use this same approach for other purposes. For instance, Lombardy and Sakarovich [13], and Polak as well [14], build a matrix M with rows indexed by the states in $\mathscr{R}$ and the columns indexed by the states in Q, where, for each $(P, q) \in \mathscr{R} \times Q$:

$$M(P,q) = \begin{cases} 1 & \text{if } q \in P \\ 0 & \text{otherwise} \end{cases}$$

Taking into account the matrix, those states in Q that index equivalent columns are also equivalent. This is a direct result from the way the authors obtain the universal automaton for a given language.

It seems quite clear that both approaches can be seen as a variation of Brzozowski's algorithm, both with the same drawback, that is, their exponential time complexity in the worst case (the automaton $DR(A)$ can be exponentially bigger than A). We now prove that, taking into account any deterministic automata A, the computation of the classes of $\equiv_A$, hence the Nerode's equivalence relation $\pi_{L(A)}$ does not need the whole computation of $DR(A)$.

4. A polynomial algorithm

Let us first stress that, for any DFA $A = (Q, \Sigma, \delta, q_0, F)$ with n states, it suffices $n - 1$ splitters in the worst case, to refine the initial partition of Q into final and non-final states in order to distinguish all the states of A. Our algorithm takes this into account and carries out the minimization of an input DFA using a partial determinization of the reverse automaton, in which those states that do not refine the current partition are rejected. This method of minimization by *partial reverse determinization* (PRD) is depicted in Algorithm 4.1.

Note that the algorithm is similar to the one by Champarnaud et al. The main difference consist of line 16, in which it is checked whether the current partition has been refined or not. If so, the state of the $DR(A)$ that leads to the refinement (state $\delta^{-1}(S, a)$) is added to the waiting list $\mathscr{L}$ in order to be considered later. Note that the modification allows to greatly improve the time behaviour of the algorithm. Let us first show how the algorithm behaves in Example 3.

Example 3. *Let us consider the DFA in Figure 1. Table 1 depicts the behaviour of the algorithm. Each row in the table summarizes an iteration. The information shown for each iteration consist on: the splitter took into account; the waiting set; and the partition obtained (whenever it was modified with respect to the previous one).*

The algorithm considers initially the trivial partition of final and non-final states $\pi = \{\{2, 3, 4, 6, 7\}, \{1, 5, 8, 9, 10\}\}$, and updates the set $\mathscr{L}$ with the pairs $(\{2, 3, 4, 6, 7\}, a)$ for each $a \in \Sigma$. In this run we will follow a breath-first *extraction criterion.*

0	π	$\{\{2,3,4,6,7\},\{1,5,8,9,10\}\}$
	$\mathscr{L}$	$\{(\{2,3,4,6,7\},a),(\{2,3,4,6,7\},b)\}$
1	(S,a)	$(\{2,3,4,6,7\},a)$
	$\delta^{-1}(S,a)$	$\{1,2,3,4,5,7,9\}$
	π	$\{\{2,3,4,7\},\{6\},\{1,5,9\},\{8,10\}\}$
	$\mathscr{L}$	$\{(\{2,3,4,6,7\},b),(\{1,2,3,4,5,7,9\},a),(\{1,2,3,4,5,7,9\},b)\}$
2	(S,a)	$(\{2,3,4,6,7\},b)$
	$\delta^{-1}(S,a)$	$\{1,4,5,6\}$
	π	$\{\{2,3,7\},\{4\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{c}(\{1,2,3,4,5,7,9\},a),(\{1,2,3,4,5,7,9\},b),\\(\{1,4,5,6\},a),(\{1,4,5,6\},b)\end{array}\right\}$
3	(S,a)	$(\{1,2,3,4,5,7,9\},a)$
	$\delta^{-1}(S,a)$	$\{1,2,4,5,7,9\}$
	π	$\{\{2,7\},\{3\},\{4\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{c}(\{1,2,3,4,5,7,9\},b),(\{1,4,5,6\},a),(\{1,4,5,6\},b),\\(\{1,2,4,5,7,9\},a),(\{1,2,4,5,7,9\},b)\end{array}\right\}$
4	(S,a)	$(\{1,2,3,4,5,7,9\},b)$
	$\delta^{-1}(S,a)$	$\{1,2,3,4,5,6,7\}$
	$\mathscr{L}$	$\left\{\begin{array}{c}(\{1,4,5,6\},a),(\{1,4,5,6\},b),\\(\{1,2,4,5,7,9\},a),(\{1,2,4,5,7,9\},b)\end{array}\right\}$
5	(S,a)	$(\{1,4,5,6\},a)$
	$\delta^{-1}(S,a)$	$\{2,3,4,7\}$
	$\mathscr{L}$	$\{(\{1,4,5,6\},b),(\{1,2,4,5,7,9\},a),(\{1,2,4,5,7,9\},b)\}$
6	(S,a)	$(\{1,4,5,6\},b)$
	$\delta^{-1}(S,a)$	$\{2,3\}$
	π	$\{\{2\},\{7\},\{3\},\{4\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{c}(\{1,2,4,5,7,9\},a),(\{1,2,4,5,7,9\},b),\\(\{2,3\},a),(\{2,3\},b)\end{array}\right\}$
7	(S,a)	$(\{1,2,4,5,7,9\},a)$
	$\delta^{-1}(S,a)$	$\{1,2,4,5,7,9\}$
	$\mathscr{L}$	$\{(\{1,2,4,5,7,9\},b),(\{2,3\},a),(\{2,3\},b)\}$
8	(S,a)	$(\{1,2,4,5,7,9\},b)$
	$\delta^{-1}(S,a)$	$\{2,3,4,7\}$
	$\mathscr{L}$	$\{((\{2,3\},a),(\{2,3\},b)\}$
9	(S,a)	$(\{2,3\},a)$
	$\delta^{-1}(S,a)$	$\{1\}$
	π	$\{\{2\},\{7\},\{3\},\{4\},\{6\},\{1\},\{5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\{(\{2,3\},b),(\{1\},a),(\{1\},b)\}$

Table 1. Run of PRD algorithm when the input is the automaton in Figure 1. Note that the table does not show the last iterations (that completely process the waiting set) because the partition is not further modified. Note also that the partition is shown only when it is modified

Algorithm 4.1 A minimization algorithm by *partial reverse determinization* (PRD)

Require: A DFA A
Ensure: The minimal DFA equivalent to A

Method
$\pi = \{F,\ Q - F\}$
$S = F$
$\mathscr{L} = \{\}$
for all $a \in \Sigma$ **do**
  $\mathscr{L} = Append(\mathscr{L},\ (S, a))$
end for
while $\mathscr{L} \neq \{\}$ **do**
  Extract (S, a) in $\mathscr{L}$
  Delete (S, a) from $\mathscr{L}$
  $\pi' = \pi$
  for all $B \in \pi$ which is refined by (S, a) **do**
    Let B' and B'' the result of the split $(S, a)|B$
    Substitute in π the block B for B' and B''
  end for
  if $\pi \neq \pi'$ **then**
    for $b \in \Sigma$ **do**
      $\mathscr{L} = Append(\mathscr{L},\ (\delta^{-1}(S, a), b))$
    end for
  end if
end while
Return (A/π)
End Method.

The algorithm considers in each iteration a splitter to refine the current partition. For instance, in iteration 2 *the algorithm considers the splitter* $(\{2, 3, 4, 6, 7\}, b)$. *Therefore, the set* $\delta^{-1}(\{2, 3, 4, 6, 7\}, b) = \{1, 4, 5, 6\}$ *guide the refinement of the partition to obtain the following one:*

$$\pi = \{\{2, 3, 7\}, \{4\}, \{6\}, \{1, 5\}, \{9\}, \{8, 10\}\},$$

and the update of the waiting set $\mathscr{L}$ *lead to the set:*

$$\mathscr{L} = \{(\{1, 2, 3, 4, 5, 7, 9\}, a), (\{1, 2, 3, 4, 5, 7, 9\}, b), (\{1, 4, 5, 6\}, a), (\{1, 4, 5, 6\}, b)\}.$$

The last modification of the partition is carried out by the consideration of the splitter $(\{2, 3\}, a)$. *Note that* $\delta^{-1}(\{2, 3\}, a) = \{1\}$, *that leads to obtain the partition:*

$$\pi = \{\{2\}, \{7\}, \{3\}, \{4\}, \{6\}, \{1\}, \{5\}, \{9\}, \{8, 10\}\},$$

which is not further modified by the algorithm.

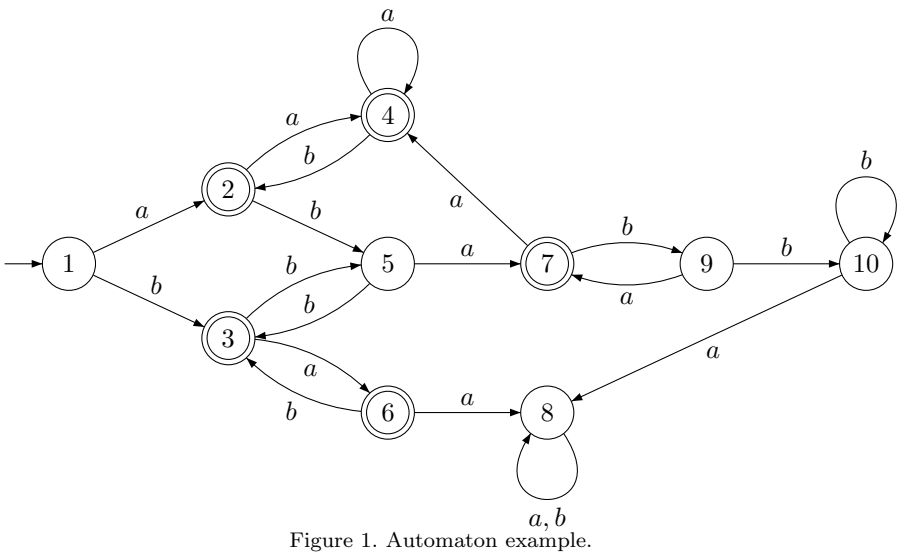

Figure 1. Automaton example.

In order to prove the correctness of the algorithm, we will prove that a splitter (a state of the automaton $DR(A)$) that does not refine the current partition can be discarded. As we mentioned before this is the difference with respect Champarnaud et al. algorithm.

Proposition 4. *For any given DFA $A = (Q, \Sigma, \delta, q_0, F)$ Algorithm 4.1 outputs the minimum DFA equivalent to A.*

Proof. *Let us first recall Proposition 2 that proves that, it is possible to minimize the automaton using the set of states of the automaton $DR(A)$ in a splitting process. We also stress that it is not necessary to use all the states in that automaton, because, in order to minimize any DFA with n states, only $n - 1$ splitters are needed in the worst case.*

Consider that, at a given iteration, the algorithm has taken into account the set of splitters $\mathscr{S} = \{P_1, P_2, \ldots, P_n\}$ (states of the automaton $DR(A)$), being π the current partition. Let us also consider that the algorithm considers (P_i, a) as a splitter and it is such that $\delta^{-1}(P_i, a) = P$ where:

- *P does not refine the partition π.*
- *There is a string $x \in \Sigma^*$ such that $\delta^{-1}(P, x)$ allows to distinguish two states q_1 and q_2.*

note that this means that

$$q_1 \in \delta^{-1}(P, x) \text{ if and only if } q_2 \notin \delta^{-1}(P, x),$$

or, in other terms

$$\delta(q_1, x^r) \in P \text{ if and only if } \delta(q_2, x^r) \notin P.$$

Note that P distinguishes the states $\delta(q_1, x^r)$ and $\delta(q_2, x^r)$, but as stated above, P does not refine the partition π. Therefore, there must be a set $P' \in \mathscr{S}$ that distinguished the states $\delta(q_1, x^r)$ and $\delta(q_2, x^r)$ in a previous iteration. Therefore,

$$\delta(q_1, x^r) \in P' \text{ if and only if } \delta(q_2, x^r) \notin P',$$

which implies that

$$q_1 \in \delta^{-1}(P', x) \text{ if and only if } q_2 \notin \delta(P', x),$$

and therefore, the discard of P does not affect to the minimization process. □

We note that, as opposed to the method proposed by Champarnaud in cite, PRD algorithm does not need to compute completely the automaton $DR(A)$. Nevertheless, as it is proposed, PRD algorithm is a variant of Champarnaud's algorithm, where its time complexity is quadratic with respect the number of states of the automaton.

Proposition 5. *Algorithm 4.1 run with $\mathcal{O}(k\ n^2)$ time complexity, where $k = |\Sigma|$ and $n = |Q|$.*

Proof. *First note that the number of iterations of the loop in line 8 is determined by the number of elements in $\mathscr{L}$ which is linear with both the number of states (it suffices $n-1$ splitters in the worst case to distinguish the states of the automaton) and the number of symbols in the alphabet. Taking into account that the split operation in line 12 can be carried out in linear time with respect n, the final bound is obtained.* □

Let us consider here the case when the input automaton is non-deterministic. We note that in this case, the partial computation of the states of $DR(A)$, in line with the PRD algorithm, does not allow the computation of the equivalence of the right languages of the states (i.e. the computation of the relation $\equiv_A$). Note that it would imply that the equivalence of (non-deterministic) automata could be stablished with polynomial time bound, which is known to be false.

5. Hopcroft's algorithm

The most time efficient algorithm known to minimize DFA is due to Hopcroft [4]. A careful implementation of this algorithm lead to a worst case time of $\mathcal{O}(kn\ logn)$. Many papers are devoted to describe this method [15, 16, 17, 18, 11], in spite of that, no clear relationship among Hopcroft and Brzozowski has been described so far.

Hopcroft's method is outlined in Algorithm5.1. Briefly, the algorithm maintains a *waiting set* $\mathscr{L}$ of splitters to consider in the refinement of the current partition π. Usually, the pair $(\pi, \mathscr{L})$ is referred to as a *configuration* of the algorithm. Note that the algorithm does not fix any order to extract an element from $\mathscr{L}$.

Algorithm 5.1 Hopcroft's DFA minimization algorithm.

Require: A DFA A
Ensure: The minimal DFA equivalent to A

Method
$\pi = \{F,\ Q - F\}$
$S =$ the smallest of the sets F and $Q - F$
$\mathscr{L} = \{\}$
for all $a \in \Sigma$ **do**
  $\mathscr{L} = Append(\mathscr{L},\ (S, a))$
end for
while $\mathscr{L} \neq \{\}$ **do**
  Extract (S, a) in $\mathscr{L}$
  Delete (S, a) from $\mathscr{L}$
  for $B \in \pi$ such that B is refined by (S, a) **do**
    Let B' and B'' the result of the split $(S, a)|B$
    Substitute in π the block B for B' and B''
    $C =$ the smallest of the sets B' and B''
    for all $a \in \Sigma$ **do**
      if $(B, a) \in \mathscr{L}$ **then**
        Update $\mathscr{L}$ by substituting (B, a) for (B', a) and (B'', a)
      else
        $\mathscr{L} = Append(\mathscr{L}, (C, a))$
      end if
    end for
  end for
end while
Return (A/π)
End Method.

The clever choice of the smallest set obtained in each refinement is the key to achieve the, best up to now, time complexity of a DFA minimization method. In [11], Berstel et al. give a proof of the correctness and termination of the algorithm. The proof takes into account Lemma 6 and prove a condition that is fulfilled in every configuration of any run of Hopcroft's algorithm. Proposition 7 enunciates the condition.

Lemma 6 (Hopcroft) *Let P be a set of Q, and let $\pi = P_1, P_2$ be a partition of P. For any $R \subset Q$ and $a \in \Sigma$, it is fulfilled that:*

$$(P, a)|R \wedge (P_1, a)|R = (P, a)|R \wedge (P_2, a)|R = (P_1, a)|R \wedge (P_2, a)|R.$$

Proposition 7 (Berstel et al.[11]) *Let $(\pi, \mathscr{L})$ be a configuration in some execution of Hopcroft's algorithm on an automaton A. For any $P \in \pi$, any subset R of*

a class of π *and* $a \in A$*, one has*

$$(P, a)|R \geq \bigwedge_{(S,a)\in\mathscr{L}} (S, a)|R.$$

These results imply that the partition output by Hopcroft's algorithm cannot be refined, and therefore it denotes the classes of the relation $\equiv_A$. For further details we refer the interested reader to [11].

6. A modification of *PRD* algorithm

Algorithm 4.1 takes into account some states in $\mathscr{R}$ to refine the initial (trivial) partition of the states. It is worth to be noted here that, for any $P \in \mathscr{R}$ considered in this process, in the general case, not all the states in P are relevant to refine the current partition, thus, it is possible to modify the algorithm in order to consider just those *relevant* states. For instance, let us consider the partition $\pi = \{\{1, 2, 5\}, \{3, 4, 6\}, \{7, 8\}\}$, and the splitter $\{1, 2, 7, 8\}$. Note that the partition is refined and that the new one is $\pi = \{\{1, 2\}, \{5\}, \{3, 4, 6\}, \{7, 8\}\}$. The modification we refer above implies to consider the set of *relevant* states (the set $\{1, 2\}$ in this case) instead of the whole set.

Another modification that can be considered consist on, once a block is known to be refined, to select from the split result, the smallest set obtained. In the previous example, it leads to consider the set $\{5\}$ instead of the set $\{1, 2\}$.

Note that, in the new algorithm, it is impossible for any splitter to appear twice in the queue, because, in that case it would mean that some (non-refined) blocks are joined to obtain such splitter. Both modification to PRD algorithm are summarized in Algorithm 6.1.

Following example illustrates the behaviour of this revised version of PRD algorithm.

Example 8. *Let us consider again the* DFA *in Figure 1. Table 2 depicts the behaviour of the algorithm.*

The algorithm considers initially the trivial partition of final and non-final states $\pi = \{\{2, 3, 4, 6, 7\}, \{1, 5, 8, 9, 10\}\}$*, and updates the set* $\mathscr{L}$ *with the pairs* $(\{2, 3, 4, 6, 7\}, a)$ *for each* $a \in \Sigma$*. In this run we follow a random criterion to extract the splitter.*

Note, for instance, that iteration 1 *considers the splitter* $(\{2, 3, 4, 6, 7\}, a)$*. Therefore, the set* $\delta^{-1}(\{2, 3, 4, 6, 7\}, a) = \{1, 2, 3, 4, 5, 7, 9\}$ *guide the refinement of the partition to obtain the following one:*

$$\pi = \{\{2, 3, 7\}, \{4\}, \{6\}, \{1, 5\}, \{9\}, \{8, 10\}\}.$$

In this situation, previous version of PRD *algorithm included the pairs* $(\{1, 2, 3, 4, 5, 7, 9\}, a)$ *and* $(\{1, 2, 3, 4, 5, 7, 9\}, b)$ *into the waiting set* $\mathscr{L}$*. In this version, the algorithm considers that the blocks* $\{2, 3, 4, 6, 7\}$ *and* $\{1, 5, 8, 9, 10\}$ *are splitted. The smallest sets obtained by the split operations are* $\{8, 10\}$ *and* $\{6\}$*, which are*

Algorithm 6.1 $PRD2$ algorithm.

Require: A DFA A
Ensure: The minimal DFA equivalent to A

Method
$\pi = \{F,\ Q - F\}$
$S =$ the smallest of the sets F and $Q - F$
$\mathscr{L} = \{\}$
for all $a \in \Sigma$ **do**
  $\mathscr{L} = Append(\mathscr{L},\ (S, a))$
end for
while $\mathscr{L} \neq \{\}$ **do**
  Extract (S, a) in $\mathscr{L}$
  Delete (S, a) from $\mathscr{L}$
  $S = \emptyset$
  for all $B \in \pi$ which is refined by (S, a) **do**
    $(B',\ B'') = (S, a)|B$
    if $B' \neq \emptyset$ **and** $B'' \neq \emptyset$ **then**
      Update $\mathscr{L}$ by substituting any (B, a) for (B', a) and (B'', a)
      $C =$ the smallest of the sets B' and B''
      $S = S \cup C$
    end if
  end for
  if $S \neq \emptyset$ **then**
    for $b \in \Sigma$ **do**
      $\mathscr{L} = Append(\mathscr{L},\ (S, b))$
    end for
  end if
end while
Return (A/π)
End Method.

joined to obtain the pairs $(\{6, 8, 10\}, a)$ *and* $(\{6, 8, 10\}, b)$ *that update the waiting set* $\mathscr{L}$*.*

The last modification of the partition is carried out by the consideration of the splitter $(\{2, 3\}, a)$*. Table 2 does not show the remaining iterations because the partition is not further modified.*

It is worth to be noted that the modified version of PRD algorithm is closely related with Hopcroft's algorithm. The main difference lies in how the split of a block is considered to further refine the partition. In this sense, Hopcroft's algorithm, for each block splitted, considers the smallest set obtained. The algorithm we propose, considers the union of these sets instead of using them independently. Following

0	π	$\{\{2,3,4,6,7\},\{1,5,8,9,10\}\}$
	$\mathscr{L}$	$\{(\{2,3,4,6,7\},a),(\{2,3,4,6,7\},b)\}$
1	(S,a)	$(\{2,3,4,6,7\},a)$
	$\delta^{-1}(S,a)$	$\{1,2,3,4,5,7,9\}$
	π	$\{\{2,3,4,7\},\{6\},\{1,5,9\},\{8,10\}\}$
	$\mathscr{L}$	$\{(\{2,3,4,6,7\},b),(\{6,8,10\},a),(\{6,8,10\},b)\}$
2	(S,a)	$(\{6,8,10\},b)$
	$\delta^{-1}(S,a)$	$\{8,9,10\}$
	π	$\{\{2,3,4,7\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{2,3,4,6,7\},b),(\{6,8,10\},a),\\(\{9\},a),(\{9\},b)\end{array}\right\}$
3	(S,a)	$(\{9\},b)$
	$\delta^{-1}(S,a)$	$\{7\}$
	π	$\{\{2,3,4\},\{7\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{2,3,4,6,7\},b),(\{6,8,10\},a),\\(\{9\},a),(\{7\},a),(\{7\},b)\end{array}\right\}$
4	(S,a)	$\{2,3,4,6,7\},b)$
	$\delta^{-1}(S,a)$	$\{1,4,5,6\}$
	π	$\{\{2,3\},\{4\},\{7\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{6,8,10\},a),(\{9\},a),(\{7\},a),\\(\{7\},b),(\{4\},a),(\{4\},b)\end{array}\right\}$
5	(S,a)	$(\{4\},b)$
	$\delta^{-1}(S,a)$	$\emptyset$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{6,8,10\},a),(\{9\},a),(\{7\},a),\\(\{7\},b),(\{4\},a)\end{array}\right\}$
6	(S,a)	$(\{6,8,10\},a)$
	$\delta^{-1}(S,a)$	$\{3,6,8,10\}$
	π	$\{\{2\},\{7\},\{3\},\{4\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{9\},a),(\{7\},a),(\{7\},b),\\(\{4\},a),(\{3\},a),(\{3\},b)\end{array}\right\}$
7	(S,a)	$(\{7\},a)$
	$\delta^{-1}(S,a)$	$\{5,9\}$
	π	$\{\{2\},\{7\},\{3\},\{4\},\{6\},\{1\},\{5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{9\},a),(\{7\},b),(\{4\},a),\\(\{3\},a),(\{3\},b),(\{1\},a),(\{1\},b)\end{array}\right\}$

Table 2. Run of PRD 2 algorithm when the input is the automaton in Figure 1. As in the previous example, we do not show the remaining iterations because they do not modify the partition.

lemma proves that the refinement of a partition does not change when the sets are

united.

Lemma 9. *Let $A = (Q, \Sigma, \delta, I, F)$ be a DFA. Let $P, P_1, P_2 \subset Q$ be such that $P_1 \subset P$ and $P_2 \cap P = \emptyset$. For any $R \subset Q$ and $a \in \Sigma$, it is fulfilled that:*

$$(P,a)|R \wedge (P_1,a)|R \wedge (P_2,a)|R = (P,a)|R \wedge (P_1 \cup P_2,a)|R.$$

Proof. *Let us first note that, for every symbol a, the inclusion relationships between P, P_1 and P_2 also hold when the related sets $P' = \delta^{-1}(P,a)$, $P'_1 = \delta^{-1}(P_1,a)$ and $P'_2 = \delta^{-1}(P_2,a)$ are considered.*

Let us remark that, given $P_1, P_2, \ldots, P_k \subset Q$, it is fulfilled that:

$$\bigwedge_{i=1}^{k} P_i|R = \bigwedge_{j=1}^{n} B_j|R$$

where B_j are the blocks of the partition of Q obtained by:

$$\bigwedge_{i=1}^{k} P_i|Q$$

thus,we will consider in the following argument the effect of P, P_1 and P_2 over Q.

Note that the split $(P,a)|Q$ returns the partition $\{P' \cap Q,\ Q - P'\}$. Taking into account the relationship between the sets P', P'_1 and P'_2, it can be seen that the consideration of the splitter (P_1,a) leads to $\{(P' - P'_1) \cap Q,\ P'_1 \cap Q,\ Q - P'\}$, and finally, the partition $\{(P' - P'_1) \cap Q,\ P'_1 \cap Q,\ P'_2 \cap Q,\ Q - (P' \cup P'_2)\}$ is obtained when (P_2,a) is considered.

In a similar way, it can be seen that, when the splitters (P,a) and $(P_1 \cup P_2,a)$, the same partition of the set Q is obtained. □

To prove the correctness of Algorithm 6.1, we will follow an approach similar to the one by Berstel et al. in [11], where Proposition 10 plays the role of Proposition 7 in the proof of Hopcroft's algorithm.

Proposition 10. *Given any execution of Algorithm 6.1, let $\pi_0, \pi_1 \ldots$ denote the sequence of partitions of the set of states obtained. Let also $\mathscr{L}_i$ denote the waiting set once obtained π_i and let the set $C_i = \{B_j \in \pi_j \ :\ 0 \leq j \leq i\}$. For any $a \in \Sigma$ it is fulfilled that:*

$$\pi_i \wedge \bigwedge_{(S,a)\in\mathscr{L}_i} (S,a)|Q \ = \ \pi_i \wedge \bigwedge_{B\in C_i} (B,a)|Q.$$

Proof. *We will prove the proposition by induction on the sequence of partitions obtained by Algorithm 6.1.*

Initially, $\pi_0 = \{F, Q - F\}$ and $\mathscr{L}_0 = \{(T,a) \ :\ a \in \Sigma\}$ where T is the smallest set of F and $Q - F$. Note that:

$$\pi_0 \wedge (T,a)|Q = \pi_0 \wedge (T,a)|Q \wedge (Q - T,a)|Q$$

because $\pi_0 \wedge (T,a)|Q = \pi_0 \wedge (Q-T,a)|Q$.

Let us suppose that the proposition fulfills for $i \leq k$. *Let the configuration of the algorithm be* $(\pi_k, \mathscr{L}_k)$ *and let* $(S,a) \in \mathscr{L}_k$ *be the splitter to be considered.*

Note that $S = P_1 \cup P_2 \cup \ldots \cup P_r$ *where* $P_i \in \pi_m$ *for* $1 \leq i \leq r$, *and for some* $m \leq k$. *Note also that, for every* i, *there exists* $P_i' \in C_k$ *such that* $P_i \subset P_i'$ *and* $P_j \cap P_i = \emptyset$ *for* $j \neq i$. *Lemma 9 implies that:*

$$\pi_k \wedge (P_i',a)|Q \wedge (P_i \cup P_j,a)|Q = \pi_k \wedge (P_i',a)|Q \wedge (P_i,a)|Q \wedge (P_j,a)|Q,$$

and therefore:

$$\pi_k \wedge (P_i \cup \ldots \cup P_r,a)|Q = \pi_k \wedge (P_1,a)|Q \wedge \ldots \wedge (P_r,a)|Q,$$

thus, we will study, without loss of generality, the case of just one $P \in C_k$. *Let then be* $\pi_{k+1} = \pi_k \wedge (P,a)|Q$. *Two situations arise:*

On the one hand, it is possible that, the splitter does not refine any block, that is, $(S,a)|B = B$ *for each block* $B \in \pi_k$. *Then,* $\pi_{k+q} = \pi_k$ *and the algorithm ends and fulfill the proposition.*

On the other hand, (S,a) *refine the partition, let us then define the set:*

$$\mathscr{B}_k = \{B \in \pi_k \ : \ (P,a)|B \neq B\},$$

note that these are the new blocks to take into account in the minimization process. More formally:

$$\pi_{k+1} = (\pi_k - \mathscr{B}_k) \cup \{(P,a)|B_i \ : \ B_i \in \mathscr{B}_k\}.$$

Let $B_i = B_{i1} \cup B_{i2}$ *for each* $B_i \in \mathscr{B}_k$. *Let us also assume that* $|B_{i1}| \leq |B_{i2}|$. *Thus:*

$$\pi_{k+1} \wedge \bigwedge_{(S,a)\in\mathscr{L}_{k+1}} (S,a)|Q \ = \ \pi_k \wedge \bigwedge_{(S,a)\in\mathscr{L}_k} (S,a)|Q \wedge \bigwedge_{B_i\in\mathscr{B}_k} (B_i,a)|Q,$$

by induction hypothesis we have this equals:

$$\pi_k \wedge \bigwedge_{B\in C_k} (B,a)|Q \wedge \bigwedge_{B_i\in\mathscr{B}_k} (B_i,a)|Q,$$

and, by Lemma 6 it equals also:

$$\pi_k \wedge \bigwedge_{B\in C_k} (B,a)|Q \wedge \bigwedge_{B_i\in\mathscr{B}_k} (B_{i1},a)|Q \wedge (B_{i2},a)|Q,$$

and therefore:

$$\pi_{k+1} \wedge \bigwedge_{(S,a)\in\mathscr{L}_{k+1}} (S,a)|Q \ = \ \pi_k \wedge \bigwedge_{B\in C_{k+1}} (B,a)|Q.$$

□

Corollary 11. *Given any execution of Algorithm 6.1, let* $(\pi_0,\mathscr{L}_0),(\pi_1,\mathscr{L}_1)\ldots$ *denote the sequence of configurations obtained. For each partition obtained* π_i *and each* $B \in \pi_i$, *it is fulfilled that:*

$$\pi_i \wedge (B,a)|Q \geq \pi_i \wedge \bigwedge_{(S,a)\in\mathscr{L}_i} (S,a)|Q.$$

Following Proposition provide the correctness and termination proofs for Algorithm 6.1.

Proposition 12. *For any given input DFA A, Algorithm 6.1 computes the equivalence relation* $\equiv_A$.

Proof. *Note that, once obtained* π_L *the waiting set* $\mathscr{L}$ *is empty and thus, for each* B *in the partition obtained* π*:*

$$\pi \wedge (B,a)|Q \geq \pi.$$ □

Please, note that both Algorithm 6.1 and Hopcroft's have the same time complexity.

In this paper we have related the Brzozowski double reversal minimization algorithm with the different split minimization methods. To do so we first proposed PRD algorithm that, for any given DFA A, takes into account some (linearly-bounded number of) states of the $DR(A)$ automaton to obtain the Nerode's equivalence. We remark that, general split-minimization algorithms (for instance the algorithms by Hopcroft, Moore, as well as $PRD2$ here proposed), consider splitters that may not be states of the $DR(A)$ automaton.

Let us note that any valid splitter in a minimization method denote the union of some equivalence classes of the relation $\equiv_A$. Let us take into account the automaton A^{DR} as the output of the determinization of $R(A)$ automaton where the non-accesible states are also considered. Note that the set of states of A^{DR} include all the states in $DR(A)$, and therefore, the set contains information to effectively split the set of states of A into the classes of the relation $\equiv_A$. In fact, the set of states of A^{DR} include every set that splits the set of states of A while respecting the classes of equivalence of the relation $\equiv_A$. Therefore, any split-minimization method could be seen as a method that selects some states of A^{DR}.

Let us also recall that it is possible to denote any union of classes in terms of the intersection and complement of some other classes. Therefore, the whole set of possible splitters used by any split-minimization algorithm (already proposed or not) can be obtained by the intersection and complement closure of the set of accesible states of the $DR(A)$ automaton. This is summarized in Proposition 13 that extends the previous result by Champarnaud et al.

Proposition 13. *Let* $A = (Q, \Sigma, \delta, q_0, F)$ *be a DFA that accepts a language* L *and let* $\mathscr{R}$ *the set of states of* $DR(A)$*. Let* $\mathscr{R}'$ *be the closure of* $\mathscr{R}$ *under intersection and complement with respect* Q*. For each pair of states* $p, q \in Q$*,* $R_p^A = R_q^A$ *if and only if, for all* $P \in \mathscr{R}'$*, it is fulfilled that* $p \in P \Leftrightarrow q \in P$.

Proof. *Easy to prove taking into account Proposition 2 and the properties of intersection and complement.* □

Note that the set $\mathscr{R}'$ contains every possible splitter that can be used to minimize the input automaton. Therefore, the different algorithmic approaches of the split minimization algorithms can be related to a traverse of the set in order to selects a subset of splitters from it.

7. Conclusions

Both Brzozowski and Hopcroft algorithms have important features that make them interesting. The most important feature of Hopcroft's algorithm is its time complexity (in fact it is the most efficient algorithm known). Brzozowski's algorithm is very concise, elegant, easy to implement and, within the recently proposed double reversal framework [9], still arouses interest. Despite the time complexity of Champarnaud et al. [7] algorithm, it can be seen as an interesting attempt to relate Brzozowski's algorithm, when applied to DFA, with other minimization methods.

In the same way algorithm by Champarnaud et al. does, the first algorithm we propose substitute the second determinization by the split of the partitions using the states in $DR(A)$. In contrast to Champarnaud's approach, we do not consider the whole set of states, but a portion (linearly-bounded) of the set, which allows us to carry out the minimization with polynomial time complexity. The consideration of some ideas from Hopcroft's algorithm, a processing of the set of states of $DR(A)$ leads to $PRD2$, that maintains the same structure of PRD but running with a time complexity equal than Hopcroft's.

Finally, we formalize the set of every possible splitter that can be used to minimize an input automaton A as the intersection and complement closure of the set of states of the automaton $DR(A)$. Thus, any split minimization algorithm can be related to a traverse of this set in order to select a subset of splitters from it.

Bibliography

[1] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company, 1979.

[2] M. Almeida, N. Moreira, and R. Reis. Incremental DFA minimisation. In Michael Domaratzki and Kai Salomaa, editors, *CIAA*, volume 6482 of *Lecture Notes in Computer Science*, pages 39–48. Springer, 2010.

[3] E. F. Moore. Gedanken experiments on sequential machines. In C. E. Shannon and J. Mc-Carthy, editors, *Automata Studies*. Princeton Universty Press, 1956.

[4] J. E. Hopcroft. An $n \cdot \log n$ algorithm for minimizing states in a finite automaton. Technical report, Stanford University, Stanford, CA, USA, 1971.

[5] D. Wood. *Theory of Computation*. John Wiley & sons, 1987.

[6] J.A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical Theory of Automata*, pages 529–561, 1962. MRI Symposia Series, Polytecnic Press, Polytecnic Institute of Brooklyn.

[7] J-M. Champarnaud, A. Khorsi, and T. Paranthoën. Split and join for minimizing : Brzozowski´s algorithm. Technical report, Czech Technical University of Prague, 2002. Proceedings of the Prague Stringology Conference 2002 (PSC'02).
[8] B. Watson. A taxonomy of finite automata construction algorithms. Technical report, Computing Science, 1993.
[9] J. A. Brzozowski and H. Tamm. Theory of átomata. In Giancarlo Mauri and Alberto Leporati, editors, *Developments in Language Theory*, volume 6795 of *Lecture Notes in Computer Science*, pages 105–116. Springer, 2011.
[10] M. Vázquez de Parga, P. García, and D. López. A polynomial double reversal minimization algorithm for deterministic finite automata. *Theoretical Computer Science*, 487:17–22, 2013.
[11] J. Berstel, L. Boasson, O. Carton, and I. Fagnot. *Automata: from Mathematics to Applications*, chapter Minimization of automata. European Mathematical Society. (arXiv:1010.5318v3). To appear.
[12] B. Courcelle, D. Niwinski, and A. Podelski. A geometrical view of the determinization and minimization of finite-state automata. *Mathematical Systems Theory*, 24(2):117–146, 1991.
[13] S. Lombardy and J. Sakarovitch. Star height of reversible languages and universal automata. *LNCS*, 2286:76–90, 2002. Proceedings of the 5th LATIN conference.
[14] L. Polák. Minimalizations of NFA using the universal automaton. *Int. J. Found. Comput. Sci.*, 16(5):999–1010, 2005.
[15] D. Gries. Describing an algorithm by Hopcroft. *Acta Informatica*, 2:97–109, 1973.
[16] A. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley Publishing Company, 1974.
[17] N. Blum. A $\mathcal{O}(n \log n)$ implementation of the standard method for minimizing n-state finite automata. *Information Processing Letters*, 57:65–69, 1996.
[18] T. Knuutila. Re-describing an algorithm by Hopcroft. *Theoretical Computer Science*, 250:333–363, 2001.