Reachability Problems on Partially Lossy Queue Automata

Chris Köcher$^{[0000-0003-4575-9339]}$

Technische Universität Ilmenau, Automata and Logics Group
chris.koecher@tu-ilmenau.de

Abstract. We study the reachability problem for queue automata and lossy queue automata. Concretely, we consider the set of queue contents which are forwards resp. backwards reachable from a given set of queue contents. Here, we prove the preservation of regularity if the queue automaton loops through some special sets of transformations. This is a generalization of the results by Boigelot et al. and Abdulla et al. regarding queue automata looping through a single sequence of transformations. We also prove that our construction is effective and efficient.

Keywords: Partially Lossy Queue · Reachability · Queue Automaton

1 Introduction

Nearly all problems in verification ask whether in a program or automaton one can reach some given configurations from other given configurations. In some computational models this question is decidable, e.g., in finite state machines, pushdown automata [5, 8, 9] or one-counter automata. In some other, mostly Turing-complete computational models this reachability problem is undecidable.

So, for queue automata reachability is undecidable [6], while this problem is decidable for so-called lossy queue automata [1] which are allowed to forget any parts of their content at any time. In this case, for a regular set of configurations, the set of reachable configurations is regular [10] but it is impossible to compute finite automata accepting these sets [14]. Surprisingly, the set of backwards reachable configurations is effectively regular [1], even though this construction is not primitive recursive [7, 15].

Due to the undecidability resp. inefficiency of the reachability problem for reliable and lossy queue automata, one may consider approximations of this problem. One trivial approach is to simulate the automaton's computation step by step until a given configuration (or a given set of configurations) was found. Then, starting from a given set of configurations we simply add or remove a single letter from the queue's contents. An even better and more efficient approach is to consider so-called "meta-transformations" as described in [3, 4]. Such a meta-transformation is a combination of multiple transitions of the queue. In particular, given a loop in the queue's control component we combine iterations of this loop to one big step of the queue automaton. With this trick it is possible to explore infinitely many contents of the queue in a small amount of time.

Considering reliable queue automata, we know from Boigelot et al. [4] that, starting from a regular language of queue contents, the set of reachable queue contents after application of such meta-transformation is effectively regular. A similar result was proven for lossy queue automata by Abdulla et al. in [2].

In this paper we consider a generalization of this result which regards iterations through certain regular languages. Concretely, we consider so-called *read-write independent* sets where for each two words s, t from this language there is another word from this language consisting of the write actions from s and the read actions from t. For these generalized meta-transformations we prove the preservation of regularity of sets of configurations. We will see that our construction is possible in polynomial time.

Additionally, we consider another type of meta-transformations: sets of transformations which are closed under some special (context-sensitive) commutations of the atomic transformations. For such meta-transformations, the set of reachable configurations is also effectively regular. Moreover, if we start from a context-free set of configurations, the set of reachable configurations is effectively context-free, again. Here, both constructions can be carried out in polynomial time.

In this paper, we first prove the stated results for reliable queue automata. Later we consider so-called partially lossy queue automata which were first introduced in [12, 13]. This is a generalization of reliable and lossy queue automata where we can specify which letters can be forgotten at any time. We will see then, that the sets of reachable configurations can be computed from the ones of a reliable queue automaton. Hence, all of our results and the results from [2, 4] do also hold for arbitrary partially lossy queue automata.

2 Preliminaries

2.1 Words and Languages

At first, we have to introduce some basic definitions. To this end, let Γ be an alphabet. A word $v \in \Gamma^*$ is a prefix of $w \in \Gamma^*$ iff $w \in v\Gamma^*$. Similarly, v is a *suffix* of w iff $w \in \Gamma^* v$ and v is an *infix* of w iff $w \in \Gamma^* v \Gamma^*$. The *complementary prefix* (resp. *suffix*) of w wrt. v is the word $w/_v \in \Gamma^*$ (resp. ${}_v\backslash w \in \Gamma^*$) with $w = w/_v \cdot v$ (resp. $w = v \cdot {}_v\backslash w$). The *right quotient* of a language $L \subseteq \Gamma^*$ wrt. $K \subseteq \Gamma^*$ is the language $L/_K = \{u \in \Gamma^* \mid \exists v \in K \colon uv \in L\}$. Similarly, we can define the *left quotient* ${}_K\backslash L$ of L wrt. K.

For a word $w = a_1 a_2 \dots a_n \in \Gamma^*$ we define its *reversal* by $w^{\mathrm{R}} := a_n \dots a_2 a_1$. The *reversal* of a language L is $L^{\mathrm{R}} = \{w^{\mathrm{R}} \mid w \in L\}$. The *shuffle* of L and K is the following language:

$$L \sqcup\!\sqcup K := \left\{ v_1 w_1 v_2 w_2 \dots v_n w_n \,\middle|\, \begin{array}{c} n \in \mathbb{N}, v_i, w_i \in \Gamma^*, \\ v_1 v_2 \dots v_n \in L, w_1 w_2 \dots w_n \in K \end{array} \right\} .$$

The word $v \in \Gamma^*$ is a *subword* of $w \in \Gamma^*$ (denoted by $v \preceq w$) iff $w \in \{v\} \sqcup\!\sqcup \Gamma^*$. Note that the relation $\preceq$ is a partial ordering on Γ^*.

Let $\leq$ be a partial ordering on Γ^* and $L \subseteq \Gamma^*$ be a language. The *upclosure* of L wrt. $\leq$ is $\uparrow_{\leq} L = \{w \in \Gamma^* \mid \exists v \in L \colon v \leq w\}$. Similarly, we can define the *downclosure* $\downarrow_{\leq} L = \{v \in \Gamma^* \mid \exists w \in L \colon v \leq w\}$.

Let $\sim$ be an equivalence relation on Γ^*. The *equivalence class* of $v \in \Gamma^*$ wrt. $\sim$ is $[v]_\sim = \{u \in \Gamma^* \mid u \sim v\}$. A language $L \subseteq \Gamma^*$ is *closed under* $\sim$ if for each $v \in L$ we have $[v]_\sim \subseteq L$.

Let $S \subseteq \Gamma$. Then the *projection* $\pi_S \colon \Gamma^* \to S^*$ to S is the monoid homomorphism induced by $\pi_S(a) = a$ for each $a \in S$ and $\pi_S(a) = \varepsilon$ for each $a \in \Gamma \setminus S$. Additionally, for $w \in \Gamma^*$ we write $|w|_S := |\pi_S(w)|$.

2.2 Automata

A *finite automaton* (*NFA* for short) is a quintuple $\mathcal{A} = (Q, \Gamma, I, \Delta, F)$ where Q is a finite set of states, $I, F \subseteq Q$ are the sets of initial and final states, and $\Delta \subseteq Q \times (\Gamma \cup \{\varepsilon\}) \times Q$ is the transition relation. Then, the *configuration graph* of $\mathcal{A}$ is $\mathfrak{G}_{\mathcal{A}} := (Q, \Delta)$ which is a finite, edge-labeled, and directed graph. For $p, q \in Q$ and $w \in \Gamma^*$ we write $p \xrightarrow{w}_{\mathcal{A}} q$ if there is a w-labeled path in $\mathfrak{G}_{\mathcal{A}}$ from p to q. The *accepted language* of $\mathcal{A}$ is $L(\mathcal{A}) := \{w \in \Gamma^* \mid I \xrightarrow{w}_{\mathcal{A}} F\}$. A language $L \subseteq \Gamma^*$ is *regular*, if there is an NFA $\mathcal{A}$ accepting L. The class of regular languages is effectively closed under Boolean operations, left and right quotients, shuffle, reversal, up- and downclosures wrt. the subword ordering, and projections.

Let $\mathcal{A} = (Q, \Gamma, I, \Delta, F)$ be an NFA, $Q_i, Q_f \subseteq Q$. Then we set $\mathcal{A}_{Q_i \to Q_f} := (Q, \Gamma, Q_i, \Delta, Q_f)$, i.e., $\mathcal{A}_{Q_i \to Q_f}$ is the NFA constructed from $\mathcal{A}$ with initial states Q_i and final states Q_f. For example, we have $L(\mathcal{A}) = \bigcup_{q \in Q} L(\mathcal{A}_{I \to q}) L(\mathcal{A}_{q \to F})$.

A *pushdown automaton* (*PDA* for short) is a tuple $\mathcal{P} = (Q, \Sigma, \Gamma, \#, I, \Delta, F)$ where Q is a finite set of states, Σ and Γ are alphabets, $\# \in \Gamma$ is the stack bottom, $I, F \subseteq Q$ are the sets of initial and final states, and $\Delta \subseteq Q \times \Gamma \times (\Sigma \cup \{\varepsilon\}) \times Q \times \Gamma^*$ is the *finite* transition relation. A *configuration* of $\mathcal{P}$ is a tuple from $\mathrm{Conf}_{\mathcal{P}} := Q \times \Gamma^*$. We denote the set of initial configurations by $\mathrm{Init}_{\mathcal{P}} := I \times \{\#\}$ and the set of accepting configurations by $\mathrm{Final}_{\mathcal{P}} := F \times \Gamma^*$. For $p, q \in Q$, $x, y \in \Gamma^*$, and $a \in \Sigma \cup \{\varepsilon\}$ we write $(p, x) \xrightarrow{a}_{\mathcal{P}} (q, y)$ if there are $X \in \Gamma$ and $\gamma, z \in \Gamma^*$ with $(p, X, a, q, \gamma) \in \Delta$, $x = Xz$, and $y = \gamma z$. Then, $\mathfrak{G}_{\mathcal{P}} := (\mathrm{Conf}_{\mathcal{P}}, \bigcup_{a \in \Sigma \cup \{\varepsilon\}} \xrightarrow{a}_{\mathcal{P}})$ is called the *configuration graph* of $\mathcal{P}$. For $(p, x), (q, y) \in \mathrm{Conf}_{\mathcal{P}}$ and $w \in \Sigma^*$ we write $(p, x) \xrightarrow{w}_{\mathcal{P}} (q, y)$ if there is a w-labeled path from (p, x) to (q, y) in $\mathfrak{G}_{\mathcal{P}}$. The accepted language of $\mathcal{P}$ is $L(\mathcal{P}) = \{w \in \Sigma^* \mid \mathrm{Init}_{\mathcal{P}} \xrightarrow{w}_{\mathcal{P}} \mathrm{Final}_{\mathcal{P}}\}$. A language $L \subseteq \Sigma^*$ is *context-free* if there is a PDA $\mathcal{P}$ with $L = L(\mathcal{P})$.

Let $C \subseteq \mathrm{Conf}_{\mathcal{P}}$ be a set of configurations of $\mathcal{P}$. Then we denote the set of configurations of $\mathcal{P}$ reachable from C by

$$\mathrm{post}^*(C) := \{d \in \mathrm{Conf}_{\mathcal{P}} \mid \exists w \in \Sigma^* \colon C \xrightarrow{w}_{\mathcal{P}} d\}.$$

An NFA $\mathcal{A}$ *recognizes* C if $L(\mathcal{A}) = \{q\gamma \mid (q, \gamma) \in C\}$ holds. In this case we call C *regular*.

Theorem 2.1 ([8,9]). *Let $\mathcal{P}$ be a PDA and $C \subseteq \mathrm{Conf}_{\mathcal{P}}$ be a regular set of configurations. Then $\mathrm{post}^*(C)$ is effectively regular. An NFA recognizing $\mathrm{post}^*(C)$ can be computed from an NFA recognizing C in polynomial time.* □

3 Queues and Queue Automata

In this section we want to recall basic knowledge on queues and queue automata. A queue can store entries from a given alphabet A. Since A is the alphabet of queue entries, the content of a queue is a word from A^*. For any letter $a \in A$ we have two actions: writing of a at the end of the queue (denoted by a) and reading of a from the head of the queue (denoted by $\overline{a}$). We assume that the alphabet $\overline{A}$ containing each such reading operation $\overline{a}$ is a disjoint copy of A. By $\Sigma := A \cup \overline{A}$ we denote the set of all actions on the queue. For $w = a_1a_2\dots a_n \in A^*$ we also write $\overline{w} := \overline{a_1}\,\overline{a_2}\dots\overline{a_n}$ and for $L \subseteq A^*$ we write $\overline{L} := \{\overline{w} \mid w \in L\}$. Formally, we describe the queue's behavior by a function $\circ$ associating a word $v \in A^*$ and a sequence of atomic transformations $t \in \Sigma^*$ with another word $v \circ t \in A^*$ which is the queue's content after application of t on the content v.

Definition 3.1. *Let A be an alphabet and $\bot \notin A$. Then the map $\circ\colon (A^* \cup \{\bot\}) \times \Sigma^* \to (A^* \cup \{\bot\})$ is defined for each $v \in A^*$, $a, b \in A$ with $a \neq b$, and $t \in \Sigma^*$ as follows:*

(1) $v \circ \varepsilon = v$ *(3)* $av \circ \overline{a}t = v \circ t$
(2) $v \circ at = va \circ t$ *(4)* $bv \circ \overline{a}t = \varepsilon \circ \overline{a}t = \bot \circ t = \bot$

We will say "$v \circ t$ is undefined" if $v \circ t = \bot$.

A *queue automaton* is a finite automaton on Σ equipped with such a queue. Considering the expression "$L \circ T$" then $L \subseteq A^*$ is a set of possible queue inputs, $T \subseteq \Sigma^*$ is the set of transformations, and $(L \circ T) \setminus \{\bot\}$ is the set of outputs of the queue automaton. Since T is represented by a finite automaton, the set T is always a regular language in this paper. All in all, we may define our reachability problems as follows:

Definition 3.2. *Let A be an alphabet, $L \subseteq A^*$ be a set of queue contents, and $T \subseteq \Sigma^*$ be a regular set of transformations. The set of queue contents that are reachable from L via T is*

$$\textsc{Reach}(L, T) := (L \circ T) \setminus \{\bot\}$$

and the set of queue contents that can reach L via T is

$$\textsc{BackReach}(L, T) := \{v \in A^* \mid (v \circ T) \cap L \neq \emptyset\}\,.$$

In general, for a recursively enumerable language $L \subseteq A^*$ and a regular set $T \subset \Sigma^*$ the language $\textsc{Reach}(L, T)$ is (effectively) recursively enumerable. Since a finite automaton with a queue can simulate a Turing machine [6], the language $\textsc{Reach}(L, T)$ can be any recursively enumerable language. This is true even if $|L| = 1$ and T is the Kleene closure of a finite set of transformations, i.e., if L and T are somewhat "simple" languages:

Remark 3.3. Let $\mathcal{G} = (N, \Gamma, P, S)$ be a (type-0) grammar and $\# \notin N \cup \Gamma$. The set of possible queue entries is $A := N \cup \Gamma \cup \{\#\}$. We construct the set of transformations $T \subseteq \Sigma^*$ as follows:

$$T := \left(\{\bar{\ell}r \mid (\ell, r) \in P\} \cup \{\overline{a}a \mid a \in N \cup \Gamma \cup \{\#\}\}\right)^* ,$$

i.e., the queue can apply any rule from $\mathcal{G}$ and move any letter from the head to its end. Then we have $\textsc{Reach}(\{\#S\}, T) \cap \#\Gamma^* = \#L(\mathcal{G})$ which can be any recursively enumerable language.

Due to Remark 3.3 there are sets L and T such that $\textsc{Reach}(L, T)$ is undecidable. Therefore, we need some approximation to decide whether a given regular set of configurations can be reached from the regular language L of queue inputs by application of the transformations from T. A trivial approach is to compute $\textsc{Reach}(L, T_n)$ where T_n is the set of prefixes of T of length at most n for increasing $n \in \mathbb{N}$. Unfortunately, this algorithm is not very efficient: consider $L \subseteq A^*$ be a finite language of queue contents and $T \subseteq \Sigma^*$ be a regular language of transformations. Then T_n is finite for any $n \in \mathbb{N}$ and, hence, $\textsc{Reach}(L, T_n)$ is finite as well.

Boigelot et al. improved this trivial approximation in [3, 4] by introduction of so-called *meta-transformations*. This means, that we partition the regular language T into sequences of certain regular languages $S \subseteq \Sigma^*$ such that the mappings $L \mapsto \textsc{Reach}(L, S)$ and $L \mapsto \textsc{BackReach}(L, S)$ can be computed efficiently and preserve regularity. For example, such languages can be a regular language of write actions or a regular language of read actions as considered in the following proposition:

Proposition 3.4. *Let A be an alphabet and $L, T \subseteq A^*$. Then the following statements hold:*
(1) $\textsc{Reach}(L, T) = LT$ *(3)* $\textsc{BackReach}(L, T) = \textsc{Reach}(L^{\mathrm{R}}, \overline{T^{\mathrm{R}}})^{\mathrm{R}}$
(2) $\textsc{Reach}(L, \overline{T}) = {}_{T}\backslash L$ *(4)* $\textsc{BackReach}(L, \overline{T}) = \textsc{Reach}(L^{\mathrm{R}}, T^{\mathrm{R}})^{\mathrm{R}}$ □

Hence, for regular languages $L \subseteq A^*$ and $T \subseteq A^*$ (or $T \subseteq \overline{A}^*$, resp.) we can compute NFAs accepting $\textsc{Reach}(L, T)$ and $\textsc{BackReach}(L, T)$ in polynomial time. In the following two sections we consider two further types of meta-transformations T having efficiently computable mappings $L \mapsto \textsc{Reach}(L, T)$ and $L \mapsto \textsc{BackReach}(L, T)$.

4 Behavioral Equivalence

The first type of meta-transformations we want to consider are languages that are closed under the so-called behavioral equivalence. In this connection, we say that two sequences of transformations have the same behavior if for any queue input the application of both transformations lead to the same output of the queue automaton. Formally, this equivalence is defined as follows:

Definition 4.1. *Let A be an alphabet and $s, t \in \Sigma^*$. Then s and t* behave equivalently *(denoted by $s \equiv t$) if $v \circ s = v \circ t$ for each $v \in A^*$. The relation $\equiv$ is called the* behavioral equivalence.

In other words, we have $s \equiv t$ if the application of s and t have the same effect on any queue's content. For example, for $a \in A$ the sequences $aa\overline{a}$ and $a\overline{a}a$ behave equivalently: let $v \in A^*$ be any queue content. Then we have

$$v \circ aa\overline{a} = vaa \circ \overline{a} = (va \circ \overline{a}) \cdot a = v \circ a\overline{a}a\,.$$

Nevertheless, we have $a\overline{a} \not\equiv \overline{a}a$ since we have $\varepsilon \circ a\overline{a} = \varepsilon \neq \bot = \varepsilon \circ \overline{a}a$.

This equivalence relation was first introduced by Huschenbett et al. in [11]. They proved in this paper that $\equiv$ is a congruence on Σ^* and is described by a finite set of context-sensitive commutations. We recall these commutations in the following theorem:

Theorem 4.2 ([11]). *Let A be an alphabet. Then $\equiv$ is the least congruence on Σ^* satisfying the following equations for each $a, b \in A$:*
(1) $a\overline{b} \equiv \overline{b}a$ if $a \neq b$, *(2) $a\overline{a}\overline{b} \equiv \overline{a}a\overline{b}$, and* *(3) $ba\overline{a} \equiv b\overline{a}a$.* □

The behavioral equivalence was further considered in [12]. Concretely, we regarded the languages which are regular and closed under the behavioral equivalence $\equiv$ and gave some interesting properties of these languages. In that paper, we defined some kind of rational expressions constructing these sets as well as some MSO-logic describing them. In particular, let $T \subseteq \Sigma^*$ be a language that is closed under $\equiv$. Then, we know that T is regular if, and only if, $T \cap \overline{A}^* A^* \overline{A}^*$ is regular.

Example 4.3 ([12]). Let $W, R \subseteq A^*$ be regular languages. Then $[W \sqcup\!\sqcup \overline{R}]_\equiv = W \sqcup\!\sqcup \overline{R}$ is regular and closed under $\equiv$.

Now, let $a \in A$. Then $[(a\overline{a})^*]_\equiv$ is *not* regular since (by Theorem 4.2) we can prove $[(a\overline{a})^*]_\equiv \cap \overline{A}^* A^* \overline{A}^* = \{a^n \overline{a}^n \mid n \in \mathbb{N}\}$ which is not regular.

Let $T \subseteq \Sigma^*$ be regular. Using the equations from Theorem 4.2, we can decide whether T is closed under behavioral equivalence:

Remark 4.4. We can understand the equations from Theorem 4.2 as a finite Thue-system. Then for each rule $(\ell \to r)$ we can compute $T_\ell := T \cap \Sigma^* \ell \Sigma^*$ and $T_r := T \cap \Sigma^* r \Sigma^*$. Applying $(\ell \to r)$ on T_ℓ we obtain a regular language T'_r. Finally, we have to check whether $T_r = T'_r$ holds. The language T is closed under behavioral equivalence if, and only if, all of these tests succeed.

However, given a regular language $T \subseteq \Sigma^*$, it is impossible to compute the closure of T under behavioral equivalence. Moreover, it is undecidable whether the closure of T under $\equiv$ is regular, again [12].

Next, we want to prove that, for meta-transformations $T \subseteq \Sigma^*$ that are regular and closed under behavioral equivalence, the mappings $L \mapsto \textsc{Reach}(L, T)$ and $L \mapsto \textsc{BackReach}(L, T)$ preserve regularity. We do this with the help of some corollary of Theorem 4.2:

Proposition 4.5 ([11, 12]). *Let A be an alphabet and $t \in \Sigma^*$. Then there is $s \in \overline{A}^* A^* \overline{A}^*$ with $s \equiv t$. From a given word t we can compute such a word s in polynomial time.* □

Now, we can prove the main theorem in this section:

Theorem 4.6. *Let A be an alphabet, $L \subseteq A^* \cup \{\perp\}$ be regular, and $T \subseteq \Sigma^*$ be regular and closed under $\equiv$. Then* $\textsc{Reach}(L, T)$ *and* $\textsc{BackReach}(L, T)$ *are effectively regular. In particular, from NFAs accepting L and T we can construct NFAs accepting* $\textsc{Reach}(L, T)$ *and* $\textsc{BackReach}(L, T)$ *in polynomial time.*

Proof. Let $\mathcal{T} = (Q, \Sigma, I, \Delta, F)$ be an NFA with $L(\mathcal{T}) = T$. Since T is closed under $\equiv$ we have, by Proposition 4.5,

$$\textsc{Reach}(L, T) = \textsc{Reach}(L, T \cap \overline{A}^* A^* \overline{A}^*)\,.$$

We partition $T \cap \overline{A} * A^* \overline{A}^*$ as follows: let $p, q \in Q$ be any pair of states. Then we can compute the following three regular languages in polynomial time:

$$\overline{K_1^{p,q}} = L(\mathcal{T}_{I \to p}) \cap \overline{A}^*\,, \quad K_2^{p,q} = L(\mathcal{T}_{p \to q}) \cap A^*\,, \text{ and } \quad \overline{K_3^{p,q}} = L(\mathcal{T}_{q \to F}) \cap \overline{A}^*\,.$$

Then it is easy to see that $T \cap \overline{A}^* A^* \overline{A}^* = \bigcup_{p,q \in Q} \overline{K_1^{p,q}} K_2^{p,q} \overline{K_3^{p,q}}$ holds.

Hence, due to Proposition 3.4 and the closure properties of the class of regular languages $\textsc{Reach}(L, \overline{K_1^{p,q}} K_2^{p,q} \overline{K_3^{p,q}})$ is effectively regular and, hence, $\textsc{Reach}(L, T)$. □

Remark 4.7. Since the left or right quotient of a context-free language with a regular language is again context-free, we can compute, from a PDA accepting L and an NFA accepting T, PDAs accepting $\textsc{Reach}(L, T)$ resp. $\textsc{BackReach}(L, T)$ in polynomial time.

5 Read-Write Independence

Another kind of meta-transformations was first considered in the research of Boigelot et al. [4] (and similarly for lossy queue automata by Abdulla et al. [2]). There, the authors considered queue automata looping through a single sequence of transformations. This means, we consider queue automata having exactly one initial state which is the only final state and there is exactly one labeled path from the initial state back to itself, again.

Concretely, in that paper the authors have proven that beginning with a regular language of queue contents we reach a regular set of queue contents, again. In particular, one can compute infinitely many succeeding queue contents at once in polynomial time. So, a natural question would be, whether this result can be generalized to meta-transformations consisting of multiple such loops starting from a single initial state. Unfortunately, already for queue automata having two loops the set of reachable queue contents is not regular in general:

Example 5.1. Let A be an alphabet and $a, b \in A$ be distinct letters. Then we have $\textsc{Reach}(\{a\}, \{\overline{a}bb, \overline{b}a\}^*) \cap \{a\}^* = \{a^{2^n} \mid n \in \mathbb{N}\}$ which is not even context-free.

Moreover, in Remark 3.3 we have seen that such queue automata consisting of a finite number of such loops are Turing-complete.

In both cases, there are two words $s, t \in T$ having different sub-sequences of write or read actions. One trivial solution would be considering only words having the same sub-sequences of write and read actions. Another even stronger approach is to choose a set T such that independently of which word from $\pi_{\overline{A}}(T)$ we read from the queue, we can write any word from $\pi_A(T)$. In this case, it is impossible that a special queue content can enforce a unique, complicated, infinite run of the queue automaton since we can now write any word from $\pi_A(T)$ at any time into the queue. This can be understood as lifting of a word having sub-sequences of write and read actions to an object having a set of sub-sequences of write and read actions. Formally, we are considering the following sets of transformations:

Definition 5.2. *Let A be an alphabet. A set $T \subseteq \Sigma^*$ is* read-write independent *if for each $s, t \in T$ we have $\pi_A(s)\pi_{\overline{A}}(t) \in T$.*

We may see read-write independent sets as some kind of a Cartesian product of a set of write actions $W \subseteq A^*$ with a set of read actions $\overline{R} \subseteq \overline{A}^*$ where for each element $(w, \overline{r}) \in W \times \overline{R}$ we have the transformation $w\overline{r}$. Some simple read-write independent sets are given in the following example:

Example 5.3. Let $W, R \subseteq A^*$. Then $W\overline{R}$ and $W \sqcup\!\sqcup \overline{R}$ are read-write independent.

Obviously, each language $T \subseteq \Sigma^*$ with $\pi_A(T)\pi_{\overline{A}}(T) \subseteq T$ is read-write independent. Hence, for a given regular language it is clear how to check read-write independency.

In the following we will prove that the mapping $L \mapsto \textsc{Reach}(L, T^*)$, for any regular, read-write independent set $T \subseteq \Sigma^*$, preserves regularity and is computable in polynomial time. But first we focus on a special case where the read-write independent set is the product of a language of write actions W with a language of read actions $\overline{R}$. Here, we consider regular subsets $W\overline{R} \subseteq A^*\overline{A}^*$ where A is some alphabet having a special letter \$ which marks the begin of a word from W and is used for synchronization between writing and reading actions.

Theorem 5.4. *Let A be an alphabet and $\$ \in A$ be some letter. Additionally, let $L \subseteq (A \setminus \{\$\})^*$, $W \subseteq \$(A \setminus \{\$\})^*$, and $R \subseteq A^*$ be regular languages such that $R = \$^* \sqcup\!\sqcup \pi_{A \setminus \{\$\}}(R)$ holds. Then $\textsc{Reach}(L, (W\overline{R})^*)$ is effectively regular. In particular, from NFAs accepting L, W, and R we can construct an NFA accepting $\textsc{Reach}(L, (W\overline{R})^*)$ in polynomial time.*

We prove Theorem 5.4 by reduction to the reachability problem in pushdown automata. A first, trivial idea would be a simple replacement of the queue by a stack, i.e., from the queue's content v we reach w if, and only if, the PDA reaches $w\#$ from $v\#$. Unfortunately, this construction is not possible since our queue automaton modifies its content at both ends which cannot be simulated with a single stack. Hence, we need a more abstract presentation of the queue's contents. To this end, we consider some computation of the queue on a word $v \in L$. So, let $v_0, \dots, v_k \in A^*$ and $\alpha_0, \dots, \alpha_{k-1} \in \Sigma$ with $v_0 = v$, $v_{i+1} = v_i \circ \alpha_i \neq \bot$ for each $0 \leq i < k$, and $\alpha_0 \dots \alpha_{k-1}$ be some prefix of $(W\overline{R})^*$. Consider an NFA $\mathcal{C}$ accepting LW^*. Then, there is some path from an initial state p_0 to a final state q_0 of $\mathcal{C}$ with label v_0. When applying α_0 to v_0 this corresponds either to moving q_0 by one edge labeled with $\alpha_0 \in A$ to state q_1 or to moving p_0 by one edge labeled with a (where $\alpha_0 = \overline{a} \in \overline{A}$) to state p_1. Application of the following actions α_i similarly moves one of the states p_i and q_i by one edge to p_{i+1} resp. q_{i+1}. The result is that v_k is the labeling of some path from p_k to q_k in $\mathcal{C}$. In this sense, we can abstract v_k and its corresponding path in $\mathcal{C}$ by these two states p_k and q_k and a number $n \in \mathbb{N}$ representing the number of W-loops in this path (and hence, the number of words from W to be contained in v_k or the number of \$ on this path). Additionally, since $\alpha_0 \dots \alpha_{n-1}$ is some prefix of $(W\overline{R})^*$ there is some path in an NFA $\mathcal{T}$ accepting $(W\overline{R})^*$ from an initial state to some state s labeled with $\alpha_0 \dots \alpha_{n-1}$.

Alternatively, we can understand the components p_k, q_k, and n as follows: since the queue automaton starts with some word from L, adds a prefix of W^* at the end, and removes some prefix of R^* from the head, the word v_k is some infix of LW^*. Hence, there is a suffix w_0 of $L \cup W$, some words $w_1, \dots, w_{n-1} \in W$, and a prefix w_n of W with $v_k = w_0 w_1 \dots w_{n-1} w_n$. In this case, w_0 is the labeling of some path from p_k to the final states of $\mathcal{C}$ and w_n is the labeling of some path from $\mathcal{C}$'s final states to q_k (cf. Fig. 1).

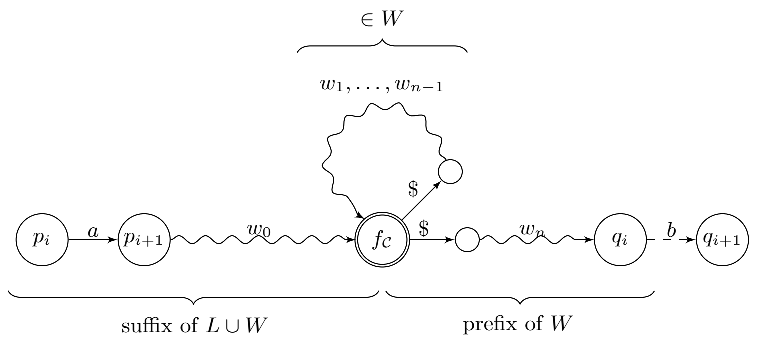

Fig. 1. A path labeled with v_i from p_i to q_i in $\mathcal{C}$ and its three components.

Now, we want to construct a PDA $\mathcal{P}$ which handles exactly the four components named above. In this sense, $\mathcal{P}$'s states contain the three states p_k, q_k, and s of $\mathcal{C}$ and $\mathcal{T}$ and the number n is stored in the stack of $\mathcal{P}$. To this end, let $\mathcal{C} = (Q_\mathcal{C}, A, I_\mathcal{C}, \Delta_\mathcal{C}, F_\mathcal{C})$ be an NFA accepting LW^* (i.e., the possible queue **C**ontents) and $\mathcal{T} = (Q_\mathcal{T}, \Sigma, I_\mathcal{T}, \Delta_\mathcal{T}, F_\mathcal{T})$ be an NFA accepting $(W\overline{R})^*$ (i.e., the possible **T**ransformations). W.l.o.g., we can assume that both, $\mathcal{C}$ and $\mathcal{T}$, are reduced in the sense that each state is reachable from the initial state and can reach some final state. Additionally, we assume that $\mathcal{C}$ and $\mathcal{T}$ have exactly one final state called $f_\mathcal{C}$ resp. $f_\mathcal{T}$. Note that we can compute these two automata in polynomial time from NFAs accepting L, W, and R.

Recall that the queue's content is abstracted by three states from $\mathcal{C}$ and $\mathcal{T}$ and by some natural number. Then the PDA $\mathcal{P} = (Q_\mathcal{P}, \Sigma, \Gamma, \#, I_\mathcal{P}, \Delta_\mathcal{P}, F_\mathcal{P})$ is defined as follows:

- $\Gamma := \{\$, \#\}$
- $Q_\mathcal{P} := Q_\mathcal{C} \times Q_\mathcal{C} \times Q_\mathcal{T}$. Here, the first and second component represent the two states characterizing the queue's content as described above. The third component represents the actions we have already executed on the queue.
- $I_\mathcal{P} := I_\mathcal{C} \times Q_L \times I_\mathcal{T}$ where $Q_L := \{q \in Q_\mathcal{C} \mid \exists v \in L \colon I_\mathcal{C} \xrightarrow{v}_\mathcal{C} q\}$ is the set of states being reachable via L (i.e., the final states of the NFA accepting L)
- $F_\mathcal{P} := Q_\mathcal{C} \times F_\mathcal{C} \times F_\mathcal{T}$
- $\Delta_\mathcal{P}$ contains exactly the following transitions for $a \in A \setminus \{\$\}$, $X \in \Gamma$, $p, p, q, q' \in Q_\mathcal{C}$, and $s, s' \in Q_\mathcal{T}$:
  (I) *Simulate writing of the letter a into the queue*:
  $((p, q, s), X, a, (p, q', s'), X) \in \Delta_\mathcal{P}$ if $(q, a, q') \in \Delta_\mathcal{C}$ and $(s, a, s') \in \Delta_\mathcal{T}$.
  (II) *Simulate writing of the letter \$ into the queue*:
  $((p, q, s), X, \$, (p, q', s'), \$X) \in \Delta_\mathcal{P}$ if $(q, \$, q') \in \Delta_\mathcal{C}$ and $(s, \$, s') \in \Delta_\mathcal{T}$.
  (III) *Simulate reading of the letter a from the queue*:
  $((p, q, s), X, \overline{a}, (p', q, s'), X) \in \Delta_\mathcal{P}$ if $(p, a, p') \in \Delta_\mathcal{C}$ and $(s, \overline{a}, s') \in \Delta_\mathcal{T}$.
  (IV) *Simulate reading of the letter \$ from the queue*:
  $((p, q, s), \$, \overline{\$}, (p', q, s'), \varepsilon) \in \Delta_\mathcal{P}$ if $(p, \$, p') \in \Delta_\mathcal{C}$ and $(s, \overline{\$}, s') \in \Delta_\mathcal{T}$.

It is easy to see that the stack's contents are words from $\$^*\#$ at any time.

Now, we assign the configuration $((p, q, s), \$^n\#)$ to the set of all words being the labeling of some path from p to q in $\mathcal{C}$, containing n appearances of the letter \$ (which marks the beginning of a word from W), and are reachable by application of some infix of $(W\overline{R})^*$ that is the labeling of some path from $I_\mathcal{T}$ to s in $\mathcal{T}$. Since we do not care about the queue's control component and its states, we only focus on the path from p to q in $\mathcal{C}$ and the n appearances of \$. Formally, our assignment is the mapping $[\![.]\!] \colon \mathrm{Conf}_\mathcal{P} \to 2^{A^*}$ with

$$[\![(p, q, s), \$^n\#]\!] = L(\mathcal{C}_{p \to q}) \cap (\$^n \sqcup\!\sqcup (A \setminus \{\$\})^*)$$

for each $p, q \in Q_\mathcal{C}$, $s \in Q_\mathcal{T}$, and $n \in \mathbb{N}$.

Next, we can prove that the set of reachable queue contents coincides with this semantics of the reachable, accepting configurations of the PDA $\mathcal{P}$.

Proposition 5.5. *We have* $\textsc{Reach}(L, (W\overline{R})^*) = \bigcup_{\sigma \in \mathrm{post}^*(\mathrm{Init}_\mathcal{P}) \cap \mathrm{Final}_\mathcal{P}} [\![\sigma]\!]$. □

With Proposition 5.5 in mind, we are ready to prove the effective regularity of the set of reachable configurations of our special queue automata:

Proof (of Theorem 5.4). From Theorem 2.1 we know that $\mathrm{post}^*(\mathrm{Init}_{\mathcal{P}})$ is effectively regular. Let $\mathcal{A}$ be the NFA recognizing $\mathrm{post}^*(\mathrm{Init}_{\mathcal{P}})$ which can be computed in polynomial time. Then the following language is effectively regular as well:

$$K := \bigcup_{(p,q,s)\in F_{\mathcal{P}}} (L(\mathcal{C}_{p\to q}) \cap (\pi_{\$}(L(\mathcal{A}) \cap (p,q,s)\$^*\#) \sqcup\!\sqcup (A \setminus \{\$\})^*))\,.$$

Hence, using Proposition 5.5, we can prove

$$K = \bigcup_{\sigma\in\mathrm{post}^*(\mathrm{Init}_{\mathcal{P}})\cap\mathrm{Final}_{\mathcal{P}}} [\![\sigma]\!] = \textsc{Reach}(L, T^*)\,. \qquad \square$$

Until now we have seen the effective preservation of regularity if our read-write independent set $T \subseteq \Sigma^*$ satisfies a special condition. From this special case we infer now the effective preservation of regularity for arbitrary read-write independent sets.

Theorem 5.6. *Let A be an alphabet, $L \subseteq A^*$ be regular, and $T \subseteq \Sigma^*$ be read-write independent and regular. Then* $\textsc{Reach}(L, T^*)$ *is effectively regular. In particular, from NFAs accepting L and T we can compute an NFA accepting* $\textsc{Reach}(L, T^*)$ *in polynomial time.*

Proof. First, we can prove that for each $t \in T$ and $v \in L$ with $v \circ t \neq \bot$ there is $t' \in \pi_A(T)\pi_{\overline{A}}(T)$ with $v \circ t' = v \circ t$. Hence, we have $\textsc{Reach}(L, T^*) = \textsc{Reach}(L, (\pi_A(T)\pi_{\overline{A}}(T))^*)$. Now, let $\$ \notin A$ be a new letter. Then we set $W := \$\pi_A(T)$ and $R := \pi_{\overline{A}}(T) \sqcup\!\sqcup \overline{\$}^*$ which are effectively regular. By Theorem 5.4 the set $\textsc{Reach}(L, (W\overline{R})^*)$ is effectively regular as well. Finally, we can prove $\textsc{Reach}(L, (W\overline{R})^*) = \textsc{Reach}(L, (\pi_A(T)\pi_{\overline{A}}(T))^*) = \pi_A(\textsc{Reach}(L, T^*))$. $\square$

From Theorem 5.6 and Proposition 3.4 we can infer that also the set of backwards reachable queue contents is effectively regular.

Corollary 5.7. *Let A be an alphabet, $L \subseteq A^*$ be regular, and $T \subseteq \Sigma^*$ be read-write independent and regular. Then* $\textsc{BackReach}(L, T^*)$ *is effectively regular. In particular, from NFAs accepting L and T we can construct an NFA accepting* $\textsc{BackReach}(L, T^*)$ *in polynomial time.* $\square$

Theorem 5.6 can also be used to prove the effective regularity of other language classes. First, with the help of the behavioral equivalence $\equiv$ we can see that the result of [4] is a direct corollary of the result above.

Corollary 5.8. *Let A be an alphabet, $L \subseteq A^*$ be regular, and $T \subseteq \Sigma^*$ be regular. Then* $\textsc{Reach}(L, T^*)$ *and* $\textsc{BackReach}(L, T^*)$ *are effectively regular, if*

(1) $T = \{t\}$ for some $t \in \Sigma^*$ (cf. [4]),
(2) $T = \overline{R_1}W\overline{R_2}$ for some regular sets $W, R_1, R_2 \subseteq A^*$, or
(3) $T \subseteq A^* \cup \overline{A}^*$.

In all of these cases the computation of NFAs accepting $\textsc{Reach}(L, T^*)$ *and* $\textsc{BackReach}(L, T^*)$*, respectively, is possible in polynomial time.*

Proof. First, we prove (1). To this end, let $s = \overline{u}v\overline{w} \in \overline{A}^* A^* \overline{A}^*$ with $s \equiv t$ as in Proposition 4.5. Then we have $t^* \equiv s^* = \overline{u}(v\overline{wu})^*v\overline{w} \cup \{\varepsilon\}$. Since $\{v\overline{wu}\}$ is read-write independent, $\textsc{Reach}(L, t^*) = \textsc{Reach}(L, s^*)$ is effectively regular by Proposition 3.4 and Theorem 5.6.

The proof of (2) is similar to (1).

Finally, we prove (3). Set $S := (T \cap A^*)^*(T \cap \overline{A}^*)^*$. Then S is effectively regular and read-write independent. Additionally, we have $S^* = T^*$ and, hence, $\textsc{Reach}(L, T^*) = \textsc{Reach}(L, S^*)$. □

Note that Corollary 5.8(2) also implies that $\textsc{Reach}(L, (\overline{R}W)^*)$ is effectively regular for some regular languages $W, R \subseteq A^*$.

Though, it is still open whether $\textsc{Reach}(L, T^*)$ is regular for each regular $T \subseteq \Sigma^*$ with $\pi_{\overline{A}}(T)\pi_A(T) \subseteq T$. At least the reduction in Theorem 5.6, where we have de-shuffled the words from T, does not hold in this case. E.g., we have $\textsc{Reach}(\{\varepsilon\}, \{a\overline{a}a, \overline{a}aa\}^*) = a^* \neq \{\varepsilon\} = \textsc{Reach}(\{\varepsilon\}, \{\overline{a}aa\}^*)$. However, we believe that $\textsc{Reach}(L, T^*)$ is effectively (and efficiently) regular for each $T \subseteq \Sigma^*$ such that for each $s, t \in T$ there is $r \in T$ with $\pi_A(r) = \pi_A(s)$ and $\pi_{\overline{A}}(r) = \pi_{\overline{A}}(t)$. Possibly, the construction of our PDA $\mathcal{P}$ can be generalized to this case.

6 Partially Lossy Queues

Until now we have only considered queue automata which are reliable. We can also prove the results from the previous sections for (partially) lossy queue automata. These partially lossy queue automata are queue automata with an additional uncontrollable action which is forgetting parts of its contents that are specified by a so-called lossiness alphabet.

Definition 6.1. *A* lossiness alphabet *is a tuple* $\mathcal{L} = (A, U)$ *where* A *is an alphabet (with* $|A| \geq 2$*) and* $U \subseteq A$.

In this connection, U contains the *unforgettable* letters of the partially lossy queue and $A \setminus U$ contains the *forgettable* letters.

In fact, a partially lossy queue automaton is allowed to forget any letter from $A \setminus U$ in its content at any time. Here, we first consider partially lossy queues with restricted lossiness. Concretely, we consider only the computations of the automata where the queue forgets letters when necessary. That is, if the queue tries to read some letter which is preceded by some forgettable letters.

Formally, the transformations of a restricted partially lossy queue are defined as follows:

Definition 6.2. *Let* $\mathcal{L} = (A, U)$ *be a lossiness alphabet and* $\perp \notin A$*. Then the map* $\circ_{\mathcal{L}} \colon (A^* \cup \{\perp\}) \times \Sigma^* \to (A^* \cup \{\perp\})$ *is defined for each* $v \in A^*$*,* $a, b \in A$*, and* $t \in \Sigma^*$ *as follows:*

(1) $v \circ_{\mathcal{L}} \varepsilon = v$
(2) $v \circ_{\mathcal{L}} at = va \circ t$
(3) $av \circ_{\mathcal{L}} \overline{a}t = v \circ_{\mathcal{L}} t$
(4) $bv \circ_{\mathcal{L}} \overline{a}t = v \circ_{\mathcal{L}} \overline{a}t$ *if* $b \in A \setminus (U \cup \{a\})$
(5) $bv \circ_{\mathcal{L}} \overline{a}t = \perp$ *if* $b \in U \setminus \{a\}$
(6) $\varepsilon \circ_{\mathcal{L}} \overline{a}t = \perp \circ_{\mathcal{L}} t = \perp$

Let $\mathcal{L} = (A, U)$ be a lossiness alphabet and $u, v \in A^*$. We say that v is an *$\mathcal{L}$-subword* of w (denoted by $v \preceq_{\mathcal{L}} w$) if $\pi_U(w) \preceq v \preceq w$ holds. It is easy to see, that $\preceq_{(A,A)}$ is the equality relation and $\preceq_{(A,\emptyset)}$ is the subword relation on A as defined in the preliminaries.

Then a (non-restricted) partially lossy queue with some content $w \in A^*$ may contain any $\mathcal{L}$-subword of w after a single forgetting action. Moreover, for $v \in A^*$ and $t \in \Sigma^*$ with $v \circ_{\mathcal{L}} t \neq \perp$ the set $\downarrow_{\preceq_{\mathcal{L}}}(v \circ_{\mathcal{L}} t)$ is the set of all reachable queue contents after application of the transformation t on v (cf. [13]). Hence, we define our reachability problems as follows:

Definition 6.3. *Let $\mathcal{L} = (A, U)$ be a lossiness alphabet, $L \subseteq A^*$ be a set of queue contents, and $T \subseteq \Sigma^*$ be a regular set of transformations. The set of queue contents that are reachable from L via T is*

$$\textsc{Reach}_{\mathcal{L}}(L, T) := \downarrow_{\preceq_{\mathcal{L}}}((L \circ_{\mathcal{L}} T) \setminus \{\perp\})$$

and the set of queue contents that can reach L via T is

$$\textsc{BackReach}_{\mathcal{L}}(L, T) := \uparrow_{\preceq_{\mathcal{L}}}\{v \in A^* \mid (v \circ_{\mathcal{L}} T) \cap L \neq \emptyset\}\,.$$

Now, we consider fully lossy queues: let $\mathcal{L} = (A, \emptyset)$ be a lossiness alphabet. Then, for regular languages $L \subseteq A^*$ and $T \subseteq \Sigma^*$, the set $\textsc{Reach}_{\mathcal{L}}(L, T)$ has a decidable membership problem [1] and, since it is downwards closed under the subword ordering $\preceq$ [10], it is regular. Though, we cannot compute an NFA accepting this set - even if $L = \{w\}$ [14]. Surprisingly, the set $\textsc{BackReach}_{\mathcal{L}}(L, T)$ is effectively regular [1], but the computation of an NFA accepting this set is not primitive recursive [7, 15].

Hence, again we try to approximate the reachability problem with the help of meta-transformations. To this end, we need the following partial ordering: we say v is a *reduced $\mathcal{L}$-subword* of w (denoted by $v \sqsubseteq_{\mathcal{L}} w$) if, and only if, there are $a_1, \ldots, a_n \in A$ and $w_i \in (A \setminus (U \cup \{a_i\}))^*$ with $v = a_1 \ldots a_n$ and $w = w_1 a_1 \ldots w_n a_n$. Note that $v \sqsubseteq_{\mathcal{L}} w$ implies $v \preceq_{\mathcal{L}} w$ but not vice versa, since for $v \preceq_{\mathcal{L}} w$ it is allowed to add some forgettable letters at the end of v. It is very easy to verify that in the reliable case (i.e., $A = U$) this ordering is the equality relation on A^*. With the help of $\sqsubseteq_{\mathcal{L}}$ we can prove the following statement:

Lemma 6.4. *Let $\mathcal{L} = (A, U)$ be a lossiness alphabet and $v, w, t \in A^*$. Then we have $v \circ_{\mathcal{L}} \overline{t} = w$ if, and only if, there is $s \in A^*$ with $t \sqsubseteq_{\mathcal{L}} s$ and $v = sw$.* □

With the help of Lemma 6.4 we can finally prove the following reductions from reachability in partially lossy queues to reachability in reliable queues:

Proposition 6.5. *Let $\mathcal{L} = (A, U)$ and $\mathcal{K} = (A, A)$ be lossiness alphabets and $L, T \subseteq A^*$. Then the following statements hold:*

(1) $L \circ_{\mathcal{L}} T = L \circ_{\mathcal{K}} T$
(2) $L \circ_{\mathcal{L}} \overline{T} = L \circ_{\mathcal{K}} \overline{\uparrow_{\sqsubseteq_{\mathcal{L}}} T}$
(3) $\textsc{Reach}_{\mathcal{L}}(L, T) = \downarrow_{\preceq_{\mathcal{L}}} \textsc{Reach}_{\mathcal{K}}(L, T)$
(4) $\textsc{Reach}_{\mathcal{L}}(L, \overline{T}) = \downarrow_{\preceq_{\mathcal{L}}} \textsc{Reach}_{\mathcal{K}}(L, \overline{\uparrow_{\sqsubseteq_{\mathcal{L}}} T})$
(5) $\textsc{BackReach}_{\mathcal{L}}(L, T) = \uparrow_{\preceq_{\mathcal{L}}} \textsc{BackReach}_{\mathcal{K}}(L, T)$
(6) $\textsc{BackReach}_{\mathcal{L}}(L, \overline{T}) = \uparrow_{\preceq_{\mathcal{L}}} \textsc{BackReach}_{\mathcal{K}}(L, \overline{\uparrow_{\sqsubseteq_{\mathcal{L}}} T})$ □

Finally, we can prove that our results from the previous sections also hold for arbitrary partially lossy queues:

Theorem 6.6. *Let $\mathcal{L} = (A, U)$ be a lossiness alphabet, $L \subseteq A^*$ be regular, and $T \subseteq \Sigma^*$ be regular. Then $\textsc{Reach}_{\mathcal{L}}(L, T)$ and $\textsc{BackReach}_{\mathcal{L}}(L, T)$ are effectively regular, if*

(1) T is closed under $\equiv_{\mathcal{L}}$ (where $s \equiv_{\mathcal{L}} t$ if $v \circ_{\mathcal{L}} s = v \circ_{\mathcal{L}} t$ for each $v \in A^*$),
(2) $T = S^*$ for some regular, read-write independent $S \subseteq \Sigma^*$,
(3) $T = t^*$ for some $t \in \Sigma^*$ (cf. [2,4]),
(4) $T = (\overline{R_1} W \overline{R_2})^*$ for some regular sets $W, R_1, R_2 \subseteq A^*$, or
(5) $T = S^*$ where $S \subseteq A^* \cup \overline{A}^*$ is regular.

In all of these cases the computation of NFAs accepting $\textsc{Reach}_{\mathcal{L}}(L, T)$ and $\textsc{BackReach}_{\mathcal{L}}(L, T)$, respectively, is possible in polynomial time. □

7 Conclusion

In this paper we introduced so-called partially lossy queue automata (plq automata for short) which are queue automata that are allowed to forget specified parts of their contents at any time. Here, we considered the forwards and backwards reachability problem of such plq automata. Since those automata are Turing-complete (except of the ones allowed to forget everything) Boigelot et al. [4] and Abdulla et al. [1] tried to approximate the reachability problem with the help of so-called meta-transformations. These are regular languages of transformations such that we can easily compute the set of reachable queue contents. Here, we considered two special kinds of meta-transformations:

1. the set of possible sequences of queue transformations is closed under certain (context-sensitive) commutations of the atomic transformations.
2. the plq automaton alternates between writing of words from a regular language and reading of words from another regular language. This is a generalization of the results [2, 4] where the authors considered queue automata looping through a single sequence of transformations.

In both cases we could prove that, starting with a regular language of queue contents the queue reaches a regular set of new contents.

Acknowledgment

The author would like to thank Dietrich Kuske and the anonymous reviewers of this paper for their helpful suggestions to improve this paper.

References

1. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. Information and Computation **127**(2), 91–101 (1996). https://doi.org/10.1006/inco.1996.0053
2. Abdulla, P.A., Collomb-Annichini, A., Bouajjani, A., Jonsson, B.: Using Forward Reachability Analysis for Verification of Lossy Channel Systems. Formal Methods in System Design **25**(1), 39–65 (2004). https://doi.org/10.1023/B:FORM.0000033962.51898.1a
3. Boigelot, B., Godefroid, P.: Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. Formal Methods in System Design **14**(3), 237–255 (1999). https://doi.org/10.1023/A:1008719024240
4. Boigelot, B., Godefroid, P., Willems, B., Wolper, P.: The power of QDDs. In: Static Analysis. Lecture Notes in Computer Science, vol. 1302, pp. 172–186. Springer (1997). https://doi.org/10.1007/BFb0032741
5. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR '97: Concurrency Theory. Lecture Notes in Computer Science, vol. 1243, pp. 135–150. Springer (1997). https://doi.org/10.1007/3-540-63141-0_10
6. Brand, D., Zafiropulo, P.: On Communicating Finite-State Machines. Journal of the ACM **30**(2) (1983). https://doi.org/10.1145/322374.322380
7. Chambart, P., Schnoebelen, P.: The Ordinal Recursive Complexity of Lossy Channel Systems. In: LICS'08. pp. 205–216. IEEE Computer Society Press (2008). https://doi.org/10.1109/LICS.2008.47
8. Esparza, J., Hansel, D., Rossmanith, P., Schwoon, S.: Efficient Algorithms for Model Checking Pushdown Systems. In: Emerson, E.A., Sistla, A.P. (eds.) Computer Aided Verification. Lecture Notes in Computer Science, vol. 1855, pp. 232–247. Springer (2000). https://doi.org/10.1007/10722167_20
9. Finkel, A., Willems, B., Wolper, P.: A Direct Symbolic Approach to Model Checking Pushdown Systems. Electronic Notes in Theoretical Computer Science **9**, 27–37 (1997). https://doi.org/10.1016/S1571-0661(05)80426-8
10. Haines, L.H.: On free monoids partially ordered by embedding. Journal of Combinatorial Theory **6**(1), 94–98 (1969). https://doi.org/10.1016/S0021-9800(69)80111-0
11. Huschenbett, M., Kuske, D., Zetzsche, G.: The monoid of queue actions. Semigroup forum **95**(3), 475–508 (2017). https://doi.org/10.1007/s00233-016-9835-4
12. Köcher, C.: Rational, Recognizable, and Aperiodic Sets in the Partially Lossy Queue Monoid. In: STACS'18. LIPIcs, vol. 96, pp. 45:1–45:14. Dagstuhl Publishing (2018). https://doi.org/10.4230/LIPIcs.STACS.2018.45
13. Köcher, C., Kuske, D., Prianychnykova, O.: The Inclusion Structure of Partially Lossy Queue Monoids and their Trace Submonoids. RAIRO - Theoretical Informatics and Applications **52**(1), 55–86 (2018). https://doi.org/10.1051/ita/2018003
14. Mayr, R.: Undecidable problems in unreliable computations. Theoretical Computer Science **297**(1), 337–354 (2003). https://doi.org/10.1016/S0304-3975(02)00646-1
15. Schnoebelen, P.: Verifying lossy channel systems has nonprimitive recursive complexity. Information Processing Letters **83**(5), 251–261 (2002). https://doi.org/10.1016/S0020-0190(01)00337-4