# A Congruence-based Perspective on Automata Minimization Algorithms

**Pierre Ganty**
IMDEA Software Institute, Madrid, Spain
pierre.ganty@imdea.org

**Elena Gutiérrez**
IMDEA Software Institute, Madrid, Spain
Universidad Politécnica de Madrid, Spain
elena.gutierrez@imdea.org

**Pedro Valero**
IMDEA Software Institute, Madrid, Spain
Universidad Politécnica de Madrid, Spain
pedro.valero@imdea.org

## Abstract

In this work we use a framework of finite-state automata constructions based on equivalences over words to provide new insights on the relation between well-known methods for computing the minimal deterministic automaton of a language.

**2012 ACM Subject Classification** Theory of computation → Formal languages and automata theory; Theory of computation → Regular languages

**Keywords and phrases** Double-Reversal Method, Minimization, Automata, Congruences, Regular Languages

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2019.77

**Related Version** An extended version is available at `https://arxiv.org/abs/1906.06194`

**Funding** *Pierre Ganty*: Supported by the Spanish Ministry of Economy and Competitiveness project No. PGC2018-102210-B-I00, BOSCO - Foundations for the development, analysis and understanding of BlOck chains and Smart COntracts, by the Madrid Regional Government project No. S2018/TCS-4339, BLOQUES - Contratos inteligentes y Blockchains Escalables y Seguros mediante Verificación y Análisis, and by a Ramón y Cajal fellowship RYC-2016-20281.
*Elena Gutiérrez*: Supported by BES-2016-077136 grant from the Spanish Ministry of Economy, Industry and Competitiveness.

## 1 Introduction

In this paper we consider the problem of building the minimal deterministic finite-state automaton generating a given regular language. This is a classical issue that arises in many different areas of computer science such as verification, regular expression searching and natural language processing, to name a few.

There exists a number of methods, such as Hopcroft's [10] and Moore's algorithms [14], that receive as input a deterministic finite-state automaton (DFA for short) generating a language and build the minimal DFA for that language. In general, these methods rely on computing a partition of the set of states of the input DFA which is then used as the set of states of the minimal DFA.

On the other hand, Brzozowski [4] proposed the *double-reversal method* for building the minimal DFA for the language generated by an input non-deterministic automaton (NFA for short). This algorithm alternates a reverse operation and a determinization operation twice,


relying on the fact that, for any given NFA $\mathcal{N}$, if the reverse automaton of $\mathcal{N}$ is deterministic then the determinization operation yields the minimal DFA for the language of $\mathcal{N}$. This method has been recently generalized by Brzozowski and Tamm [5]. They showed the following *necessary and sufficient condition*: the determinization operation yields the minimal DFA for the language of $\mathcal{N}$ if and only if the reverse automaton of $\mathcal{N}$ is *atomic*.

It is well-known that all these approaches to the DFA minimization problem aim to compute Nerode's equivalence relation for the considered language. However, the double-reversal method and its later generalization appear to be quite isolated from other methods such as Hopcroft's and Moore's algorithms. This has led to different attempts to better explain Brzozowski's method [3] and its connection with other minimization algorithms [1, 7, 16]. We use a framework of automata constructions based on equivalence classes over *words* to give new insights on the relation between these algorithms.

In this paper we consider equivalence relations over words on an alphabet $\Sigma$ that induce finite partitions over $\Sigma^*$. Furthermore, we require that these partitions are well-behaved with respect to concatenation, namely, *congruences*. Given a regular language $L$ and an equivalence relation satisfying these conditions, we use well-known automata constructions that yield automata generating the language $L$ [6, 13]. In this work, we consider two types of equivalence relations over words verifying the required conditions.

First, we define a *language-based equivalence*, relative to a regular language, that behaves well with respect to *right* concatenation, also known as the right Nerode's equivalence relation for the language. When applying the automata construction to the right Nerode's equivalence, we obtain the minimal DFA for the given language [6, 13]. In addition, we define an *automata-based equivalence*, relative to an NFA. When applying the automata construction to the automata-based equivalence we obtain a determinized version of the input NFA.

On the other hand, we also obtain counterpart automata constructions for relations that are well-behaved with respect to *left* concatenation. In this case, language-based and automata-based equivalences yield, respectively, the minimal co-deterministic automaton and a co-deterministic NFA for the language.

The relation between the automata constructions resulting from the language-based and the automata-based congruences, together with the the duality between right and left congruences, allows us to relate determinization and minimization operations. As a result, we formulate a sufficient and necessary condition that guarantees that determinizing an automaton yields the minimal DFA. This formulation evidences the relation between the double-reversal and the state partition refinement minimization methods.

We start by giving a simple proof of Brzozowski's double-reversal method [4], to later address the generalization of Brzozowski and Tamm [5]. Furthermore, we relate the iterations of Moore's partition refinement algorithm, which works on the states of the input DFA, to the iterations of the greatest fixpoint algorithm that builds the right Nerode's partition on words. We conclude by relating the automata constructions introduced by Brzozowski and Tamm [5], named the *átomaton* and *the partial átomaton*, to the automata constructions described in this work.

**Structure of the paper.** After preliminaries in Section 2, we introduce in Section 3 the automata constructions based on congruences on words and establish the duality between these constructions when using right and left congruences. Then, in Section 4, we define language-based and automata-based congruences and analyze the relations between the resulting automata constructions. In Section 5, we study a collection of well-known constructions for the minimal DFA. Finally, we give further details on related work in Section 6. For space reasons, missing proofs are deferred to the extended version of this paper [9].

## 2 Preliminaries

**Languages.** Let $\Sigma$ be a finite nonempty *alphabet* of symbols. Given a word $w \in \Sigma^*$, $w^R$ denotes the *reverse* of $w$. Given a language $L \subseteq \Sigma^*$, $L^R \stackrel{\text{def}}{=} \{w^R \mid w \in L\}$ denotes the *reverse language* of $L$. We denote by $L^c$ the *complement* of the language $L$. The *left (resp. right) quotient* of L by a word $u$ is defined as the language $u^{-1}L \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid ux \in L\}$ (resp. $Lu^{-1} \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid xu \in L\}$).

**Automata.** A *(nondeterministic) finite-state automaton* (NFA for short), or simply *automaton*, is a 5-tuple $\mathcal{N} = (Q, \Sigma, \delta, I, F)$, where $Q$ is a finite set of *states*, $\Sigma$ is an alphabet, $I \subseteq Q$ are the *initial* states, $F \subseteq Q$ are the *final* states, and $\delta : Q \times \Sigma \to \wp(Q)$ is the *transition* function. We denote the *extended transition function* from $\Sigma$ to $\Sigma^*$ by $\hat{\delta}$. Given $S, T \subseteq Q$, $W^{\mathcal{N}}_{S,T} \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \exists q \in S, q' \in T : q' \in \hat{\delta}(q, w)\}$. In particular, when $S = \{q\}$ and $T = F$, we define *the right language* of state $q$ as $W^{\mathcal{N}}_{q,F}$. Likewise, when $S = I$ and $T = \{q\}$, we define the *left language* of state $q$ as $W^{\mathcal{N}}_{I,q}$. We define $\text{post}^{\mathcal{N}}_w(S) \stackrel{\text{def}}{=} \{q \in Q \mid w \in W^{\mathcal{N}}_{S,q}\}$ and $\text{pre}^{\mathcal{N}}_w(S) \stackrel{\text{def}}{=} \{q \in Q \mid w \in W^{\mathcal{N}}_{q,S}\}$. In general, we omit the automaton $\mathcal{N}$ from the superscript when it is clear from the context. We say that a state $q$ is *unreachable* iff $W^{\mathcal{N}}_{I,q} = \emptyset$ and we say that $q$ is *empty* iff $W^{\mathcal{N}}_{q,F} = \emptyset$. Finally, note that $\mathcal{L}(\mathcal{N}) = \bigcup_{q \in I} W^{\mathcal{N}}_{q,F} = \bigcup_{q \in F} W^{\mathcal{N}}_{I,q} = W^{\mathcal{N}}_{I,F}$.

Given an NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$, the *reverse NFA* for $\mathcal{N}$, denoted by $\mathcal{N}^R$, is defined as $\mathcal{N}^R = (Q, \Sigma, \delta_r, F, I)$ where $q \in \delta_r(q', a)$ iff $q' \in \delta(q, a)$. Clearly, $\mathcal{L}(\mathcal{N})^R = \mathcal{L}(\mathcal{N}^R)$.

A *deterministic finite-state automaton* (DFA for short) is an NFA such that, $I = \{q_0\}$, and, for every state $q \in Q$ and every symbol $a \in \Sigma$, there exists exactly one $q' \in Q$ such that $\delta(q, a) = q'$. According to this definition, DFAs are always *complete*, i.e., they define a transition for each state and input symbol. In general, we denote NFAs by $\mathcal{N}$, using $\mathcal{D}$ for DFAs when the distinction is important. A *co-deterministic finite-state automata* (co-DFA for short) is an NFA $\mathcal{N}$ such that $\mathcal{N}^R$ is deterministic. In this case, co-DFAs are always *co-complete*, i.e., for each target state $q'$ and each input symbol, there exists a source state $q$ such that $\delta(q, a) = q'$. Recall that, given an NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$, the well-known *subset construction* builds a DFA $\mathcal{D} = (\wp(Q), \Sigma, \delta_d, \{I\}, F_d)$ where $F_d = \{S \in \wp(Q) \mid S \cap F \neq \emptyset\}$ and $\delta_d(S, a) = \{q' \mid \exists q \in S, q' \in \delta(q, a)\}$ for every $a \in \Sigma$, that accepts the same language as $\mathcal{N}$ [11]. Given an NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$, we denote by $\mathcal{N}^D$ the DFA that results from applying the subset construction to $\mathcal{N}$ where only subsets (including the empty subset) that are reachable from the initial subset of $\mathcal{N}^D$ are used. Then, $\mathcal{N}^D$ possibly contains empty states but no state is unreachable. A DFA for the language $\mathcal{L}(\mathcal{N})$ is *minimal*, denoted by $\mathcal{N}^{DM}$, if it has no unreachable states and no two states have the same right language. The minimal DFA for a regular language is unique modulo isomorphism.

**Equivalence Relations and Partitions.** Recall that an *equivalence relation* on a set $X$ is a binary relation $\sim$ that is reflexive, symmetric and transitive. Every equivalence relation $\sim$ on $X$ induces a *partition* $P_\sim$ of $X$, i.e., a family $P_\sim = \{B_i\}_{i \in \mathcal{I}} \subseteq \wp(X)$ of subsets of $X$, with $\mathcal{I} \subseteq \mathbb{N}$, such that:

**(i)** $B_i \neq \emptyset$ for all $i \in \mathcal{I}$;
**(ii)** $B_i \cap B_j = \emptyset$, for all $i, j \in \mathcal{I}$ with $i \neq j$; and
**(iii)** $X = \bigcup_{i \in \mathcal{I}} B_i$.

We say that a partition is *finite* when $\mathcal{I}$ is finite. Each $B_i$ is called a *block* of the partition. Given $u \in X$, then $P_\sim(u)$ denotes the unique block that contains $u$ and corresponds to the *equivalence class* $u$ w.r.t. $\sim$, $P_\sim(u) \stackrel{\text{def}}{=} \{v \in X \mid u \sim v\}$. This definition can be extended in a natural way to a set $S \subseteq X$ as $P_\sim(S) \stackrel{\text{def}}{=} \bigcup_{u \in S} P_\sim(u)$. We say that the partition $P_\sim$

*represents precisely* $S$ iff $P_\sim(S) = S$. An equivalence relation $\sim$ is *of finite index* iff $\sim$ defines a finite number of equivalence classes, i.e., the induced partition $P_\sim$ is finite. In the following, we will always consider equivalence relations of finite index, i.e., finite partitions.

Finally, denote $Part(X)$ the set of partitions of $X$. We use the standard refinement ordering $\preceq$ between partitions: let $P_1, P_2 \in Part(X)$, then $P_1 \preceq P_2$ iff for every $B \in P_1$ there exists $B' \in P_2$ such that $B \subseteq B'$. Then, we say that $P_1$ is *finer than* $P_2$ (or equivalently, $P_2$ is *coarser than* $P_1$). Given $P_1, P_2 \in Part(X)$, define the *coarsest common refinement*, denoted by $P_1 \curlywedge P_2$, as the coarsest partition $P \in Part(X)$ that is finer than both $P_1$ and $P_2$. Likewise, define the *finest common coarsening*, denoted by $P_1 \curlyvee P_2$, as the finest partition $P$ that is coarser than both $P_1$ and $P_2$. Recall that $(Part(X), \preceq, \curlyvee, \curlywedge)$ is a complete lattice where the top (coarsest) element is $\{X\}$ and the bottom (finest) element is $\{\{x\} \mid x \in X\}$.

## 3 Automata Constructions from Congruences

We will consider equivalence relations on $\Sigma^*$ (and their corresponding partitions) with good properties w.r.t. concatenation. An equivalence relation $\sim$ is a *right (resp. left) congruence* iff for all $u, v \in \Sigma^*$, we have that $u \sim v \Rightarrow ua \sim va$, for all $a \in \Sigma$ (resp. $u \sim v \Rightarrow au \sim av$). We will denote right congruences (resp. left congruences) by $\sim^r$ (resp. $\sim^\ell$). The following lemma gives a characterization of right and left congruences.

▶ **Lemma 1.** *The following properties hold:*
1. $\sim^r$ *is a right congruence iff* $P_{\sim^r}(v)u \subseteq P_{\sim^r}(vu)$*, for all* $u, v \in \Sigma^*$.
2. $\sim^\ell$ *is a left congruence iff* $uP_{\sim^\ell}(v) \subseteq P_{\sim^\ell}(uv)$*, for all* $u, v \in \Sigma^*$.

Given a right congruence $\sim^r$ and a regular language $L \subseteq \Sigma^*$ such that $P_{\sim^r}$ represents precisely $L$, i.e., $P_{\sim^r}(L) = L$, the following automata construction recognizes exactly the language $L$ [13].

▶ **Definition 2** (Automata construction $\mathsf{H}^r(\sim^r, L)$)**.** *Let* $\sim^r$ *be a right congruence and let* $P_{\sim^r}$ *be the partition induced by* $\sim^r$*. Let* $L \subseteq \Sigma^*$ *be a language. Define the automaton* $\mathsf{H}^r(\sim^r, L) = (Q, \Sigma, \delta, I, F)$ *where* $Q = \{P_{\sim^r}(u) \mid u \in \Sigma^*\}$*,* $I = \{P_{\sim^r}(\varepsilon)\}$*,* $F = \{P_{\sim^r}(u) \mid u \in L\}$*, and* $\delta(P_{\sim^r}(u), a) = P_{\sim^r}(v)$ *iff* $P_{\sim^r}(u)a \subseteq P_{\sim^r}(v)$*, for all* $u, v \in \Sigma^*$ *and* $a \in \Sigma$*.*

▶ Remark 3. Note that $\mathsf{H}^r(\sim^r, L)$ is *finite* since we assume $\sim^r$ is of finite index. Note also that $\mathsf{H}^r(\sim^r, L)$ is a complete *deterministic* finite-state automaton since, for each $u \in \Sigma^*$ and $a \in \Sigma$, there exists *exactly one* block $P_{\sim^r}(v)$ such that $P_{\sim^r}(u)a \subseteq P_{\sim^r}(v)$, which is $P_{\sim^r}(ua)$. Finally, observe that $\mathsf{H}^r(\sim^r, L)$ possibly contains empty states but no state is unreachable.

▶ **Lemma 4.** *Let* $\sim^r$ *be a right congruence and let* $L \subseteq \Sigma^*$ *be a language such that* $P_{\sim^r}(L) = L$*. Then* $\mathcal{L}(\mathsf{H}^r(\sim^r, L)) = L$*.*

Due to the left-right duality between $\sim^\ell$ and $\sim^r$, we can give a similar automata construction such that, given a left congruence $\sim^\ell$ and a language $L \subseteq \Sigma^*$ with $P_{\sim^\ell}(L) = L$, recognizes exactly the language $L$.

▶ **Definition 5** (Automata construction $\mathsf{H}^\ell(\sim^\ell, L)$)**.** *Let* $\sim^\ell$ *be a left congruence and let* $P_{\sim^\ell}$ *be the partition induced by* $\sim^\ell$*. Let* $L \subseteq \Sigma^*$ *be a language. Define the automaton* $\mathsf{H}^\ell(\sim^\ell, L) = (Q, \Sigma, \delta, I, F)$ *where* $Q = \{P_{\sim^\ell}(u) \mid u \in \Sigma^*\}$*,* $I = \{P_{\sim^\ell}(u) \mid u \in L\}$*,* $F = \{P_{\sim^\ell}(\varepsilon)\}$*, and* $P_{\sim^\ell}(v) \in \delta(P_{\sim^\ell}(u), a)$ *iff* $aP_{\sim^\ell}(v) \subseteq P_{\sim^\ell}(u)$*, for all* $u, v \in \Sigma^*$ *and* $a \in \Sigma$*.*

▶ Remark 6. In this case, $\mathsf{H}^\ell(\sim^\ell, L)$ is a co-complete *co-deterministic* finite-state automaton since, for each $v \in \Sigma^*$ and $a \in \Sigma$, there exists *exactly one* block $P_{\sim^\ell}(u)$ such that $aP_{\sim^\ell}(v) \subseteq P_{\sim^\ell}(u)$, which is $P_{\sim^\ell}(av)$. Finally, observe that $\mathsf{H}^\ell(\sim^\ell, L)$ possibly contains unreachable states but no state is empty.

▶ **Lemma 7.** *Let $\sim^\ell$ be a left congruence and let $L \subseteq \Sigma^*$ be a language such that $P_{\sim^\ell}(L) = L$. Then $\mathcal{L}(\mathsf{H}^\ell(\sim^\ell, L)) = L$.*

Lemma 8 shows that $\mathsf{H}^\ell$ and $\mathsf{H}^r$ inherit the left-right duality between $\sim^\ell$ and $\sim^r$.

▶ **Lemma 8.** *Let $\sim^r$ and $\sim^\ell$ be a right and left congruence respectively, and let $L \subseteq \Sigma^*$ be a language. If the following property holds*

$$u \sim^r v \Leftrightarrow u^R \sim^\ell v^R \tag{1}$$

*then $\mathsf{H}^r(\sim^r, L)$ is isomorphic to $\left(\mathsf{H}^\ell(\sim^\ell, L^R)\right)^R$.*

## 4 Language-based Congruences and their Approximation using NFAs

Given a language $L \subseteq \Sigma^*$, we recall the following equivalence relations on $\Sigma^*$, which are often denoted as *Nerode's equivalence relations* (e.g., see [13]).

▶ **Definition 9** (Language-based Equivalences). *Let $u, v \in \Sigma^*$ and let $L \subseteq \Sigma^*$ be a language. Define:*

$$u \sim_L^r v \Leftrightarrow u^{-1}L = v^{-1}L \qquad \text{Right-}\textit{language-based Equivalence} \tag{2}$$

$$u \sim_L^\ell v \Leftrightarrow Lu^{-1} = Lv^{-1} \qquad \text{Left-}\textit{language-based Equivalence} \tag{3}$$

Note that the right and left language-based equivalences defined above are, respectively, right and left *congruences*. Furthermore, when $L$ is a regular language, $\sim_L^r$ and $\sim_L^\ell$ are of *finite index* [6, 13]. Since we are interested in congruences of finite index (or equivalently, finite partitions), we will always assume that $L$ is a regular language over $\Sigma$.

The following result states that, given a language $L$, the right Nerode's equivalence induces the coarsest partition of $\Sigma^*$ which is a right congruence and precisely represents $L$.

▶ **Lemma 10** (de Luca and Varricchio [8]). *Let $L \subseteq \Sigma^*$ be a regular language. Then,*

$$P_{\sim_L^r} = \bigvee \{P_{\sim^r} \mid \sim^r \textit{ is a right congruence and } P_{\sim^r}(L) = L\} \ .$$

In a similar way, one can prove that the same property holds for the left Nerode's equivalence. Therefore, as we shall see, applying the construction $\mathsf{H}$ to these equivalences yields minimal automata. However, computing them becomes unpractical since languages are possibly infinite, even if they are regular. Thus, we will consider congruences based on the states of the NFA-representation of the language which induce finer partitions of $\Sigma^*$ than Nerode's equivalences. In this sense, we say that the automata-based equivalences *approximate* Nerode's equivalences.

▶ **Definition 11** (Automata-based Equivalences). *Let $u, v \in \Sigma^*$ and let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be an NFA. Define:*

$$u \sim_{\mathcal{N}}^r v \Leftrightarrow \mathrm{post}_u^{\mathcal{N}}(I) = \mathrm{post}_v^{\mathcal{N}}(I) \qquad \text{Right-}\textit{automata-based Equivalence} \tag{4}$$

$$u \sim_{\mathcal{N}}^\ell v \Leftrightarrow \mathrm{pre}_u^{\mathcal{N}}(F) = \mathrm{pre}_v^{\mathcal{N}}(F) \qquad \text{Left-}\textit{automata-based Equivalence} \tag{5}$$

Note that the right and left automata-based equivalences defined above are, respectively, right and left *congruences*. Furthermore, they are of *finite index* since each equivalence class is represented by a subset of states of $\mathcal{N}$.

The following result gives a sufficient and necessary condition for the language-based (Definition 9) and the automata-based equivalences (Definition 11) to coincide.

▶ **Lemma 12.** *Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be an automaton with $L = \mathcal{L}(\mathcal{N})$. Then,*

$$\sim_L^r = \sim_{\mathcal{N}}^r \quad \textit{iff} \quad \forall u, v \in \Sigma^*,\ W^{\mathcal{N}}_{\mathrm{post}_u^{\mathcal{N}}(I),F} = W^{\mathcal{N}}_{\mathrm{post}_v^{\mathcal{N}}(I),F} \Leftrightarrow \mathrm{post}_u^{\mathcal{N}}(I) = \mathrm{post}_v^{\mathcal{N}}(I) \ . \tag{6}$$

## 4.1 Automata Constructions

In what follows, we will use $\mathsf{Min}$ and $\mathsf{Det}$ to denote the construction $\mathsf{H}$ when applied, respectively, to the language-based congruences induced by a regular language and the automata-based congruences induced by an NFA.

▶ **Definition 13.** *Let $\mathcal{N}$ be an NFA generating the language $L = \mathcal{L}(\mathcal{N})$. Define:*

$$\mathsf{Min}^r(L) \stackrel{\text{def}}{=} \mathsf{H}^r(\sim_L^r, L) \qquad \mathsf{Det}^r(\mathcal{N}) \stackrel{\text{def}}{=} \mathsf{H}^r(\sim_{\mathcal{N}}^r, L)$$
$$\mathsf{Min}^\ell(L) \stackrel{\text{def}}{=} \mathsf{H}^\ell(\sim_L^\ell, L) \qquad \mathsf{Det}^\ell(\mathcal{N}) \stackrel{\text{def}}{=} \mathsf{H}^\ell(\sim_{\mathcal{N}}^\ell, L) \ .$$

Given an NFA $\mathcal{N}$ generating the language $L = \mathcal{L}(\mathcal{N})$, all constructions in the above definition yield automata generating $L$. However, while the constructions using the right congruences result in DFAs, the constructions relying on left congruences result in co-DFAs. Furthermore, since the pairs of relations (2)-(3) and (4)-(5), from Definition 9 and 11 respectively, are dual, i.e., they satisfy the hypothesis of Lemma 8, it follows that $\mathsf{Min}^\ell(L)$ is isomorphic to $(\mathsf{Min}^r(L^R))^R$ and $\mathsf{Det}^\ell(\mathcal{N})$ is isomorphic to $(\mathsf{Det}^r(\mathcal{N}^R))^R$.

On the other hand, since $\mathsf{Min}^r$ relies on the language-based congruences, the resulting DFA is minimal, which is not guaranteed to occur with $\mathsf{Det}^r$. This easily follows from the fact that the states of the automata constructions are the equivalence classes of the given congruences and there is no right congruence (representing $L$ precisely) that is coarser than the right Nerode's equivalence (see Lemma 10).

Finally, since every co-deterministic automaton satisfies the right-hand side of Equation (6), it follows that determinizing ($\mathsf{Det}^r$) a co-deterministic automaton ($\mathsf{Det}^\ell(\mathcal{N})$) results in the minimal DFA ($\mathsf{Min}^r(\mathcal{L}(\mathcal{N}))$), as already proved by Sakarovitch [15, Proposition 3.13].

We formalize all these notions in Theorem 14. Finally, Figure 1 summarizes all these well-known connections between the automata constructions given in Definition 13.

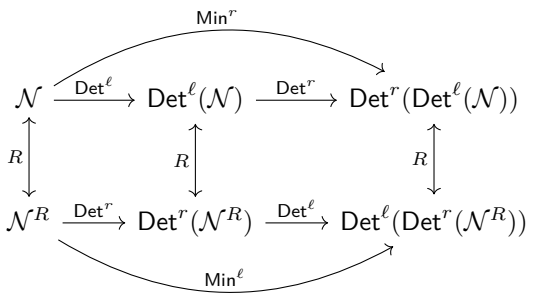
▶ **Theorem 14.** *Let $\mathcal{N}$ be an NFA generating language $L = \mathcal{L}(\mathcal{N})$. Then the following properties hold:*

**(a)** $\mathcal{L}(\mathsf{Min}^r(L)) = \mathcal{L}(\mathsf{Min}^\ell(L)) = L = \mathcal{L}(\mathsf{Det}^r(\mathcal{N})) = \mathcal{L}(\mathsf{Det}^\ell(\mathcal{N}))$.
**(b)** $\mathsf{Min}^r(L)$ *is isomorphic to the minimal deterministic automaton for* $L$.
**(c)** $\mathsf{Det}^r(\mathcal{N})$ *is isomorphic to* $\mathcal{N}^D$.
**(d)** $\mathsf{Min}^\ell(L)$ *is isomorphic to* $(\mathsf{Min}^r(L^R))^R$.
**(e)** $\mathsf{Det}^\ell(\mathcal{N})$ *is isomorphic to* $(\mathsf{Det}^r(\mathcal{N}^R))^R$.
**(f)** $\mathsf{Det}^r(\mathsf{Det}^\ell(\mathcal{N}))$ *is isomorphic to* $\mathsf{Min}^r(L)$.

# 5 A Congruence-based Perspective on Known Algorithms

We can find in the literature several well-known independent techniques for the construction of minimal DFAs. Some of these techniques are based on refining a state partition of an input DFA, such as Moore's algorithm [14], while others directly manipulate an input NFA, such as the double-reversal method [4]. Now, we establish a connection between these algorithms through Theorem 16, which gives a necessary and sufficient condition on an NFA so that determinizing it yields the minimal DFA.

▶ **Lemma 15.** *Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be an NFA with $L = \mathcal{L}(\mathcal{N})$ and $\sim_L^r = \sim_{\mathcal{N}}^r$. Then $\forall q \in Q,\ P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) = W_{I,q}^{\mathcal{N}}$.*




The upper part of the diagram follows from Theorem 14 (f). Both squares of the diagram follow from Theorem 14 (e), which states that $\mathsf{Det}^\ell(\mathcal{N})$ is isomorphic to $(\mathsf{Det}^r(\mathcal{N}^R))^R$. Finally, the bottom curved arc follows from Theorem 14 (d). Incidentally, the diagram shows a new relation which follows from the left-right dualities between $\sim_L^\ell$ and $\sim_L^r$, and $\sim_{\mathcal{N}}^\ell$ and $\sim_{\mathcal{N}}^r$: $\mathsf{Min}^\ell(\mathcal{L}(\mathcal{N}^R))$ is isomorphic to $\mathsf{Det}^\ell(\mathsf{Det}^r(\mathcal{N}^R))$.

**Figure 1** Relations between the constructions $\mathsf{Det}^\ell, \mathsf{Det}^r, \mathsf{Min}^\ell$ and $\mathsf{Min}^r$. Note that constructions $\mathsf{Min}^r$ and $\mathsf{Min}^\ell$ are applied to the language generated by the automaton in the origin of the labeled arrow, while constructions $\mathsf{Det}^r$ and $\mathsf{Det}^\ell$ are applied directly to the automaton.

**Proof.**

$$\begin{aligned}
P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) = & \quad [\text{By definition of } P_{\sim_L^r}] \\
\{w \in \Sigma^* \mid \exists u \in W_{I,q}^{\mathcal{N}},\ w^{-1}L = u^{-1}L\} = & \quad [\text{Since } \sim_L^r = \sim_{\mathcal{N}}^r] \\
\{w \in \Sigma^* \mid \exists u \in W_{I,q}^{\mathcal{N}},\ \mathrm{post}_w^{\mathcal{N}}(I) = \mathrm{post}_u^{\mathcal{N}}(I)\} \subseteq & \quad [u \in W_{I,q}^{\mathcal{N}} \iff q \in \mathrm{post}_u^{\mathcal{N}}(I)] \\
\{w \in \Sigma^* \mid q \in \mathrm{post}_w^{\mathcal{N}}(I)\} = & \quad [\text{By definition of } W_{I,q}^{\mathcal{N}}] \\
W_{I,q}^{\mathcal{N}} \ . &
\end{aligned}$$

By reflexivity of $\sim_L^r$, we conclude that $P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) = W_{I,q}^{\mathcal{N}}$. ◀

▶ **Theorem 16.** *Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be an NFA with $L = \mathcal{L}(\mathcal{N})$. Then $\mathsf{Det}^r(\mathcal{N})$ is the minimal DFA for $L$ iff $\forall q \in Q,\ P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) = W_{I,q}^{\mathcal{N}}$.*

**Proof.** Assume $\mathsf{Det}^r(\mathcal{N})$ is minimal. Then $P_{\sim_{\mathcal{N}}^r}(u) = P_{\sim_L^r}(u)$ for all $u \in \Sigma^*$, i.e. $\sim_L^r = \sim_{\mathcal{N}}^r$. It follows from Lemma 15 that $P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) = W_{I,q}^{\mathcal{N}}$.

Now, assume that $P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) = W_{I,q}^{\mathcal{N}}$, for each $q \in Q$. Then, for every $u \in \Sigma^*$,

$$P_{\sim_{\mathcal{N}}^r}(u) = \bigcap_{q \in \mathrm{post}_u^{\mathcal{N}}(I)} W_{I,q}^{\mathcal{N}} \cap \bigcap_{q \notin \mathrm{post}_u^{\mathcal{N}}(I)} (W_{I,q}^{\mathcal{N}})^c = \bigcap_{q \in \mathrm{post}_u^{\mathcal{N}}(I)} P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) \cap \bigcap_{q \notin \mathrm{post}_u^{\mathcal{N}}(I)} (P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}))^c$$

where the first equality follows by rewriting $P_{\sim_{\mathcal{N}}^r}(u) = \{v \in \Sigma^* \mid \mathrm{post}_u^{\mathcal{N}}(I) = \mathrm{post}_v^{\mathcal{N}}(I)\}$ with universal quantifiers, hence intersections, and the last equality follows from the initial assumption $P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) = W_{I,q}^{\mathcal{N}}$.

It follows that $P_{\sim_{\mathcal{N}}^r}(u)$ is a *union* of blocks of $P_{\sim_L^r}$. Recall that $\sim_L^r$ induces the coarsest right congruence such that $P_{\sim_L^r}(L) = L$ (Lemma 10). Since $\sim_{\mathcal{N}}^r$ is a right congruence satisfying $P_{\sim_{\mathcal{N}}^r}(L) = L$ then $P_{\sim_{\mathcal{N}}^r} \preccurlyeq P_{\sim_L^r}$. Therefore, $P_{\sim_{\mathcal{N}}^r}(u)$ necessarily corresponds to one single block of $P_{\sim_L^r}$, namely, $P_{\sim_L^r}(u)$. Since $P_{\sim_{\mathcal{N}}^r}(u) = P_{\sim_L^r}(u)$ for each $u \in \Sigma^*$, we conclude that $\mathsf{Det}^r(\mathcal{N}) = \mathsf{Min}^r(L)$. ◀

## 5.1 Double-reversal Method

In this section we give a simple proof of the well-known double-reversal minimization algorithm of Brzozowski [4] using Theorem 16. Note that, since $\mathsf{Det}^r(\mathcal{N})$ is isomorphic to $\mathcal{N}^D$ by Theorem 14 (c), the following result coincides with that of Brzozowski.

▶ **Theorem 17** ([4]). *Let $\mathcal{N}$ be an NFA. Then* $\mathsf{Det}^r((\mathsf{Det}^r(\mathcal{N}^R))^R)$ *is isomorphic to the minimal DFA for* $\mathcal{L}(\mathcal{N})$.

**Proof.** Let $L = \mathcal{L}(\mathcal{N})$. By definition, $\mathcal{N}' = (\mathsf{Det}^r(\mathcal{N}^R))^R$ is a co-DFA and, therefore, satisfies the condition on the right-hand side of Equation (6). It follows from Lemma 12 that $\sim_L^r = \sim_{\mathcal{N}'}^r$ which, by Lemma 15 and Theorem 16, implies that $\mathsf{Det}^r(\mathcal{N}')$ is minimal. ◀

Note that Theorem 17 can be inferred from Figure 1 by following the path starting at $\mathcal{N}$, labeled with $R - \mathsf{Det}^r - R - \mathsf{Det}^r$ and ending in $\mathsf{Min}^r(\mathcal{L}(\mathcal{N}))$.

## 5.2 Generalization of the Double-reversal Method

Brzozowski and Tamm [5] generalized the double-reversal algorithm by defining a necessary and sufficient condition on an NFA which guarantees that the determinized automaton is minimal. They introduced the notion of *atomic* NFA and showed that $\mathcal{N}^D$ is minimal iff $\mathcal{N}^R$ is atomic. We shall show that this result is equivalent to Theorem 16 due to the left-right duality between the language-based equivalences (Lemma 8).

▶ **Definition 18** (Atom [5]). *Let $L$ be a regular language. Let $\{K_i \mid 0 \leq i \leq n-1\}$ be the set of left quotients of $L$. An* atom *is any non-empty intersection of the form $\widetilde{K_0} \cap \widetilde{K_1} \cap \ldots \cap \widetilde{K_{n-1}}$, where each $\widetilde{K_i}$ is either $K_i$ or $K_i^c$.*

This notion of atom coincides with that of equivalence class for the left language-based congruence $\sim_L^\ell$. This was first noticed by Iván [12].

▶ **Lemma 19.** *Let $L$ be a regular language. Then for every $u \in \Sigma^*$,*

$$P_{\sim_L^\ell}(u) = \bigcap_{\substack{u \in w^{-1}L \\ w \in \Sigma^*}} w^{-1}L \;\cap\; \bigcap_{\substack{u \notin w^{-1}L \\ w \in \Sigma^*}} (w^{-1}L)^c \ .$$

▶ **Definition 20** (Atomic NFA [5]). *An NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ is* atomic *iff for every state $q \in Q$, the right language $W_{q,F}^{\mathcal{N}}$ is a union of atoms of $\mathcal{L}(\mathcal{N})$.*

It follows from Lemma 19 that the set of atoms of a language $L$ corresponds to the partition $P_{\sim_L^\ell}$. Therefore, a set $S \subseteq \Sigma^*$ is a union of atoms iff $P_{\sim_L^\ell}(S) = S$. This property, together with Definition 20, shows that an NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ with $L = \mathcal{L}(\mathcal{N})$ is *atomic* iff

$$\forall q \in Q,\ P_{\sim_L^\ell}(W_{q,F}^{\mathcal{N}}) = W_{q,F}^{\mathcal{N}} \ . \tag{7}$$

We are now in condition to give an alternative proof of the generalization of Brzozowski and Tamm [5] relying on Theorem 16.

▶ **Lemma 21.** *Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be an NFA with $L = \mathcal{L}(\mathcal{N})$. Then $\mathcal{N}^R$ is atomic iff* $\mathsf{Det}^r(\mathcal{N})$ *is the minimal DFA for $L$.*

**Proof.** Let $\mathcal{N}^R = (Q, \Sigma, \delta_r, F, I)$ and $L^R = \mathcal{L}(\mathcal{N}^R)$. Then,

$$\begin{aligned}
\forall q \in Q,\ P_{\sim_{L^R}^\ell}(W_{q,I}^{\mathcal{N}^R}) = W_{q,I}^{\mathcal{N}^R} \iff\ & [\text{By } A = B \Leftrightarrow A^R = B^R] \\
\forall q \in Q,\ \left(P_{\sim_{L^R}^\ell}(W_{q,I}^{\mathcal{N}^R})\right)^R = \left(W_{q,I}^{\mathcal{N}^R}\right)^R \iff\ & [\text{By } u \sim_L^\ell v \Leftrightarrow u^R \sim_{L^R}^r v^R] \\
\forall q \in Q,\ P_{\sim_L^r}\left(\left(W_{q,I}^{\mathcal{N}^R}\right)^R\right) = \left(W_{q,I}^{\mathcal{N}^R}\right)^R \iff\ & [\text{By } \left(W_{q,I}^{\mathcal{N}^R}\right)^R = W_{I,q}^{\mathcal{N}}] \\
\forall q \in Q,\ P_{\sim_L^r}(W_{I,q}^{\mathcal{N}}) = W_{I,q}^{\mathcal{N}} \ .
\end{aligned}$$

It follows from Theorem 16 that $\mathsf{Det}^r(\mathcal{N})$ is minimal. ◀

We conclude this section by collecting all the conditions described so far that guarantee that determinizing an automaton yields the minimal DFA.

▶ **Corollary 22.** *Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be an NFA with $L = \mathcal{L}(\mathcal{N})$. The following are equivalent:*

**(a)** $\mathsf{Det}^r(\mathcal{N})$ *is minimal.*
**(b)** $\sim^r_{\mathcal{N}} = \sim^r_L$.
**(c)** $\forall u, v \in \Sigma^*,\ W^{\mathcal{N}}_{\mathrm{post}^{\mathcal{N}}_u(I),F} = W^{\mathcal{N}}_{\mathrm{post}^{\mathcal{N}}_v(I),F} \Leftrightarrow \mathrm{post}^{\mathcal{N}}_u(I) = \mathrm{post}^{\mathcal{N}}_v(I)$.
**(d)** $\forall q \in Q,\ P_{\sim^r_L}(W^{\mathcal{N}}_{I,q}) = W^{\mathcal{N}}_{I,q}$.
**(e)** $\mathcal{N}^R$ *is atomic.*

## 5.3 Moore's Algorithm

Given a DFA $\mathcal{D}$, Moore [14] builds the minimal DFA for the language $L = \mathcal{L}(\mathcal{D})$ by removing unreachable states from $\mathcal{D}$ and then performing a stepwise refinement of an initial partition of the set of reachable states of $\mathcal{D}$. Since we are interested in the refinement step, in what follows we assume that all DFAs have no unreachable states. In this section, we will describe Moore's state-partition $\mathcal{Q}^{\mathcal{D}}$ and the right-language-based partition $P_{\sim^r_L}$ as greatest fixpoint computations and show that there exists an isomorphism between the two at each step of the fixpoint computation. In fact, this isomorphism shows that Moore's DFA $M$ satisfies $P_{\sim^r_L}(W^M_{I,q}) = W^M_{I,q}$ for every state $q$. Thus, by Theorem 16, $M$ is isomorphic to $\mathsf{Min}^r(\mathcal{L}(\mathcal{D}))$.

First, we give Moore's algorithm which computes the state-partition that is later used to define Moore's DFA.

**Moore's Algorithm** Algorithm for constructing Moore's partition.

**Data:** DFA $\mathcal{D} = \langle Q, \Sigma, \delta, I, F\rangle$ with $L = \mathcal{L}(\mathcal{D})$.
**Result:** $\mathcal{Q}^{\mathcal{D}} \in Part(Q)$.

```
Q^D := {F, F^c}, Q' := ∅;
while Q^D ≠ Q' do
    Q' := Q^D;
    forall a ∈ Σ do
        Q_a := ⅄_{p∈Q^D} {pre^D_a(p), (pre^D_a(p))^c};
    Q^D := Q^D ⋏ ⅄_{a∈Σ} Q_a;
return Q^D;
```

▶ **Definition 23** (Moore's DFA). *Let $\mathcal{D} = (Q, \Sigma, \delta, I, F)$ be a DFA, and let $\mathcal{Q}^{\mathcal{D}}$ be the partition of $Q$ built by using Moore's algorithm.* Moore's DFA *for $\mathcal{L}(\mathcal{D})$ is $M = (Q^M, \Sigma, \delta^M, I^M, F^M)$ where $Q^M = \mathcal{Q}^{\mathcal{D}}$, $I^M = \{\mathcal{Q}^{\mathcal{D}}(q) \mid q \in I\}$, $F^M = \{\mathcal{Q}^{\mathcal{D}}(q) \mid q \in F\}$ and, for each $S, S' \in Q^M$ and $a \in \Sigma$, we have that $\delta^M(S, a) = S'$ iff $\exists q \in S, q' \in S'$ with $\delta(q, a) = q'$.*

Next, we describe Moore's state-partition $\mathcal{Q}^{\mathcal{D}}$ and the right-language-based partition $P_{\sim^r_L}$ as greatest fixpoint computations and show that there exists an isomorphism between the two at each step of the fixpoint computation.

▶ **Definition 24** (Moore's state-partition). *Let $\mathcal{D} = (Q, \Sigma, \delta, I, F)$ be a DFA. Define* Moore's state-partition w.r.t. $\mathcal{D}$, *denoted by $\mathcal{Q}^{\mathcal{D}}$, as follows.*

$$\mathcal{Q}^{\mathcal{D}} \stackrel{\text{def}}{=} \mathrm{gfp}(\lambda X.\ \bigwedge_{a\in\Sigma, S\in X} \{\mathrm{pre}_a(S), (\mathrm{pre}_a(S))^c\} \curlywedge \{F, F^c\})\ .$$

On the other hand, by Theorem 14 (b), each state of the minimal DFA for $L$ corresponds to an equivalence class of $\sim_L^r$. These equivalence classes can be defined in terms of non-empty intersections of complemented or uncomplemented right quotients of $L$.

▶ **Lemma 25.** *Let $L$ be a regular language. Then, for every $u \in \Sigma^*$,*

$$P_{\sim_L^r}(u) = \bigcap_{\substack{u \in Lw^{-1} \\ w \in \Sigma^*}} Lw^{-1} \cap \bigcap_{\substack{u \notin Lw^{-1} \\ w \in \Sigma^*}} (Lw^{-1})^c \ .$$

It follows from Lemma 25 that $P_{\sim_L^r} = \curlywedge_{w \in \Sigma^*} \{Lw^{-1}, (Lw^{-1})^c\}$, for every regular language $L$ . Thus, $P_{\sim_L^r}$ can also be obtained as a greatest fixpoint computation as follows.

▶ **Lemma 26.** *Let $L$ be a regular language. Then*

$$P_{\sim_L^r} = \mathrm{gfp}(\lambda X. \curlywedge_{a \in \Sigma, B \in X} \{Ba^{-1}, (Ba^{-1})^c\} \curlywedge \{L, L^c\}) \ . \tag{8}$$

The following result shows that, given a DFA $\mathcal{D}$ with $L = \mathcal{L}(\mathcal{D})$, there exists a partition isomorphism between $\mathcal{Q}^{\mathcal{D}}$ and $P_{\sim_L^r}$ at each step of the fixpoint computations given in Definition 24 and Lemma 26 respectively.

▶ **Theorem 27.** *Let $\mathcal{D} = (Q, \Sigma, \delta, I, F)$ be a DFA with $L = \mathcal{L}(\mathcal{D})$ and let $\varphi : \wp(Q) \to \wp(\Sigma^*)$ be a function defined by $\varphi(S) \stackrel{\text{def}}{=} W^{\mathcal{D}}_{I,S}$. Let $\mathcal{Q}^{\mathcal{D}(n)}$ and $P^{(n)}_{\sim_L^r}$ be the $n$-th step of the fixpoint computation of $\mathcal{Q}^{\mathcal{D}}$ (Definition 24) and $P_{\sim_L^r}$ (Lemma 26), respectively. Then, $\varphi$ is an isomorphism between $\mathcal{Q}^{\mathcal{D}(n)}$ and $P^{(n)}_{\sim_L^r}$ for each $n \geq 0$.*

**Proof.** In order to show that $\varphi$ is a partition isomorphism, it suffices to prove that $\varphi$ is a bijective mapping between the partitions. We first show that $\varphi(\mathcal{Q}^{\mathcal{D}(n)}) = P^{(n)}_{\sim_L^r}$, for every $n \geq 0$. Thus, the mapping $\varphi$ is surjective. Secondly, we show that $\varphi$ is an injective mapping from $\mathcal{Q}^{\mathcal{D}(n)}$ to $P^{(n)}_{\sim_L^r}$. Therefore, we conclude that $\varphi$ is a bijection.

To show that $\varphi(\mathcal{Q}^{\mathcal{D}(n)}) = P^{(n)}_{\sim_L^r}$, for each $n \geq 0$, we proceed by induction.

- *Base case:* By definition, $\mathcal{Q}^{\mathcal{D}(0)} = \{F, F^c\}$ and $P^{(0)}_{\sim_L^r} = \{L, L^c\}$. Since $\mathcal{D}$ is deterministic (and complete), it follows that $\varphi(F) = W^{\mathcal{D}}_{I,F} = L$ and $\varphi(F^c) = W^{\mathcal{D}}_{I,F^c} = L^c$.
- *Inductive step:* Before proceeding with the inductive step, we show that the following equations hold for each $a, b \in \Sigma$ and $S, S_i, S_j \in \mathcal{Q}^{\mathcal{D}(n)}$ with $n \geq 0$:

$$\varphi(\mathrm{pre}_a(S)^c) = ((W^{\mathcal{D}}_{I,S})a^{-1})^c \tag{9}$$

$$\varphi(\mathrm{pre}_a(S_i) \cap \mathrm{pre}_b(S_j)) = (W^{\mathcal{D}}_{I,S_i})a^{-1} \cap (W^{\mathcal{D}}_{I,S_j})b^{-1} \ . \tag{10}$$

For each $S \in \mathcal{Q}^{\mathcal{D}(n)}$ and $a \in \Sigma$ we have that:

$$\begin{aligned}
\varphi(\mathrm{pre}_a(S)^c) = & \quad [\text{By definition of } \varphi] \\
W^{\mathcal{D}}_{I,\mathrm{pre}_a(S)^c} = & \quad [I = \{q_0\} \text{ and def. of } W^{\mathcal{D}}_{I,\mathrm{pre}_a(S)^c}] \\
\{w \in \Sigma^* \mid \exists q \in \mathrm{pre}_a(S)^c,\ q = \hat{\delta}(q_0, w)\} = & \quad [\mathcal{D} \text{ is deterministic and complete}] \\
\{w \in \Sigma^* \mid \exists q \in \mathrm{pre}_a(S),\ q = \hat{\delta}(q_0, w)\}^c = & \quad [\text{By definition of } \mathrm{pre}_a(S)] \\
\{w \in \Sigma^* \mid \exists q \in S,\ q = \hat{\delta}(q_0, wa)\}^c = & \quad [\text{By definition of } (W^{\mathcal{D}}_{I,S})a^{-1}] \\
((W^{\mathcal{D}}_{I,S})a^{-1})^c \ . &
\end{aligned}$$

Therefore Equation (9) holds at each step of the fixpoint computation. Consider now Equation (10). Let $S_i, S_j \in \mathcal{Q}^{\mathcal{D}(n)}$. Then,

$$\begin{aligned}
\varphi(\mathrm{pre}_a(S_i) \cap \mathrm{pre}_b(S_j)) = & \quad [\text{By Def. } \varphi] \\
W^{\mathcal{D}}_{I,(\mathrm{pre}_a(S_i)\cap \mathrm{pre}_b(S_j))} = & \quad [I = \{q_0\} \text{ and def. } W_{I,S}] \\
\{w \in \Sigma^* \mid \exists q \in \mathrm{pre}_a(S_i) \cap \mathrm{pre}_b(S_j), q = \hat{\delta}(q_0, w)\} = & \quad [\text{By Def. of } \cap] \\
\{w \in \Sigma^* \mid \exists q \in \mathrm{pre}_a(S_i),\ q \in \mathrm{pre}_b(S_j), q = \hat{\delta}(q_0, w)\} = & \quad [\mathcal{D} \text{ is deterministic}] \\
W^{\mathcal{D}}_{I,\mathrm{pre}_a(S_i)} \cap W^{\mathcal{D}}_{I,\mathrm{pre}_b(S_j)} = & \quad [\text{By Def. of } (W^{\mathcal{D}}_{I,S})a^{-1}] \\
(W^{\mathcal{D}}_{I,S_i})a^{-1} \cap (W_{I,S_j})b^{-1} \ . &
\end{aligned}$$

Therefore Equation (10) holds at each step of the fixpoint computation.
Let us assume that $\varphi\left(\mathcal{Q}^{\mathcal{D}(n)}\right) = P^{(n)}_{\sim^r_L}$ for every $n \leq k$ with $k > 0$. Then,

$$\begin{aligned}
\varphi\big(\mathcal{Q}^{\mathcal{D}(k+1)}\big) = & \quad [\text{By Def. 24 with } X = \mathcal{Q}^{\mathcal{D}(k)}] \\
\varphi\Big(\bigwedge_{a\in\Sigma, S\in X} \{\mathrm{pre}_a(S), \mathrm{pre}_a(S)^c\} \curlywedge \{F, F^c\}\Big) = & \quad [\text{By Eqs. (9), (10) and def. of } \curlywedge] \\
\bigwedge_{\substack{a\in\Sigma \\ \varphi(S)\in\varphi(X)}} \{(W^{\mathcal{D}}_{I,S})a^{-1}, ((W^{\mathcal{D}}_{I,S})a^{-1})^c\} \curlywedge \{L, L^c\} = & \quad [\text{By induction hypothesis, } \varphi(X) = P^{(k)}_{\sim^r_L}] \\
\bigwedge_{a\in\Sigma, B\in X'} \{Ba^{-1}, (Ba^{-1})^c\} \curlywedge \{L, L^c\} = & \quad [\text{By Lemma 26 with } X' = P^{(k)}_{\sim^r_L}] \\
P^{(k+1)}_{\sim^r_L} \ . &
\end{aligned}$$

Finally, since $\mathcal{D}$ is a DFA then, for each $S_i, S_j \in \mathcal{Q}^{\mathcal{D}(n)} (n \geq 0)$ with $S_i \neq S_j$ we have that $W^{\mathcal{D}}_{I,S_i} \neq W^{\mathcal{D}}_{I,S_j}$, i.e., $\varphi(S_i) \neq \varphi(S_j)$. Therefore, $\varphi$ is an injective mapping. ◀

▶ **Corollary 28.** *Let $\mathcal{D}$ be a DFA with $L = \mathcal{L}(\mathcal{D})$. Let $\mathcal{Q}^{\mathcal{D}(n)}$ and $P^{(n)}_{\sim^r_L}$ be the $n$-th step of the fixpoint computation of $\mathcal{Q}^{\mathcal{D}}$ and $P_{\sim^r_L}$ respectively. Then, for each $n \geq 0$,*

$$P^{(n)}_{\sim^r_L}(W^{\mathcal{D}}_{I,S}) = W^{\mathcal{D}}_{I,S} \ , \text{ for each } S \in \mathcal{Q}^{\mathcal{D}(n)} \ .$$

It follows that Moore's DFA $M$, whose set of states corresponds to the state-partition at the end of the execution of Moore's algorithm, satisfies that $\forall q \in Q^M$, $P_{\sim^r_L}(W^M_{I,q}) = W^M_{I,q}$ with $L = \mathcal{L}(M)$. By Theorem 16, we have that $\mathsf{Det}^r(M) (= M$, since $M$ is a DFA) is minimal.

▶ **Theorem 29.** *Let $\mathcal{D}$ be a DFA and $M$ be Moore's DFA for $\mathcal{L}(\mathcal{D})$ as in Definition 23. Then, $M$ is isomorphic to $\mathsf{Min}^r(\mathcal{L}(\mathcal{D}))$.*

Finally, recall that Hopcroft [10] defined a DFA minimization algorithm which offers better performance than Moore's. The ideas used by Hopcroft can be adapted to our framework to devise a new algorithm for computing $P_{\sim^r_L}$. However, by doing so, we could not derive a better explanation than the one provided by Berstel et al. [2].

## 6 Related Work and Conclusions

Brzozowski and Tamm [5] showed that every regular language defines a unique NFA, which they call *átomaton*. The átomaton is built upon the minimal DFA $\mathcal{N}^{DM}$ for the language, defining its states as non-empty intersections of complemented or uncomplemented right languages of $\mathcal{N}^{DM}$, i.e., the atoms of the language. They also observed that the atoms correspond to intersections of complemented or uncomplemented left quotients of the language. Then they proved that the átomaton is isomorphic to the reverse automaton of the minimal deterministic DFA for the reverse language.

Intuitively, the construction of the átomaton based on the right languages of the minimal DFA corresponds to $\mathsf{Det}^{\ell}(\mathcal{N}^{DM})$, while its construction based on left quotients of the language corresponds to $\mathsf{Min}^{\ell}(\mathcal{L}(\mathcal{N}))$.

▶ **Corollary 30.** *Let $\mathcal{N}^{DM}$ be the minimal DFA for a regular language $L$. Then,*
**(a)** $\mathsf{Det}^{\ell}(\mathcal{N}^{DM})$ *is isomorphic to the átomaton of $L$.*
**(b)** $\mathsf{Min}^{\ell}(L)$ *is isomorphic to the átomaton of $L$.*

In the same paper, they also defined the notion of *partial átomaton* which is built upon an NFA $\mathcal{N}$. Each state of the partial atomaton is a non-empty intersection of complemented or uncomplemented right languages of $\mathcal{N}$, i.e., union of atoms of the language. Intuitively, the construction of the partial átomaton corresponds to $\mathsf{Det}^{\ell}(\mathcal{N})$.

▶ **Corollary 31.** *Let $\mathcal{N}$ be an NFA. Then, $\mathsf{Det}^{\ell}(\mathcal{N})$ is isomorphic to the partial átomaton of $\mathcal{N}$.*

Finally, they also presented a number of results [5, Theorem 3] related to the átomaton $\mathcal{A}$ of a minimal DFA $\mathcal{D}$ with $L = \mathcal{L}(\mathcal{D})$:
1. $\mathcal{A}$ is isomorphic to $\mathcal{D}^{RDR}$.
2. $\mathcal{A}^R$ is the minimal DFA for $L^R$
3. $\mathcal{A}^D$ is the minimal DFA for $L$.
4. $\mathcal{A}$ is isomorphic to $\mathcal{N}^{RDMR}$ for every NFA $\mathcal{N}$ accepting $L$.

All these relations can be inferred from Figure 2 which connects all the automata constructions described in this paper together with the constructions introduced by Brzozowski and Tamm. For instance, property 1 corresponds to the path starting at $\mathcal{N}^{DM}$ (the minimal DFA for $\mathcal{L}(\mathcal{N})$), labeled with $R - \mathsf{Det}^{r} - R$, and ending in the átomaton of $\mathcal{L}(\mathcal{N})$. On the other hand, property 4 corresponds to the path starting at $\mathcal{N}$, labeled with $R - \mathsf{Min}^{r} - R$ and ending in the átomaton of $\mathcal{L}(\mathcal{N})$. Finally, the path starting at $\mathcal{N}$, labeled with $R - \mathsf{Det}^{r} - R$ and ending in the partial átomaton of $\mathcal{N}$ shows that the later is isomorphic to $\mathcal{N}^{RDR}$.

In conclusion, we establish a connection between well-known independent minimization methods through Theorem 16. Given a DFA, the left languages of its states form a partition on words, $P$, and thus, each left language is identified by a state. Intuitively, Moore's algorithm merges states to enforce the condition of Theorem 16, which results in merging blocks of $P$ that belong to the same Nerode's equivalence class. Note that Hopcroft's partition refinement method [10] achieves the same goal at the end of its execution though, stepwise, the partition computed may differ from Moore's. On the other hand, any co-deterministic NFA satisfies the right-hand side of Equation (6) hence, by Lemma 15, satisfies the condition of Theorem 16. Therefore, the double-reversal method, which essentially determinizes a co-determinized NFA, yields the minimal DFA. Finally, the left-right duality (Lemma 8) of the language-based equivalences shows that the condition of Theorem 16 is equivalent to that of Brzozowski and Tamm [5].

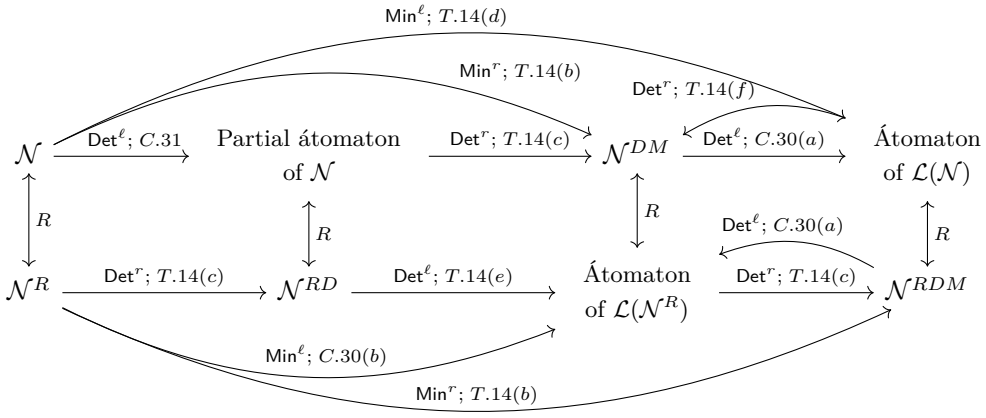



**Figure 2** Extension of the diagram of Figure 1 including the átomaton and the partial átomaton. Recall that $\mathcal{N}^{DM}$ is the minimal DFA for $\mathcal{L}(\mathcal{N})$. The results referenced in the labels are those justifying the output of the operation.

Some of these connections have already been studied in order to offer a better understanding of Brzozowski's double-reversal method [1, 3, 7, 16]. In particular, Adámek et al. [1] and Bonchi et al. [3] offer an alternative view of minimization and determinization methods in a uniform way from a category-theoretical perspective. In contrast, our work revisits these well-known minimization techniques relying on simple language-theoretical notions.

## References


1 Jirí Adámek, Filippo Bonchi, Mathias Hülsbusch, Barbara König, Stefan Milius, and Alexandra Silva. A Coalgebraic Perspective on Minimization and Determinization. In *FoSSaCS*, volume 7213 of *Lecture Notes in Computer Science*, pages 58–73. Springer, 2012.

2 Jean Berstel, Luc Boasson, Olivier Carton, and Isabelle Fagnot. Minimization of automata, 2010. `arXiv:1010.5318`.

3 Filippo Bonchi, Marcello M. Bonsangue, Helle Hvid Hansen, Prakash Panangaden, Jan J. M. M. Rutten, and Alexandra Silva. Algebra-coalgebra duality in Brzozowski's minimization algorithm. *ACM Trans. Comput. Log.*, 15(1):3:1–3:29, 2014.

4 Janusz A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical Theory of Automata*, 12(6):529–561, 1962.

5 Janusz A. Brzozowski and Hellis Tamm. Theory of átomata. *Theor. Comput. Sci.*, 539:13–27, 2014.

6 Julius R. Büchi. *Finite Automata, their Algebras and Grammars - Towards a Theory of Formal Expressions*. Springer, 1989.

7 Jean-Marc Champarnaud, Ahmed Khorsi, and Thomas Paranthoën. Split and join for minimizing: Brzozowski's algorithm. In *Stringology*, pages 96–104. Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University, 2002.

8 Aldo de Luca and Stefano Varricchio. *Finiteness and Regularity in Semigroups and Formal Languages*. Springer Publishing Company, Incorporated, 1st edition, 2011.

9 Pierre Ganty, Elena Gutiérrez, and Pedro Valero. A Congruence-based Perspective on Automata Minimization Algorithms (extended version), 2019. `arXiv:1906.06194`.

10 John E. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In *Theory of machines and computations*, pages 189–196. Elsevier, 1971.

11 John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation - (2. ed.)*. Addison-Wesley-Longman, 2001.

12 Szabolcs Iván. Complexity of atoms, combinatorially. *Inf. Process. Lett.*, 116(5):356–360, 2016.

13 Bakhadyr Khoussainov and Anil Nerode. *Automata Theory and Its Applications*. Birkhauser Boston, Inc., Secaucus, NJ, USA, 2001.

14 Edward F. Moore. Gedanken-experiments on sequential machines. *Automata studies*, 23(1):60–60, 1956.

15 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

16 Manuel Vázquez de Parga, Pedro García, and Damián López. A polynomial double reversal minimization algorithm for deterministic finite automata. *Theor. Comput. Sci.*, 487:17–22, 2013.