Concise Representations of Reversible Automata

Giovanna J. Lavado(✉) and Luca Prigioniero(✉)

Dipartimento di Informatica, Università degli Studi di Milano, Milano, Italy
{lavado,prigioniero}@di.unimi.it

Abstract. We present two concise representations of reversible automata. Both representations have a size which is comparable with the size of the minimum equivalent deterministic automaton and can be exponentially smaller than the size of the explicit representations of corresponding reversible automata. Using those representations it is possible to simulate the computations of reversible automata without explicitly writing down their complete descriptions.

1 Introduction

Reversibility is a fundamental principle in physics: in thermodynamics a transformation is reversible if, after occurring, it can be inverted in order to recover the original state of the system. In the study of computations, reversibility means that each elementary step can be inverted, thus recovering the previous state of the system. In other words, every configuration must admit at most one predecessor. As shown by Landauer, the irreversibility in computation leads to heat dissipations [8], while Toffoli proved that it is ideally possible to build sequential circuits with zero internal power dissipation [12]. This observation suggested to study reversible computations in which there is no loss of information.

Reversibility has been studied on various computational models. In the case of general devices as Turing machines, Bennet proved that each machine can be simulated by a reversible one [1], while Lange, McKenzie, and Tapp proved that each deterministic machine can be simulated by a reversible machine which uses the same amount of space [9]. As a corollary, in the case of a constant amount of space, this implies that each regular language is accepted by a *reversible two-way deterministic finite automaton*. Actually, this result was already proved by Kondacs and Watrous [5]. In the case of *one-way* automata, the situation is different[1]. The class of languages accepted by *reversible automata* is a proper subclass of the class of regular languages. For example, the regular language a^*b^* cannot be accepted by any reversible automaton [11], even if multiple initial states are allowed. Classical automata, namely automata with a single initial state and a set of final states, have been considered in the works by Holzer, Jakobi, and Kutrib [3,6,7]. In particular, in [3] the authors gave a characterization of regular

[1] From now on, we will consider only *one-way automata*. Hence we will omit to specify "one-way" all the times.

G. Pighizzini and C. Câmpeanu (Eds.): DCFS 2017, LNCS 10316, pp. 238–249, 2017.
DOI: 10.1007/978-3-319-60252-3_19

languages which are accepted by reversible automata. This characterization is given in terms of the structure of the minimum deterministic automaton, i.e., the smallest deterministic automaton accepting the language under consideration. Furthermore, they provide an algorithm that, in the case the language is acceptable by a reversible automaton, allows to transform the minimum deterministic automaton into an equivalent reversible automaton, which in the worst case is exponentially larger than the given minimum automaton. In spite of that, the resulting automaton is minimal, namely there are no reversible automata accepting the same language with a smaller number of states. However, it is not necessarily unique, in fact there could exist different reversible automata with the same number of states accepting the same language. Further results concerning minimality and reducibility for reversible automata have been proved in [10].

Due to the above mentioned exponential state gap between deterministic automata and equivalent reversible automata, an explicit representation of a minimal reversible automaton can be exponentially larger than the representation of the corresponding minimum deterministic automaton. However, the minimal reversible automaton produced by the construction in [3] is obtained by creating copies of some parts of the minimum automaton. So, its transition table contains repeated patterns. Thus, it is interesting to investigate whether it is possible to obtain a concise representation of it, by avoiding to repeat those patterns. This is the aim of this paper, where we present two concise representations of reversible automata, which can be used to simulate reversible computations without explicitly writing down the description of the reversible automaton.

The first representation is based on a parameter β which is equal to the maximum number of incoming transitions with a same letter in each state of the given deterministic automaton A. Given β and A it is possible to simulate the computations of a reversible automaton A' equivalent to A, without explicitly representing it. The drawback of this simple representation is that even when the given automaton A is minimum, the simulated reversible automaton A' is not necessarily minimal. This motivates us to search a different concise representation, which exploits a result shown in [10]. The authors have proved that all the minimal reversible automata accepting a language have the same "state structure", in the sense that for each state q of the minimum deterministic automaton they should contain exactly the same number $c(q)$ of states equivalent to q. The second representation is given by the minimum deterministic automaton A accepting the language under consideration and such function c. We prove that, using such representation, it is possible to simulate the behaviour of a minimal reversible automaton equivalent to A without explicitly representing it. Both representations have polynomial size with respect to the size of the given deterministic automaton A and require a precomputation (of the parameter β and of the function c, respectively) which can be performed in polynomial time.

2 Preliminaries

In this section we recall some basic definitions and results useful in the paper. For a detailed exposition, we refer the reader to [4]. Given a set S, let us denote by $\#S$ its cardinality and by 2^S the family of all its subsets. Given an alphabet Σ, $|w|$ denotes the length of a string $w \in \Sigma^*$ and ε the empty string.

A *deterministic automaton* is a tuple $A = (Q, \Sigma, \delta, q_I, F)$, where Q is the set of *states*, Σ is the *input alphabet*, $q_I \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and $\delta : Q \times \Sigma \to Q$ is the partial *transition function.* The function δ can be extended to words in a standard way. The *language accepted* by A is $L(A) = \{w \in \Sigma^* \mid \delta(q_I, w) \in F\}$. The *reverse* transition function of A is the function $\delta^R : Q \times \Sigma \to 2^Q$, with $\delta^R(p, a) = \{q \in Q \mid \delta(q, a) = p\}$. A state $p \in Q$ is *useful* if p is *reachable*, i.e., there is $w \in \Sigma^*$ such that $\delta(q_I, w) = p$, and *productive*, i.e., if there is $w \in \Sigma^*$ such that $\delta(p, w) \in F$. When the set of states Q is finite, the automaton A is said to be a deterministic *finite* automaton (DFA). In this paper we only consider automata with all useful states.

We say that two states $p, q \in Q$ are *equivalent* if for all $w \in \Sigma^*$, $\delta(p, w) \in F$ exactly when $\delta(q, w) \in F$. Two automata A and A' are said to be *equivalent* if they accept the same language, i.e., $L(A) = L(A')$. By *minimal automaton* (in a certain family of automata) we mean an automaton with a minimal number of states. When the minimal automaton is unique (e.g., for the family of all DFAs accepting a certain regular language) we call it *minimum.*

A *strongly connected component* (SCC) C of a DFA $A = (Q, \Sigma, \delta, q_I, F)$ is a maximal subset of Q such that in the transition graph of A there exists a path between every pair of states in C. We introduce the relation $\prec$ on the set of SCCs of A, such that, for two such components C_1 and C_2, $C_1 \prec C_2$ when no state in C_1 can be reached from a state in C_2, but a state in C_2 is reachable from a state in C_1. As usual, if $C_1 \prec C_2$ or $C_1 = C_2$ we write $C_1 \preceq C_2$. It can be verified that $\preceq$ is a partial order.

Given a DFA $A = (Q, \Sigma, \delta, q_I, F)$, a state $r \in Q$ is said to be *irreversible* when $\#\delta^R(r, a) > 1$ for some $a \in \Sigma$, otherwise r is said to be *reversible.* The DFA A is said to be *irreversible* if it contains at least one irreversible state, otherwise A is *reversible* (REV-DFA). As pointed out in [7], the notion of reversibility for a language is related to the computational model under consideration. In this paper we only consider DFAs. Hence, by saying that a language L is *reversible*, we refer to this model, namely we mean that there exists a REV-DFA accepting L. The following result presents a characterization of reversible languages:

Theorem 1 [3, Theorem 2]**.** *Let L be a regular language and $M = (Q, \Sigma, \delta, q_I, F)$ be the minimum* DFA *accepting L. Then, L is accepted by a* REV-DFA *if and only if there do not exist useful states $p, q \in Q$, a letter $a \in \Sigma$, and a string $w \in \Sigma^*$ such that $p \neq q$, $\delta(p, a) = \delta(q, a)$, and $\delta(q, aw) = q$.*

According to Theorem 1, a language L is reversible exactly when the minimum DFA accepting it does not contain the *forbidden pattern* consisting of two transitions on a same letter a entering in a same state r, with one of these transitions arriving from a state in the same SCC as r. An algorithm to convert a minimum DFA M into an equivalent REV-DFA, if any, was obtained in [3]. Furthermore, the resulting REV-DFA is minimal.

3 A Simple Concise Representation

In this section we present our first concise representation. Let us start with a construction for simulating a DFA by an equivalent REV-DFA, in which we use the information about the maximum number of incoming transitions with respect to a same letter in the irreversible states.

Let $A = (Q, \Sigma, \delta, q_I, F)$ be a DFA with all useful states and let β be the maximum number of transitions on a same letter incoming in a state of Q, i.e., $\beta = \max\{\#\delta^R(q, a) \mid q \in Q, a \in \Sigma\}$. We observe that $\beta > 1$ if and only if A is irreversible. We define the following automaton with infinitely many states $A_\infty = (Q', \Sigma, \delta', q'_I, F')$, where $Q' = Q \times \mathbf{N}$, $q'_I = \langle q_I, 0\rangle$, $F' = F \times \mathbf{N}$ and the transitions are defined as follows: let $\delta(q, a) = p$ and $\delta^R(p, a) = \{q_{j_1}, \ldots, q_{j_k}\}$, $k \geq 1$ for $q, p \in Q$, $a \in \Sigma$. For $x \geq 0$:

$$\delta'(\langle q, x\rangle, a) = \begin{cases} \langle p, x\rangle & k = 1 \\ \langle p, x\beta + i - 1\rangle & \text{otherwise} \end{cases} \tag{1}$$

where $i \in \{1, \ldots, k\}$ is such that $q = q_{j_i}$.

Notice that, if $\delta'(\langle q, x\rangle, a) = \langle p, y\rangle$ then $x \leq y$. Roughly speaking, the idea of the construction is to use the second component of the states in A_∞ as label in order to distinguish different copies of a state reached from an irreversible transition in A. The formula used for the second component allow us to obtain this goal, as we will prove in Theorem 2.

We denote by A' the automaton obtained by restricting A_∞ to useful states. We prove that A' simulates A and that it is finite if and only if A does not contain the forbidden pattern.

Theorem 2. *Let $A = (Q, \Sigma, \delta, q_I, F)$ be a* DFA *and $A' = (Q', \Sigma, \delta', q'_I, F')$ be the automaton obtained by applying the above construction to A, restricted to useful states. Then: (a) $L(A') = L(A)$, (b) A' is reversible.*

Proof. (a) It is enough to observe that each state $\langle q, x\rangle \in Q'$ is equivalent to $q \in Q$.

(b) We have to prove that for each $a \in \Sigma$, $\langle \bar{q}_1, x_1\rangle \neq \langle \bar{q}_2, x_2\rangle$ implies that if both $\delta'(\langle \bar{q}_1, x_1\rangle, a)$ and $\delta'(\langle \bar{q}_2, x_2\rangle, a)$ are defined then they are different. Observe that $\delta'(\langle \bar{q}_i, x_i\rangle, a)$ ($i \in \{1, 2\}$) can be undefined only if $\delta(\bar{q}_i, a)$ is undefined. We consider the following cases:
   - If $\bar{q}_1 = \bar{q}_2$ and $x_1 \neq x_2$ then $\delta(\bar{q}_1, a) = \delta(\bar{q}_2, a) = p$ for some $p \in Q$, otherwise M would be nondeterministic. Let $\delta^R(p, a) = \{q_{j_1}, \ldots, q_{j_k}\}$, $k \geq 1$. Then there exists i such that $\bar{q}_1 = \bar{q}_2 = q_{j_i}$. Considering (1), $\delta'(\langle \bar{q}_1, x_1\rangle, a) = \langle p, y_1\rangle$ and $\delta'(\langle \bar{q}_1, x_2\rangle, a) = \langle p, y_2\rangle$. If $k = 1$ then $y_1 = x_1$ and $y_2 = x_2$, otherwise $y_1 = x_1\beta + i - 1$ and $y_2 = x_2\beta + i - 1$.
   Since $x_1 \neq x_2$ we get $y_1 \neq y_2$. Hence, $\langle p, y_1\rangle \neq \langle p, y_2\rangle$.
   - If $\bar{q}_1 \neq \bar{q}_2$ and $\delta(\bar{q}_1, a) = p_1 \neq \delta(\bar{q}_2, a) = p_2$ then, in A' the states $\delta'(\langle \bar{q}_1, x_1\rangle, a) = \langle p_1, y_1\rangle$ and $\delta'(\langle \bar{q}_2, x_2\rangle, a) = \langle p_2, y_2\rangle$ are different regardless of the values of y_1 and y_2.

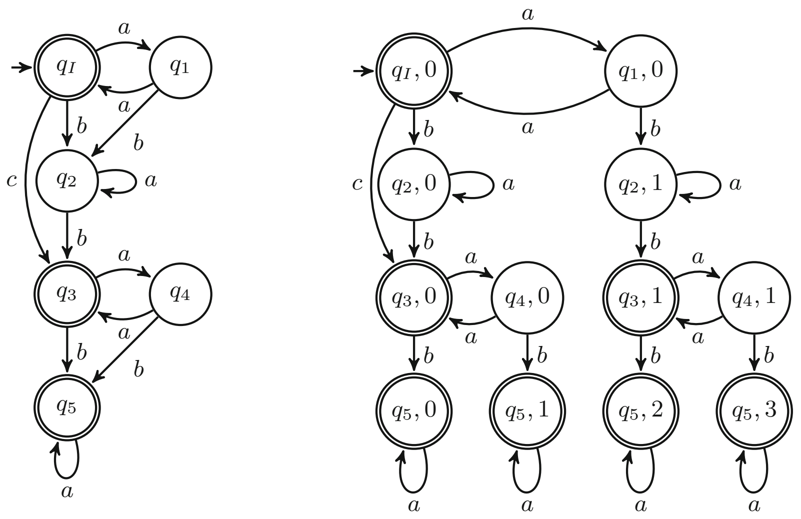

Fig. 1. A DFA where $\beta = 2$ and an equivalent REV-DFA

– If $\bar{q}_1 \neq \bar{q}_2$ and $\delta(\bar{q}_1, a) = \delta(\bar{q}_2, a) = p$, let $\delta^R(p, a) = \{q_{j_1}, \ldots, q_{j_k}\}$, with $k > 1$. Then, there exist i, i', with $i \neq i'$ such that $\bar{q}_1 = q_{j_i}$ and $\bar{q}_2 = q_{j_{i'}}$. Considering (1), $\delta'(\langle \bar{q}_1, x_1 \rangle, a) = \langle p, y_1 \rangle$ and $\delta'(\langle \bar{q}_2, x_2 \rangle, a) = \langle p, y_2 \rangle$, where $y_1 = x_1\beta + i - 1$ and $y_2 = x_2\beta + i' - 1$. In the case $x_1 = x_2$, since $i \neq i'$, we get $y_1 \neq y_2$. In the case $x_1 \neq x_2$, and supposing, without loss of generality, $x_1 > x_2$, we get $\beta x_1 \geq \beta x_2 + \beta$, and hence $\beta x_1 > \beta x_2 + \beta - 1 \geq \beta x_2 + i' - 1$ (notice that $i' \leq \beta$). Then $y_1 = x_1\beta + i - 1 \geq x_1\beta > x_2\beta + i' - 1 = y_2$. This implies that $\langle p, y_1 \rangle \neq \langle p, y_2 \rangle$.

Hence, $\delta'(\langle \bar{q}_1, x_1 \rangle, a) \neq \delta'(\langle \bar{q}_2, x_2 \rangle, a)$. This allow us to conclude that A' is reversible. □

Theorem 3. *The automaton $A' = (Q', \Sigma, \delta, q'_I, F')$ obtained by applying the above construction to a* DFA *$A = (Q, \Sigma, \delta, q_I, F)$ is infinite if and only if A contains the forbidden pattern.*

Two examples related to the previous construction are shown in Figs. 1 and 2, where $\beta = 2$. Let us apply the construction to transform the DFA shown in Fig. 1 through an equivalent REV-DFA. Given for instance $\delta(q_3, b) = q_5$, we have $\delta^R(q_5, b) = \{q_3, q_4\}$, $k > 1$ and $i = 1$. Then $\delta'(\langle q_3, 1 \rangle, b) = \langle q_5, 2 \rangle$. Now we apply the same construction to the DFA in Fig. 2. Given for instance $\delta(q_1, b) = q_2$, we have $\delta^R(q_2, b) = \{q_I, q_1\}$, $k > 1$ and $i = 2$. Then $\delta'(\langle q_1, 0 \rangle, b) = \langle q_2, 1 \rangle$. This time taking $\delta(q_2, a) = q_3$, we have $\delta^R(q_3, a) = \{q_1, q_2\}$, $k > 1$ and $i = 2$. Then $\delta'(\langle q_2, 1 \rangle, a) = \langle q_3, 3 \rangle$. Actually, the simulation of a computation on a string does not require the explicit construction of the automaton A'. In fact, once we have β the computation of the automaton can be obtained, using the transition table of A and (1). For instance on *aba* we have the following steps: $q'_I \xrightarrow{a} \langle q_1, 0 \rangle \xrightarrow{b} \langle q_2, 1 \rangle \xrightarrow{a} \langle q_3, 3 \rangle$.

Notice that the second components in the states having a same q are not necessarily consecutive numbers, in the sense that, it is possible to have some gaps in the numbering as illustrated in Fig. 2 (states of the form $\langle q_3, x\rangle$ in the automaton on the right).

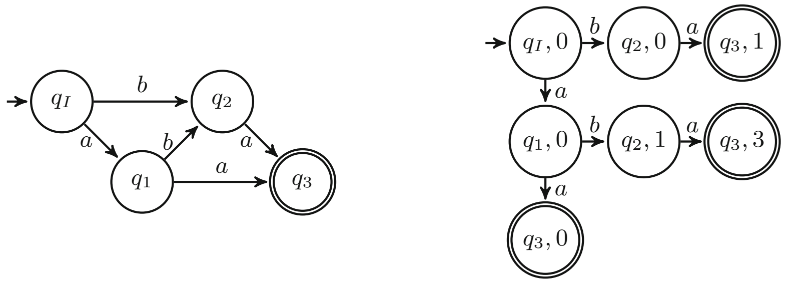

Fig. 2. A DFA where $\beta = 2$ and an equivalent REV-DFA

We point out that the automaton A' can be simulated without explicitly constructing its transition table. Indeed to simulate A' it is enough to know the value of β, which can be computed from the transition table of A, and to follow the transitions of A applying (1) to compute the states reached by A'. So, a concise representation of A' is given by the value of β and the automaton A. We will discuss later in this section how to compute β and how much the value of the second component of a state of A' can be large.

Even when applied to a minimum DFA, the above construction produces a REV-DFA which is not necessarily minimal as illustrated in Figs. 3 and 4: in Fig. 3 a minimum DFA M and an equivalent minimal REV-DFA (obtained by applying the algorithm in [3]) are shown. Notice that the minimal REV-DFA contains five states which are equivalent to q_7. Instead Fig. 4 shows the REV-DFA A' obtained by the above construction (notice that $\beta = 3$). In particular, A' contains six states equivalent to q_7.

In Theorem 3 it has been stated that when a DFA A does not contain the forbidden pattern, the automaton A' obtained by applying the above construction is finite. Furthermore, by Theorem 2, A' is reversible and, as already observed, not necessarily minimal. Hence, it is interesting to know what is the maximum value of the second component in a state of A'. In order to give a bound we will use the following lemmata.

Lemma 4. *If a* DFA *A contains less than two reversible states, then it contains the forbidden pattern.*

Lemma 5. *Let $A' = (Q', \Sigma, \delta', q'_I, F')$ be the automaton obtained by applying the above construction to a* DFA *$A = (Q, \Sigma, \delta, q_I, F)$ which does not contain the forbidden pattern. Given $w \in \Sigma^*$ and $q = \delta(q_I, w)$, consider $q_0, q_1, \ldots, q_m \in Q$, $a_1, a_2, \ldots, a_m \in \Sigma$ such that $w = a_1 \cdots a_m$, $q_0 = q_I$, $q_m = q$, and $q_i = \delta(q_{i-1}, a_i)$, for $i = 1, \ldots, m$. Then $\delta'(q'_I, w) = \langle q'_I, x\rangle$, where $\beta^{k-1} \le x < \beta^k$, and $k = \#\{i \in \{1, \ldots, m\} \mid \#\delta^R(q_i, a_i) > 1\}$.*

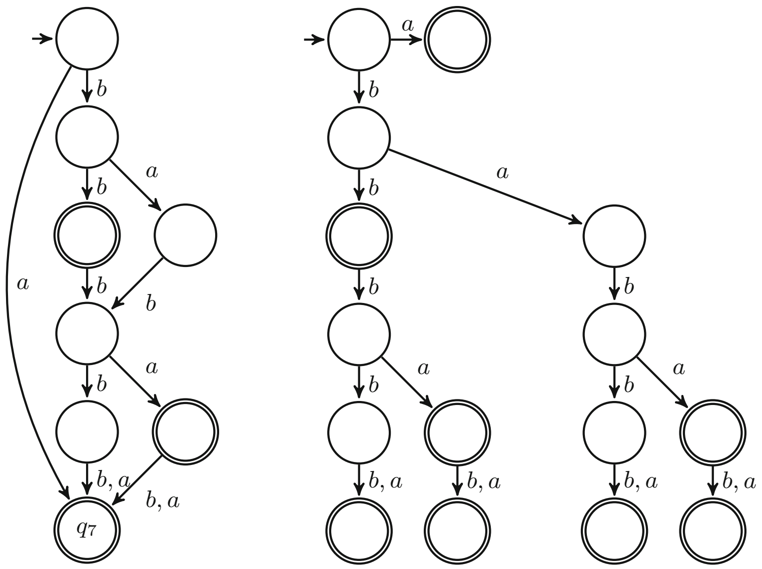

Fig. 3. A minimum DFA and an equivalent minimal REV-DFA

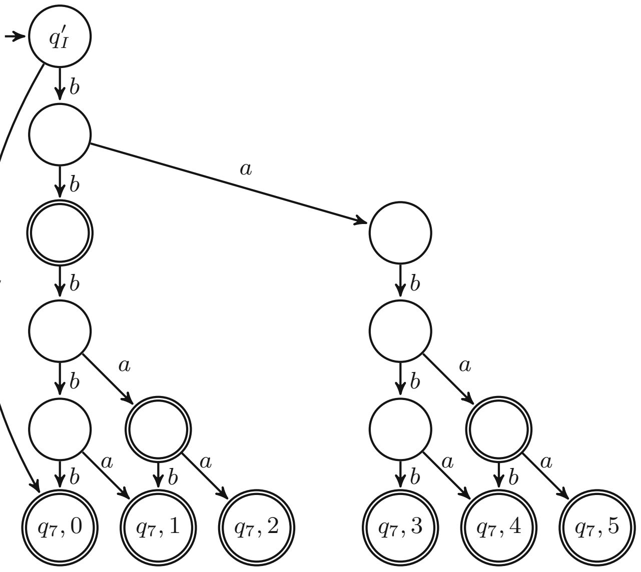

Fig. 4. A nonminimal REV-DFA obtained from the minimum DFA in Fig. 3

As a consequence of Lemma 5, the value of the second components of states of A' is smaller than β^k, where k is the maximum number of irreversible states that on a path from the initial state are reached by "irreversible transitions". Considering Lemma 4, we obtain:

Corollary 6. *If a* DFA A *does not contain the forbidden pattern, then the maximum value of the second component of a state of* A' *obtained by applying the above construction to* A *is smaller than* $\beta^{\#Q-2}$.

Observe that, the maximum value of the second component in a state of A' is reached when in each irreversible state r of A, the maximum number of incoming transitions for a same letter a is equal to β, i.e., $\#\delta^R(r,a) = \beta$. Two examples have been shown in Figs. 1 and 2. The DFA on the left of Fig. 2 has a path from q_I to q_3 reading the string $w = aba$ containing all irreversible states.

We also observe that β has an important role in the construction, so we believe useful to outline how β can be computed. Given a DFA $A = (Q, \Sigma, \delta, q_I, F)$ containing only useful states, we assume that δ resides in a transition table T of size $\#Q \cdot \#\Sigma$. The key observation is that a state is irreversible with respect to a symbol when it occurs more than one time in a column of T. Hence, the problem can be reduced to find the maximum number of occurrences of a state in a column of T, that requires time $\mathcal{O}(\#Q)$ for each symbol. So, the overall time is $\mathcal{O}(\#Q \cdot \#\Sigma)$, which is linear in the cardinality of Q when the alphabet is fixed.

4 Another Concise Representation

The drawback of the representation described in Sect. 3 is that the reversible automaton is not necessarily minimal. In this section we give a different representation which avoids such problem. To state it, some properties related to minimal REV-DFAs are useful. In [3] it has been observed that there are reversible languages having several nonisomorphic minimal REV-DFAs, while in [10, Lemmas 2 and 3] the following result has been proved:

Lemma 7. *Let* $M = (Q, \Sigma, \delta, q_I, F)$ *be the minimum* DFA *accepting a reversible language* L. *Then there exists a function* $c : Q \to \mathbf{N}$ *such that for each state* $q \in Q$, *in any* REV-DFA *equivalent to* M *there are at least* $c(q)$ *copies of* q, *and in any minimal* REV-DFA *equivalent to* M *there are exactly* $c(q)$ *copies of* q. *Furthermore, if* $p, q \in Q$ *are in the same* SCC, *then* $c(p) = c(q)$.

As a consequence of Lemma 7, all the minimal REV-DFAs accepting L have the same "state structure", in the sense that they should contain exactly $c(q)$ states equivalent to the state q of M.

Here we present an easy way to compute the value of $c(q)$, for each $q \in Q$, that is summarized in Algorithm 1. The algorithm firstly transforms the transition graph of M by decomposing it in SCCs, replacing each SCC by a single state, and linking with an edge two SCCs $\mathcal{C}'$ and $\mathcal{C}''$, with $\mathcal{C}' \neq \mathcal{C}''$, if there exist two states $p \in \mathcal{C}'$ and $q \in \mathcal{C}''$ such that $\delta(p, a) = q$ for some symbol a. This is summarized in line 1. After that, the obtained acyclic graph $\mathcal{S}_M$ can be sorted in topological order ($\preceq$, line 2). For further details about these constructions see, for example, [2, Chap. 23].

Then, all $c(q)$ are computed by analyzing the SCCs in topological order in the following way (lines 3–9): when a SCC $\mathcal{C}$ is considered, first of all the algorithm

Algorithm 1. Computation of $c(p)$ for each $p \in Q$.

```
1: Let $\mathcal{S}_M$ be the graph representing the SCCs of the transition graph of $M$
2: Let $\mathcal{L}_{\mathcal{S}_M}$ be the list of the SCCs of $M$ sorted by topological order $\preceq$
3: for all SCCs $\mathcal{C} \in \mathcal{L}_{\mathcal{S}_M}$ do
4:     max_c $\leftarrow 1$
5:     for all states $q \in \mathcal{C}$ do
6:         for all letters $a \in \Sigma$ do
7:             max_c $\leftarrow \max\{$max_c$, \sum_{p \in \delta^R(q,a) \setminus \mathcal{C}} c(p)\}$
8:     for all states $q \in \mathcal{C}$ do
9:         $c(q) \leftarrow$ max_c
10: return $c$
```

computes for each state $q \in \mathcal{C}$ the maximum number of transitions on a same symbol a entering in q from SCCs different from $\mathcal{C}$, where a transition from p to q is *counted* $c(p)$ times, i.e., the algorithm computes $\sum_{p \in \delta^R(q,a) \setminus \mathcal{C}} c(p)$, for all $q \in Q$ and $a \in \Sigma$ and stores the maximum of all such values (lines 5–7). This value is assigned as $c(q)$ to each $q \in \mathcal{C}$ (lines 8–9). Note that, analyzing the SCCs in topological order, the value of $c(p)$ is used for all the states p in the set $\delta^R(q,a) \setminus \mathcal{C}$ when the algorithm is going to compute $c(q)$, for $q \in \mathcal{C}$. Obviously, for each state q in the first SCC $\mathcal{C}_{q_I}$, $\delta^R(q,a) \setminus \mathcal{C}_{q_I} = \emptyset$.

If M does not contain the forbidden pattern, then for each $q \in \mathcal{C}_{q_I}$, $c(q) = 1$ and the set $\delta^R(r,a) \setminus \mathcal{C}_{q_I}$ is empty for any $r \in \mathcal{C}_{q_I}$. As a consequence, the instruction at line 7 does not produce any increment of *max_c* for any state in the SCC under consideration.

It is easy to see that Algorithm 1 works in polynomial time: it is well known that operations at lines 1 and 2 require time $\mathcal{O}(\#V + \#E)$, where V and E are, respectively the set of vertices and the set of edges of the graph under consideration. So, in our case, the time for compute $\mathcal{S}_{\mathcal{M}}$ and $\mathcal{L}_{\mathcal{S}_{\mathcal{M}}}$ is $\mathcal{O}(\#Q)$. From line 3 to 9 the algorithm analyzes, the incoming transitions to each state q. This can be done in time $\mathcal{O}(\#Q)$ assuming that Σ is fixed. So, the Algorithm 1 uses $\mathcal{O}(\#Q)$ time.

The following property will be useful for the construction:

Lemma 8. *Let* $\delta^R(p,a) = \{q_{j_1}, \ldots, q_{j_k}\}$, $k \geq 1$, $p, q_{j_1}, \ldots, q_{j_k} \in Q$, *and* $a \in \Sigma$. *Then* $\sum_{h=1}^{i-1} c(q_{j_h}) + x < c(p)$, *for* $i = 1, \ldots, k$, $0 \leq x < c(q_{j_i})$.

We are now ready to present the construction which leads to our second concise representation. Let $M = (Q, \Sigma, \delta, q_I, F)$ be a minimum DFA accepting a reversible language L. We define the following DFA $A' = (Q', \Sigma, \delta', q'_I, F')$, where $Q' = \{\langle q, x\rangle \mid q \in Q, 0 \leq x < c(q)\}$, $q'_I = \langle q_I, 0\rangle$, $F' = \{\langle q, x\rangle \mid q \in F, 0 \leq x < c(q)\}$, and the transitions are defined as follows: let $\delta(q,a) = p$ and $\delta^R(p,a) = \{q_{j_1}, \ldots, q_{j_k}\}$, $k \geq 1$ for $q, p \in Q$, $a \in \Sigma$. Then:

$$\delta'(\langle q, x\rangle, a) = \langle p, \sum_{h<i} c(q_{j_h}) + x\rangle, \tag{2}$$

where $i \in \{1, \ldots, k\}$ is such that $q = q_{j_i}$ and $0 \leq x < c(q)$.

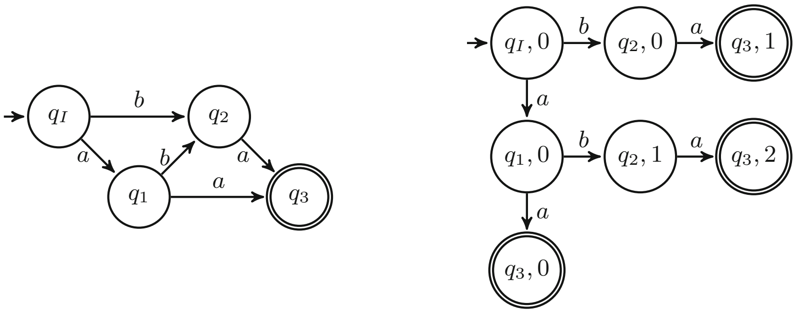

Fig. 5. A DFA and an equivalent minimal REV-DFA

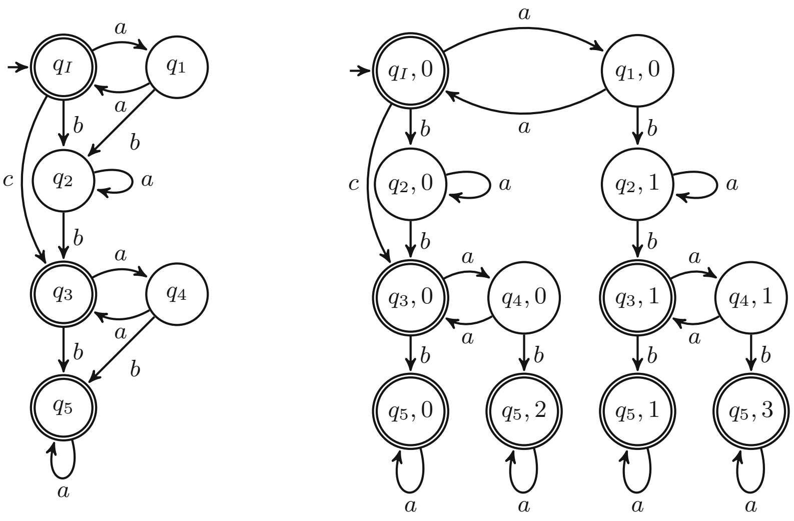

Fig. 6. A DFA and an equivalent minimal REV-DFA

Notice that by Lemma 8, this function δ' is well defined. We will prove that A' is a minimal REV-DFA equivalent to M.

Two examples related to the construction are shown in Figs. 5 and in 6. Let us apply the construction to the minimum DFA M in Fig. 5. The topological order $\preceq$ of the SCCs of M clearly is $q_I \preceq q_1 \preceq q_2 \preceq q_3$ and the number of copies $c(q)$ of a state $q \in Q$ follows the sequence of Fibonacci [3, Example 9]. In particular, $c(q_I) = c(q_1) = 1$, $c(q_2) = 2$, $c(q_3) = 3$. Given for instance $\delta(q_1, b) = q_2$, $\delta^R(q_2, b) = \{q_I, q_1\}$, then $\delta'(\langle q_1, 0\rangle, b) = \langle q_2, c(q_I) + 0\rangle = \langle q_2, 1\rangle$. Now we apply the construction to the minimum DFA M in Fig. 6. Consider the following number of copies $c(q)$: $c(q_I) = c(q_1) = 1$, $c(q_2) = c(q_3) = c(q_4) = 2$ and $c(q_5) = 4$. For instance, given $\delta(q_4, b) = q_5$, $\delta^R(q_5, b) = \{q_3, q_4\}$, we have $\delta'(\langle q_4, 0\rangle, b) = \langle q_5, c(q_3) + 0\rangle = \langle q_5, 2\rangle$.

Note that, even in this case, it is possible to simulate a computation of the REV-DFA A' without explicitly constructing it: given a letter and knowing the state $\langle state, index\rangle$ in which the automaton is, it is always possible to obtain the next state. As example, consider the minimum DFA showed in Fig. 6 (on the left) and the input string *abbab*. So, the computation of the simulated REV-DFA passes through the following states: $\langle q_I, 0\rangle \xrightarrow{a} \langle q_1, 0\rangle \xrightarrow{b} \langle q_2, 1\rangle \xrightarrow{b} \langle q_3, 1\rangle \xrightarrow{a} \langle q_4, 1\rangle \xrightarrow{b} \langle q_5, 3\rangle$.

Theorem 9. *Let $M = (Q, \Sigma, \delta, q_I, F)$ be a minimum* DFA *accepting a reversible language L and let $c(q)$ be the number of states equivalent to $q \in Q$ in any minimal* REV-DFA *equivalent to M. Let $A' = (Q', \Sigma, \delta', q'_I, F')$ be the* DFA *obtained by applying the construction to M, then: (a) $L(A') = L$, (b) A' is reversible, (c) A' is minimal.*

Proof. The proof of (a) and (b) is similar to Theorem 2. To prove (c), we observe that, by Lemma 8, A' contains at most $c(p)$ copies of any state $p \in Q$. However since A' is reversible, by Lemma 7 it should contain at least $c(p)$ copies of p. Hence we conclude that A' is a REV-DFA containing exactly $c(p)$ copies of each state p of the minimum DFA M. According to Lemma 7 this implies that A' is minimal. □

According to the results in this section, given a minimum DFA M, after computing $c(q)$ for each state q of M, we can simulate a minimal REV-DFA A' equivalent to M, without explicitly representing it, starting from the initial state q'_I and using (2) at each step to compute the next state. Since A' can have exponentially many states with respect to M, this avoids to write down a large description.

5 Conclusion

We have presented two concise representations of a reversible automaton A' equivalent to a given DFA. Both of them allow to simulate the REV-DFA without explictly writing down its transition table which, in the worst case, can be exponentially larger. The first representation in Sect. 3 requires an easy precomputation of a parameter β, but the obtained automaton is not necessarily minimal. Instead, the second representation in Sect. 4 requires the more involved precomputation of the function c, but the obtained automaton is minimal. Both precomputations can be done in polynomial time.

Even when the REV-DFA A' obtained from a minimum DFA A in the first representation is not minimal, its size is not too far from the size of a minimal REV-DFA in the following sense. In Lemma 5 we gave an upper bound of the maximum value of the second component in a state of A', i.e., β^{k_p}, where k_p is the maximum number of irreversible states on a path in A from the initial state q_I to p. Since A' is reversible we have $c(p) \leq \beta^{k_p}$ (Lemma 7). Furthermore, in the path at least two copies of each irreversible state should be created to obtain a reversible automaton. Then, $2^{k_p} \leq c(p) \leq \beta^{k_p}$. This implies that A' has a polynomial number of states with respect to the number of states of A if and only if each minimal REV-DFA equivalent to A has a polynomial number of states.

Acknowledgements. We are indebted with the anonymous referees for valuable suggestions.

References

1. Bennett, C.: Logical reversibility of computation. IBM J. Res. Dev. **17**(6), 525–532 (1973)
2. Cormen, T.H.: Introduction to Algorithms. MIT Press, Cambridge (2009)
3. Holzer, M., Jakobi, S., Kutrib, M.: Minimal reversible deterministic finite automata. In: Potapov, I. (ed.) DLT 2015. LNCS, vol. 9168, pp. 276–287. Springer, Cham (2015). doi:10.1007/978-3-319-21500-6_22
4. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Boston (1979)
5. Kondacs, A., Watrous, J.: On the power of quantum finite state automata. In: FOCS, pp. 66–75. IEEE Computer Society (1997)
6. Kutrib, M.: Aspects of reversibility for classical automata. In: Calude, C.S., Freivalds, R., Kazuo, I. (eds.) Computing with New Resources. LNCS, vol. 8808, pp. 83–98. Springer, Cham (2014). doi:10.1007/978-3-319-13350-8_7
7. Kutrib, M.: Reversible and irreversible computations of deterministic finite-state devices. In: Italiano, G.F., Pighizzini, G., Sannella, D.T. (eds.) MFCS 2015. LNCS, vol. 9234, pp. 38–52. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48057-1_3
8. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**(3), 183–191 (1961)
9. Lange, K., McKenzie, P., Tapp, A.: Reversible space equals deterministic space. J. Comput. Syst. Sci. **60**(2), 354–367 (2000)
10. Lavado, G.J., Pighizzini, G., Prigioniero, L.: Minimal and reduced reversible automata. In: Câmpeanu, C., Manea, F., Shallit, J. (eds.) DCFS 2016. LNCS, vol. 9777, pp. 168–179. Springer, Cham (2016). doi:10.1007/978-3-319-41114-9_13
11. Pin, J.-E.: On reversible automata. In: Simon, I. (ed.) LATIN 1992. LNCS, vol. 583, pp. 401–416. Springer, Heidelberg (1992). doi:10.1007/BFb0023844
12. Toffoli, T.: Reversible computing. In: Bakker, J., Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 632–644. Springer, Heidelberg (1980). doi:10.1007/3-540-10003-2_104