DFA minimization: from Brzozowski to Hopcroft

Pedro García, Damián López and Manuel Vázquez de Parga
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
{pgarcia,mvazquez,dlopez}@dsic.upv.es

Abstract

Minimization of automata is a classic problem in Computer Science which is still studied nowadays. In this paper, we first propose a polynomial minimization method directly derived from Brzozowski's algorithm, and second, we show how the consideration of some efficiency improvements on this algorithm lead to obtain an algorithm equivalent to Hopcroft's.

Keywords: DFA minimization; Brzozowski algorithm; Hopcroft 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, arouse 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 equivalent to the computation of the Nerode's equivalence relation for the language that accepts the automaton to be minimized.

The majority of the methods that compute the above mentioned relationship 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 (trivial) 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 algorithm with the best time complexity ($\mathcal{O}(n \log n)$, where n stands for the number of states of the input automaton).

The minimization algorithm proposed by Brzozowski [6] is usually set appart from the rest [7, 8]. Despite its worst-case exponential time complexity, this very concise and elegant algorithm is based on two well-known automata constructions and its implementation is very straightforward. Essentially, the algorithm computes the automaton $D(R(D(R(A))))$, where $D(A)$ computes the determinization of A by the well-known subset construction and $R(A)$ is the reverse automaton of A.

The paper by Champarnaud et al. [7] can be seen as a first attempt to relate Brzozowski's algorithm with other minimization methods. The authors propose an algorithm that compute the first step of Brzozowski algorithm (the computation of $D(R(A))$) has, take into account the set of states of the resulting automaton to sequentially split the trivial partition of the set of states of A. This, exponential in the worst case algorithm, is interesting because relates Brzozowski algorithm in a, so to speak, Hopcroft framework.

In this paper we study the relationship among these two minimization algorithms. We show that, in fact, the computation carried out by both Brzozowski and Hopcroft is, in essence, the same. We also show that it is possible to define algorithms in the space among both algorithms, also with good time complexity. The paper is structured as follows. Section 2 summarizes the notation used in this work; Section 3 recalls the Brzozowski minimization algorithm in terms that allow us to propose in Section 4 a derived algorithm with polynomial time complexity. Section 5 recalls the algorithm by Hopcroft and the essential results used by Berstel et al. in their proof of termination and correctness [9]. Section 6 show that some efficiency improvements on the algorithm proposed in Section 4 lead to an algorithm equivalent to Hopcroft's.

2 Notation and definitions

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 initial set of states, $F \subseteq Q$ is the set of final states and $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function, which can also be seen as $\delta \subseteq (Q \times \Sigma \times Q)$. The transition function can be extended in a natural way to Σ^*. Given an automaton A, we say it is accessible if, for each $q \in Q$, there exists a string x such that $\delta(q_0, x) = q$. An automaton is called *deterministic* (DFA) if, for every state q and every symbol a, the number of transitions $\delta(q, a)$ is at most one, and it has only one initial state q_0. A DFA is said to be complete whenever the number of transitions $\delta(q, a)$ is just one. In the following we will consider only complete and accessible DFA.

Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$ that accepts a language L, the reverse automaton is defined as the automaton $R(A) = (Q, \Sigma, \delta_r, F, \{q_0\})$, where

$q \in \delta_r(p, a)$ if and only if $\delta(q, a) = p$. Given any language L, we will denote the reverse language by L^r. For any automaton $A = (Q, \Sigma, \delta, I, F)$ it is known that the automaton $A' = (\mathscr{P}(Q), \Sigma, \delta', I, F')$, where $F' = \{P \in \mathscr{P}(Q) \ : \ P \cap F \neq \emptyset\}$ and $\delta'(P, a) = \cup_{p \in P} \delta(p, a)$ is a DFA equivalent to A. Let us denote the accesible (trimmed) version of A by $D(A)$.

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, the coarsest partition which refines both π_1 and π_2, and we will denote it $\pi_1 \wedge \pi_2$. The classes of this partition are the non empty sets in $P_1 \cap P_2$, where $P_1 \in \pi_1$ and $P_2 \in \pi_2$.

Let us denote the size of a set Q by $|Q|$. Let us also denote by $\mathscr{P}(Q)$ the power set of Q. 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 an automaton $A = (Q, \Sigma, \delta, q_0, F)$, 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 following sets:

$$\begin{aligned} R' &= \{q \in R \ : \ \delta(q, a) \in P\} = \delta^{-1}(P, a) \cap R \\ R'' = \overline{R'_R} &= \{q \in R \ : \ \delta(q, a) \notin P\} = R - R' \end{aligned}$$

and 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. The process consist of compute the automaton $A' = D(R(D(R(A))))$. 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 $D(R(A))$ is the minimal DFA that accepts the language L^r.*

Despite the fact that, it is possible the size of $D(R(A))$ to be exponential with respect to $|Q|$, we describe below the way Brzozowski's method obtains the minimal DFA for a given automaton.

In the following, we will only consider the case when the input to the algorithm is a complete DFA $A = (Q, \Sigma, \delta, q_0, F)$ for a given language L.

Let $D(R(A)) = (\mathscr{R}, \Sigma, \delta_{dr}, F, F')$ be the trimmed automaton obtained using the reverse and determinization constructions described above. Note that $\mathscr{R} \subseteq \mathscr{P}(Q)$ and $F' = \{P \in \mathscr{R} \ : \ q_0 \in P\}$. Let us also define, for every state $q \in Q$, the set $\mathscr{R}_q = \{P \in \mathscr{R} \ : \ q \in P\}$. The proposition below proves a valid construction for the minimal DFA for A. The terms used in the proof will be useful in the remains of the paper.

Proposition 2 *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts a given language L and let $\mathscr{R} \subseteq \mathscr{P}(Q)$ denote the set of states of the automaton $D(R(A))$. The minimal DFA for A is equal up to isomorphism to $A' = (Q', \Sigma, \delta', q_0', F')$ where:*

- $Q' = \{\mathscr{R}_q \ : \ q \in Q\}$
- $q_0' = \mathscr{R}_{q_0}$
- $F' = \{\mathscr{R}_q \ : \ q \in F\}$
- $\delta'(\mathscr{R}_p, a) = \mathscr{R}_{\delta(p,a)}$

Proof. *We will see that the construction proposed is equivalent to the output of the method by Brzozowski.*

It is easy to see that each states of the $D(R(D(R(A))))$ automaton is an element in $\mathscr{P}(\mathscr{R})$; the initial state of this automaton is $\mathscr{R}_{q_0}$; and the final states are those $\mathscr{P} \in \mathscr{P}(\mathscr{R})$ such that $F \in \mathscr{P}$.

We prove now that the definition of the set of final states is consistent. Let p be a state in A such that $\delta(p, a) = q$, and let P be a state in $D(R(A))$ such that $q \in P$. We recall here that δ_{dr} is the transition function of the automaton $D(R(A))$. First, note that $\delta_{dr}(P, a) \supseteq \delta^{-1}(q, a) \ni p$. Second, let us denote the set $\mathscr{P} = \{P \in \mathscr{R} \ : \ \delta^{-1}(q, a) \subseteq P\} \supseteq \mathscr{R}_p$ (the set of states of the automaton $D(R(D(R(A))))$ which contain $\delta^{-1}(q, a)$, which is included into $\mathscr{R}_p$). Finally, taking into account the set $\mathscr{P}$, it can be seen that there is a relationship among the transition function of the automaton $R(D(R(A)))$ (δ_{dr}^{-1}) and the transition faction of the automaton $D(R(D(R(A))))$ (δ'):

$$\delta'(\mathscr{P}, a) = \bigcup_{P \in \mathscr{P}} \delta_{dr}^{-1}(P, a) \ = \ \mathscr{R}_q$$

This relationship is also hold when the set $\mathscr{R}_p$ is considered instead of $\mathscr{P}$, because, on the one hand $\mathscr{P} \subseteq \mathscr{R}_p$, and, on the other hand $\bigcup_{P \in \mathscr{R}_p} \delta_{dr}^{-1}(P, a) \supseteq \mathscr{R}_q$, thus, it is fulfilled that:

$$\delta'(\mathscr{R}_p, a) = \bigcup_{P \in \mathscr{R}_p} \delta_{dr}^{-1}(P, a) \ = \ \mathscr{R}_q$$

which proves that $\delta'(\mathscr{R}_p, a) = \mathscr{R}_{\delta(p,a)}$. This last result, together with the fact that the final states in the automaton $D(R(D(R(A))))$ are those $\mathscr{P} \in \mathscr{P}(\mathscr{R})$

that $F \in \mathscr{P}$, imply that $F' = \{\mathscr{R}_q \ : \ q \in F\}$, which proves the correctness of the definition.

□

Please note that $|Q| \geq |Q'|$, that is, those states p and q such that $\mathscr{R}_p = \mathscr{R}_q$ are equivalent and lead to a reduction of the automaton. In a more formal way, let us define the function $\varphi : Q \to Q'$ as $\varphi(q) = \mathscr{R}_q$. This function defines a surjective homomorphism $\varphi : A \to A'$, where $ker\varphi$ defines the Nerode's equivalence over A. That is, for $p, q \in Q$, $p \sim_L q$ if and only if $\varphi(p) = \varphi(q)$. Following proposition is a direct derivation from this.

Proposition 3 *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 $D(R(A))$. For each pair of sates $p, q \in Q$, $p \sim_L q$ if and only if, for all $P \in \mathscr{R}$, it is fulfilled that $p \in P \Leftrightarrow q \in P$.*

There are several ways to compute this, Brzozowski's algorithm is one of them, but other authors use this same approach. For instance, Lombardy and Sakarovich [10], and Polak as well [11], 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 $(P, q) \in \mathscr{R} \times Q$:

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

those states with the same column are equivalent. This is a direct result from the way the authors obtain the universal automaton for a given language.

Another related result is due to Champarnaud et al. [7], where obtain the same equivalence relation using the split operation.

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 $D(R(A))$ can be exponentially bigger with respect to A). We now prove that the computation of π_L does not need the whole computation of $D(R(A))$.

4 A polynomial algorithm

Let $A = (Q, \Sigma, \delta, q_0, F)$ a DFA that accepts the language L. Any $E \subset \Sigma^*$ defines a partition π_E over Q where, for all $p, q \in Q$, $B(p, \pi_E,) = B(q, \pi_E)$ if and only if, for every $e \in E$, is fulfilled that, $\delta(p, e) \in F$ if and only if $\delta(q, e) \in F$. In the following we will refer to this set as a *set of experiments*. Note that, for any $E \subset \Sigma^*$, it is fulfilled that $L(A) \subseteq L(A/\pi_E)$.

We now extend the previous definition to consider those sets of experiments able to obtain the partition induced by Nerode equivalence.

Definition 4 *Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$, a set of experiments $E \subset \Sigma^*$ for A is defined to be* complete *if and only if, for every pair of states p and q such that $R_p^A \neq R_q^A$, there exists $e \in E$ such that $\delta(p, e) \in F$ if and only if $\delta(q, e) \notin F$.*

Example 5 *Let us consider the DFA in Figure 1:*

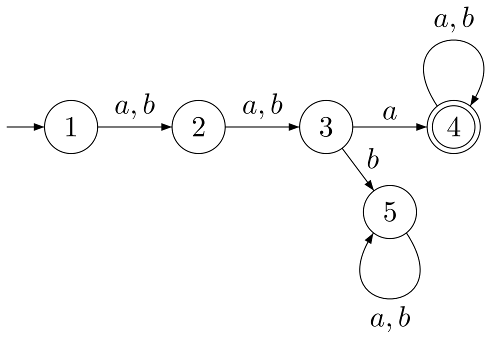

Figure 1: Automaton example.

Given that λ distinguish the final state from the other states. Note that there are different complete sets of experiments. Among others, some of these sets are:

$$\{\{\lambda, a, a^2, a^3\}, \{\lambda, a, ba, ba^2\}, \{\lambda, a, aba, a^2\}, \ldots\}$$

The property below is a direct consequence of previous definitions

Property 6 *Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$ and $E \subset \Sigma^*$ a complete set of experiments for A. It is fulfilled that $L(A) = L(A/\pi_E)$. Furthermore, A/π_E is equal up to isomorphism to the minimal DFA for $L(A)$.*

Proof. *Note that, by definition, a complete set of experiments contains strings to distinguish all the distinguishable states. Thus, for each pair of states p, q, it is fulfilled that $B(p, \pi_E,) = B(q, \pi_E)$ if and only if $R_p^A = R_q^A$, that is, $\pi_E = \pi_L$.* $\square$

We note here that, for any DFA $A = (Q, \Sigma, \delta, q_0, F)$ with n states, it suffices $n-1$ strings (a complete set of experiments with size $n-1$) in the worst case, to refine the initial partition in order to distinguish all the states of A. Taking into account the automaton in Example 5, note that both $\{\lambda, a, a^2, a^3\}$ and $\{\lambda, a, ba, ba^2\}$ are complete sets of experiments, but the first one is suffix closed. Proposition 7 proves that, for each automaton, there always exist a suffix closed set of experiments.

Proposition 7 *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA and E a set of experiments for A. Let $a \in \Sigma$ and $e \in E$ be such that $ae \notin E$ and $\pi_{E \cup \{ae\}} = \pi_E$.*

If there exists $y \in \Sigma^*$ such that $\pi_{E\cup\{yae\}}$ refines π_E, then there exists an experiment $e' \in E$ such that $\pi_{E\cup\{ye'\}} = \pi_{E\cup\{yae\}}$.

Proof. *Let $p, q \in Q$ be the pair of distinguishable states that cannot be distinguished with ae but such that yae distinguishes them. In more formal terms, p and q are such that $\delta(p, yae) \in F$ and $\delta(q, yae) \notin F$. From this follows that $\delta(\delta(p, y), ae) \in F$ and $\delta(\delta(q, y), ae) \notin F$. Let us denote $\delta(p, y) = p'$ and $\delta(q, y) = q'$, thus, $\delta(p', ae) \in F$ implies $\delta(q', ae) \notin F$.*

Note that $\pi_{E\cup\{ae\}} = \pi_E$, which means that there is an experiment $e' \in E$ such that $\delta(p', e') \in F$ and $\delta(q', e') \notin F$, that is, an experiment that distinguishes p' and q'. Therefore, $\delta(p, ye') \in F$ if and only if $\delta(q, ye') \notin F$ and $\pi_{E\cup\{ye'\}} = \pi_{E\cup\{yae\}}$. □

In other words, Proposition 7 proves that, given a set of experiments E, an experiment $e \in E$ and a symbol $a \in \Sigma$, if ae does not refine the partition induced by E, then any refinement induced by strings in Σ^+ae can be obtained by some other strings in Σ^+E. This implies that is possible to discard any experiment such that ae is one of its suffixes. As a direct consequence of previous proposition, following corollary proves a condition for a suffix-closed set of experiments to be complete.

Corollary 8 *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA and E a suffix-closed set of experiments for A. If $\pi_E = \pi_{E\cup\{ae\}}$ for all $a \in \Sigma$ and $e \in E$, then E is a complete set of experiments.*

Proposition 9 *For any automaton $A = (Q, \Sigma, \delta, q_0, F)$, there exists a suffix closed complete set of experiments E with size bounded by $n - 1$ where n denotes the number of states of A.*

Proof. *We will prove the proposition by induction on the number of strings in E.*

Let $E_0 = \{\lambda\}$, suffix-closed and bounded by $n-1$. The set of experiments E_0 produces the trivial partition $\pi_{E_0} = \{F, Q - F\}$. If $L(A) = L(A/\pi_{E_0})$ then the proposition is proved.

If $L(A) \neq L(A/\pi_{E_0})$ then let $E_k = \{e_0 = \lambda, e_1, \ldots, e_k\}$ be a suffix-closed set of experiments such that $\pi_{E_{i+1}}$ refines π_{E_i} for $1 \leq i \leq k$. If $L(A) \neq L(A/\pi_{E_k})$, then Corollary 8 proves that there exists $a \in \Sigma$ and an experiment $e_j \in E_k$ to obtain $E_{k+1} = \{e_0, \lambda, e_1, \ldots, e_k, ae_j\}$ suffix-closed and such that $\pi_{E_{k+1}}$ refines π_{E_k}., We finally note that it suffices $n-1$ strings in the worst case, to refine the initial partition in order to distinguish all the states of A. □

Let us note here that, given a set of experiments E for an automaton $A = (Q, \Sigma, \delta, q_0, F)$, any $e \in E$ produces a partition of Q in two blocks: $Q_e = \{p \in Q \ : \ \delta(p, e) \in F\}$ and $Q - Q_e$. Let us denote such partition with π_e, then:

$$\pi_L = \bigwedge_{e\in E} \pi_e$$

Let us also note that Q_e can be seen as a state of $D(R(A))$, the only one that fulfills that $\delta_{dr}(F, e^r) = Q_e$.

In the following, we propose an efficient minimization algorithm for DFA using the same approach by Champarnaud cited above. To do so, we first establish a bijection among a suffix closed complete set of experiments for an automaton A and a set of states in the automaton $D(R(A))$. We prove that such set of states can be reached in $D(R(A))$ using a prefix closed set of strings. It is known that the partition π_L can be obtained using at most $n-1$ states. Our algorithm is based on these facts to minimize A without the need of computing the whole set of states of $D(R(A))$.

Definition 10 *Let us consider a DFA $A = (Q, \Sigma, \delta, q_0, F)$ and a set $\mathscr{S} \subseteq \mathscr{P}(Q)$. We define the equivalence relation induced by the set $\mathscr{S}$, and thus, we say that $p \equiv_{\mathscr{S}} q$ if and only if, for all $P \in \mathscr{S}$, it is fulfilled that $p \in P \Leftrightarrow q \in P$.*

Definition 11 *We say that $\mathscr{S} \subseteq \mathscr{P}(Q)$ is a* complete set of states *for a DFA $A = (Q, \Sigma, \delta, q_0, F)$ if and only if, for any given $p, q \in Q$, if $R_p^A \neq R_q^A$, then there exists a set $P \in \mathscr{S}$ such that $p \in P$ if and only if $q \notin P$.*

Following proposition is direct consequence of previous definitions.

Proposition 12 *Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$, if $\mathscr{S} \subseteq \mathscr{P}(Q)$ is a complete set of states, then $\equiv_{\mathscr{S}}$ induces the partition π_L.*

The next proposition can be seen as a consequence of Proposition 3.

Proposition 13 (Brzozowski) *Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$ and $\mathscr{R} \subseteq \mathscr{P}(Q)$ the set of states of $D(R(A))$. The set $\mathscr{R}$ is a complete set of states for A.*

Example 14 *Let us consider the DFA in Figure 1. The states of the $D(R(A))$ automaton are:*

$$\mathscr{R} = \{\{4\}, \{3,4\}, \{2,3,4\}, \{2,4\}, \{1,2,3,4\}, \{1,2,4\}, \{1,3,4\}, \{1,4\}\}$$

and A is already minimal. We note here that $\mathscr{R}$ is a complete set of states. Although not stated in this way, this is the key of Brzozowski's work on automata minimization.

We also show below some complete sets of experiments E for the automaton and their corresponding sets $\mathscr{S}_E$.

E	$\mathscr{S}_E$
$\{\lambda, a, a^2, a^3\}$	$\{\{4\}, \{3,4\}, \{2,3,4\}, \{1,2,3,4\}\}$
$\{\lambda, a, ba, ba^2\}$	$\{\{4\}, \{3,4\}, \{2,4\}, \{1,3,4\}\}$
$\{\lambda, a, aba, a^2\}$	$\{\{4\}, \{3,4\}, \{1,3,4\}, \{2,3,4\}\}$

For each $P \in \mathscr{R}$, let us denote in the sequel with x_P the first string in canonical order such that $\delta_{dr}(F, x_P) = P$.

Definition 15 *We say that a set of states $\mathscr{S}$ is* prefix-closed *if the set $\{x_p^r \ : \ P \in \mathscr{S})\}$ is suffix-closed.*

Proposition 16 *Let $\mathscr{S} \subseteq \mathscr{R}$ be a complete set of states for a DFA A. The set $E_{\mathscr{S}} = \{x_P^r \ : \ P \in \mathscr{S}\}$ is a complete set of experiments for A.*

Proof. *For each pair $p, q \in Q$ such that $R_p^A \neq R_q^A$ there exists a set $P \in \mathscr{S}$ where $p \in P$ if and only if $q \notin P$.*

Note that $\delta_{dr}(F, x_P) = P$, therefore, $\delta(p, x_P^r) \in F$ if and only if $\delta(q, x_P^r) \notin F$. Therefore, $E_{\mathscr{S}}$ is a complete set of experiments for A. □

Proposition 17 *For any given DFA $A = (Q, \Sigma, \delta, q_0, F)$, there exists a prefix-closed complete set of states for A whose size is bounded above by $|Q| - 1$*

Proof. *Proposition 9 states that there exists, for any automaton A, a suffix-closed set of experiments E whose size is at most $|Q| - 1$. Therefore, $\mathscr{S}_E$ is a prefix-closed complete set of states, whose size is bounded above by $|Q| - 1$.* □

The argument used in the proof of Proposition 17 allows to carry out the minimization of a *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. We show below an example of run.

Example 18 *Let us consider the DFA in Figure 2. 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.*

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:

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

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)	$(\{2,3,4,6,7\},a)$
	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)	$(\{2,3,4,6,7\},b)$
	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)	$(\{1,2,3,4,5,7,9\},a)$
	π	$\{\{2,7\},\{3\},\{4\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{1,2,3,4,5,7,9\},b),(\{1,4,5,6\},a),(\{1,4,5,6\},b),\\(\{1,2,4,5,7,9\},a),(\{1,2,4,5,7,9\},b)\end{array}\right\}$
4	(S,a)	$\{1,2,3,4,5,7,9\},b)$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{1,4,5,6\},a),(\{1,4,5,6\},b),\\(\{1,2,4,5,7,9\},a),(\{1,2,4,5,7,9\},b)\end{array}\right\}$
5	(S,a)	$(\{1,4,5,6\},a)$
	$\mathscr{L}$	$\{(\{1,4,5,6\},b),(\{1,2,4,5,7,9\},a),(\{1,2,4,5,7,9\},b)\}$
6	(S,a)	$(\{1,4,5,6\},b)$
	π	$\{\{2\},\{7\},\{3\},\{4\},\{6\},\{1,5\},\{9\},\{8,10\}\}$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{1,2,4,5,7,9\},a),(\{1,2,4,5,7,9\},b),\\(\{2,3\},a),(\{2,3\},b)\end{array}\right\}$
7	(S,a)	$(\{1,2,4,5,7,9\},a)$
	$\mathscr{L}$	$\{(\{1,2,4,5,7,9\},b),(\{2,3\},a),(\{2,3\},b)\}$
8	(S,a)	$(\{1,2,4,5,7,9\},b)$
	$\mathscr{L}$	$\{((\{2,3\},a),(\{2,3\},b)\}$
9	(S,a)	$(\{2,3\},a)$
	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 2. 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 $A$
 1: Method
 2: $\pi = \{F,\ Q - F\}$
 3: $S = F$
 4: $\mathscr{L} = \{\}$
 5: for all $a \in \Sigma$ do
 6:    $\mathscr{L} = Append(\mathscr{L},\ (S, a))$
 7: end for
 8: while $\mathscr{L} \neq \{\}$ do
 9:    Extract $(S, a)$ in $\mathscr{L}$
10:    Delete $(S, a)$ from $\mathscr{L}$
11:    $\pi' = \pi$
12:    for all $B \in \pi$ which is refined by $(S, a)$ do
13:       $(B',\ B'') = (S, a)|B$
14:       Substitute in $\pi$ the block $B$ for $B'$ and $B''$
15:    end for
16:    if $\pi \neq \pi'$ then
17:       for $b \in \Sigma$ do
18:          $\mathscr{L} = Append(\mathscr{L},\ (\delta^{-1}((S, a), b))$
19:       end for
20:    end if
21: end while
22: Return $(A/\pi)$
23: End Method.
```

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.

We note that, as opposed to the method proposed by Champarnaud in cite, *PRD* algorithm does not need to compute completely the automaton $D(R(A))$. As it is proposed, *PRD* algorithm is a variant of Brzozowski algorithm but with time complexity bounded by $\mathcal{O}(k \cdot n^2)$, where k denotes the number of symbols and n the number of states of the automaton.

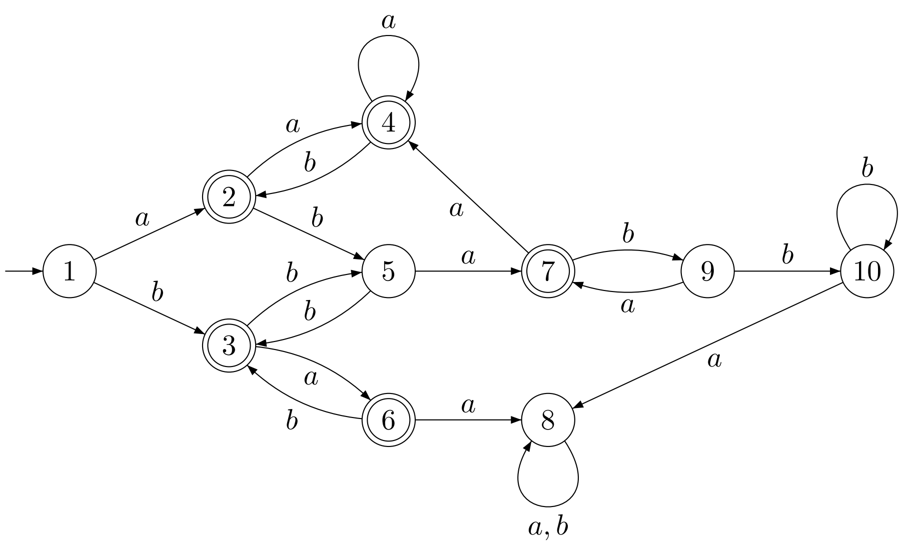

Figure 2: Automaton example.

5 Hopcroft's algorithm

The most time efficient algorithm known to minimize automata is due to Hopcroft [4]. A careful implementation of this algorithm lead to a wost case time $\mathcal{O}(k \cdot n \cdot logn)$. Many papers are devoted to describe this method [12, 13, 14, 15, 9], 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 $\mathscr{L}$.

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 [9], Berstel et al. give a proof of the correctness and termination of the algorithm. The proof takes into account the following lemma:

Lemma 19 (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:*

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

and prove a condition that is fulfilled in every configuration of any run of Hopcroft's algorithm. The following proposition enunciates the condition.

Proposition 20 (Berstel [9]) *Let $(\pi, \mathscr{L})$ be a configuration in some execution of Hopcroft's algorithm on an automaton A. For any $P \in \pi$, any*

Algorithm 5.1 Hopcroft's DFA minimization algorithm.

```
Require: A $DFA$ $A$
Ensure: The minimal $DFA$ equivalent to $A$
 1: Method
 2: $\pi = \{F,\ Q - F\}$
 3: $S$ = the smallest of the sets $F$ and $Q - F$
 4: $\mathscr{L} = \{\}$
 5: for all $a \in \Sigma$ do
 6:   $\mathscr{L} = Append(\mathscr{L},\ (S, a))$
 7: end for
 8: while $\mathscr{L} \neq \{\}$ do
 9:   Extract $(S, a)$ in $\mathscr{L}$
10:   Delete $(S, a)$ from $\mathscr{L}$
11:   for $B \in \pi$ such that $B$ is refined by $(S, a)$ do
12:     $(B',\ B'') = (S, a)|B$
13:     Substitute in $\pi$ the block $B$ for $B'$ and $B''$
14:     Update $\mathscr{L}$ by substituting any $(B, a)$ for $(B', a)$ and $(B'', a)$
15:     $C$ = the smallest of the sets $B'$ and $B''$
16:     for all $a \in \Sigma$ do
17:       $\mathscr{L} = Append(\mathscr{L}, (C, a))$
18:     end for
19:   end for
20: end while
21: Return $(A/\pi)$
22: End Method.
```

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 is the Nerode partition of the input automaton. For further details we refer the interested reader to [9].

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

Algorithm 6.1 A modification of PRD algorithm.

```
Require: A $DFA$ $A$
Ensure: The minimal $DFA$ equivalent to $A$
 1: Method
 2: $\pi = \{F,\ Q - F\}$
 3: $S$ = the smallest of the sets $F$ and $Q - F$
 4: $\mathscr{L} = \{\}$
 5: for all $a \in \Sigma$ do
 6:    $\mathscr{L} = \mathit{Append}(\mathscr{L},\ (S, a))$
 7: end for
 8: while $\mathscr{L} \neq \{\}$ do
 9:    Extract $(S, a)$ in $\mathscr{L}$
10:    Delete $(S, a)$ from $\mathscr{L}$
11:    $S = \emptyset$
12:    for all $B \in \pi$ which is refined by $(S, a)$ do
13:       $(B',\ B'') = (S, a)|B$
14:       if $B' \neq \emptyset$ and $B'' \neq \emptyset$ then
15:          Update $\mathscr{L}$ by substituting any $(B, a)$ for $(B', a)$ and $(B'', a)$
16:          $C$ = the smallest of the sets $B'$ and $B''$
17:          $S = S \cup C$
18:       end if
19:    end for
20:    if $S \neq \emptyset$ then
21:       for $b \in \Sigma$ do
22:          $\mathscr{L} = \mathit{Append}(\mathscr{L},\ (S, b))$
23:       end for
24:    end if
25: end while
26: Return $(A/\pi)$
27: End Method.
```

$\pi = \{\{1,2\},\{5\},\{3,4,6\},\{7,8\}\}$. 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 $\{1,2\}$. Both modification to PRD algorithm are summarized in Algorithm 6.1.

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

Example 21 *Let us consider again the DFA in Figure 2. 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 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 lemma proves that the refinement of a partition does not change when the sets are united.

Lemma 22 *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. *The result of the split* $(P,a)|R$ *is the set* $\{S, \overline{S_R}\}$ *where* $S = \delta^{-1}(P,a) \cap R$*.*

Note that, on the one hand, $(P,a)|R \wedge (P_1,a)|R = \{R_1, R_1'\}$ *where* $R_1 \cup R_1' = S$*, and on the other hand,* $(P,a)|R \wedge (P_2,a)|R = \{R_2, R_2'\}$ *where* $R_2 \cup R_2' = \overline{R_S}$*. Therefore:*

$$(P,a)|R \wedge (P_1,a)|R \wedge (P_2,a)|R = \{R_1, R_1', R_2, R_2'\} = (P,a)|R \wedge (P_1 \cup P_2,a)|R$$

$\square$

To prove the correctness of Algorithm 6.1, we will follow an approach similar to the one by Berstel in [9], where Proposition 23 plays the role of Proposition 20 in the proof of Hopcroft's algorithm.

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)	$(\{2,3,4,6,7\},a)$
	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)	$(\{6,8,10\},b)$
	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)	$(\{9\},b)$
	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)	$\{2,3,4,6,7\},b)$
	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)	$(\{4\},b)$
	$\mathscr{L}$	$\left\{\begin{array}{l}(\{6,8,10\},a),(\{9\},a),(\{7\},a),\\(\{7\},b),(\{4\},a)\end{array}\right\}$
6	(S,a)	$(\{6,8,10\},a)$
	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)	$(\{7\},a)$
	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* algorithm when the input is the automaton in Figure 2. We do not show the remaining iterations (that completely process the waiting set) because they do not modify the partition. The table shows the partition only when it is modified

Proposition 23 *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 22 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 19 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 24 *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 25 *Algorithm 6.1 computes the Nerode's equivalence* π_L.
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 fact both algorithms are the same.

7 Conclusions

In this paper we show that two very dissimilar in the beginning algorithms, carry out in fact very similar computations.

Both 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 and elegant and, despite its worst case time complexity, has a very straightforward implementation. We show how some computations carried out by Brzozowski's algorithm can be avoided to obtain, so to speak, a polynomial version of Brzozowski algorithm. The consideration of some efficiency improvements on this algorithm lead to obtain an algorithm equivalent to Hopcroft's.

We note that, as a byproduct, we also show that it is possible to propose other minimization algorithms between the space among these two classical algorithms. These new proposals may achieve good performance in some circumstances.

References

[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 $n \log n$ algorithm for minimizing states in a finite automaton. Technical report, 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. 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.

[10] S. Lombardy and J. Sakarovitch. Star height of reversible languages and universal automata. *LNCS*, 2286:76–90, 2002. Proceedings of the 5th LATIN conference.

[11] L. Polák. Minimalizations of NFA using the universal automaton. *Int. J. Found. Comput. Sci.*, 16(5):999–1010, 2005.

[12] D. Gries. Describing an algorithm by hopcroft. *Acta Informatica*, 2:97–109, 1973.

[13] A. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley Publishing Company, 1974.

[14] 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.

[15] T. Knuutila. Re-describing an algorithm by hopcroft. *Theoretical Computer Science*, 250:333–363, 2001.