Acta Cybernetica **13** (1997) 1-21.

Regular expression star-freeness is PSPACE-complete

László Bernátsky [*][†]

Abstract

It is proved that the problem of deciding if a regular expression denotes a star-free language is **PSPACE**-complete. The paper also includes a new proof of the **PSPACE**-completeness of the finite automaton aperiodicity problem.

1 Introduction

Star-free languages form an important subclass of regular languages: they are the ones that can be obtained from the singleton languages by a finite number of applications of the operations of union, complement and product. By Schützenberger's famous theorem [8], a regular language is star-free if and only if its syntactic monoid is aperiodic, or equivalently, if it is recognized by an aperiodic DFA. Moreover, a language is star-free if and only if it can be defined by a first-order formula of a suitable formal language, see [10]. In his 1985 paper [9], Jacques Stern proved that the problem of deciding whether a DFA is aperiodic is **Co-NP**-hard and belongs to **PSPACE**. A few years later Sang Cho and Dung T. Huynh strengthened Stern's result by showing that this problem is in fact **PSPACE**-complete, see [3]. Not knowing about their work I proved the same result while Zoltán Ésik and I were working on the description of the free Conway theories, see [1]. The present paper contains a slightly modified version of my original proof, which rests on the same basic idea as the proof of Cho and Huynh, but uses a different construction, see Construction 4.1. This different construction makes it easy to extend the proof to regular expressions.

2 Definitions and preliminary facts

2.1 Sets and relations

The set of nonnegative integers is denoted $\mathbf{N}$, and ω stands for the set of positive integers. For $n \in \mathbf{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$, so that $[0]$ is another name for the empty set $\emptyset$.

[*]Department of Computer Science, Attila József University, 6720 Szeged, Hungary, E-mail: benny@inf.u-szeged.hu

[†]Supported in part by grant no. T7383 of the National Foundation of Scientific Research of Hungary

The power-set P(S) of a set S consists of all subsets of S, and the direct product $A \times B$ of two sets A and B consists of all pairs (a, b) with $a \in A$ and $b \in B$. A binary relation from A to B is just a subset of $A \times B$, so that P$(A \times B)$ is the set of all binary relations from A to B. The composite of two binary relations $\rho \subseteq A \times B$ and $\rho' \subseteq B \times C$ is the relation

$$\rho \circ \rho' \;=\; \{(a,c) \mid \exists b \in B \;\; (a,b) \in \rho \wedge (b,c) \in \rho'\} \;\subseteq\; A \times C.$$

We use the infix notation $a\rho b$ instead of $(a, b) \in \rho$. Suppose that A' is subset of A, and B' is a subset of B. We write $A'\rho B'$ if there exist $a \in A'$ and $b \in B'$ with $a\rho b$. The image of A' under ρ is denoted $A'\rho$, i.e.,

$$A'\rho \;=\; \{b \in B \mid \exists a \in A' \;\; a\rho b\}.$$

When $A' = \{a\}$ is a singleton, we write $a\rho$ instead of $A'\rho$.

2.2 Words and languages

Suppose that A and B are **alphabets**, i.e., nonempty finite sets. We denote by A^* the set of all finite words over A including the empty word ϵ, while A^+ stands for $A^* \setminus \{\epsilon\}$. The set A^ω is the collection of all infinite words over A. The length of a finite word $u \in A^*$ is denoted $|u|$, and the ith letter of a finite or infinite word $w \in A^* \cup A^\omega$ is denoted w_i. Thus, any finite word $u \in A^*$ can be written as $u_1 u_2 \ldots u_{|u|}$, where each u_i is an element of A. A word $v \in A^*$ is called a **prefix** of a word $u \in A^* \cup A^\omega$ if $u = vw$, for some $w \in A^* \cup A^\omega$. A function $\varphi : A^* \to B^*$ is called a **homomorphism** if $\varphi(uv) = \varphi(u)\varphi(v)$, for all words $u, v \in A^*$. Note that each homomorphism $A^* \to B^*$ is totally determined by its restriction to A.

For the reader's convenience we restate the theorem of Schützenberger.

THEOREM 2.1 (Schützenberger [8]) *A regular language $L \subseteq \Sigma^*$ is star-free if and only if there exists some integer $k \geq 0$ such that*

$$uv^k w \in L \iff uv^{k+1} w \in L,$$

for all words $u, v, w \in \Sigma^*$.

A proof of the following lemma is given in the appendix.

LEMMA 2.1 *Suppose that $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is an alphabet and $l = \lceil \log_2(n + 1) \rceil$. Then there exists an injective homomorphism $\psi : \Sigma^* \to \{0,1\}^*$ satisfying the following conditions.*

- *The ψ-image of each letter $\sigma \in \Sigma$ is a word of length $2l$ beginning with a sequence of l zeros and containing the letter 1. In other words,*

$$\psi(\Sigma) \;\subseteq\; 0^l\{0,1\}^l \setminus \{0^{2l}\}. \qquad (1)$$

- *For all words $u, v, w \in \{0,1\}^*$*

$$uv^{2l}w \in \psi(\Sigma^*) \implies 2l \text{ divides } |v|. \tag{2}$$

- *For all regular languages $L \subseteq \Sigma^*$*

$$L \text{ is star-free} \iff \psi(L) \text{ is star-free.} \tag{3}$$

2.3 Regular expressions

Let $\mathcal{R}$ be the ranked alphabet consisting of the constant symbol $\emptyset$, unary symbols $^*, \sim$ and binary symbols $\cdot, \cup, \cap$. Suppose that Σ is an alphabet such that $\Sigma \cap \mathcal{R} = \emptyset$. For any subset $\mathcal{R}'$ of $\mathcal{R}$, the set $\Sigma \cup \mathcal{R}'$ can be considered as a ranked alphabet in which the elements of Σ have rank 0, and the elements of $\mathcal{R}'$ have the same rank as in $\mathcal{R}$. An **$\mathcal{R}'$-type regular expression over** Σ is a ground $(\Sigma \cup \mathcal{R}')$-term, i.e., a term over the ranked alphabet $\Sigma \cup \mathcal{R}'$ containing no variable symbols. A $\{\emptyset, \cdot, \cup, ^*\}$-type regular expression is simply called a **regular expression**, and an $(\mathcal{R} \setminus \{^*\})$-type regular expression is also called a **star-free regular expression**.

As for the syntactical conventions, we use infix notation for the binary operations $\cup, \cap$ and $\cdot$, postfix notation for *, and we write $\overline{a}$ instead of $\sim a$. The operation symbol $\cdot$ is usually omitted. If $\Sigma' = \{\sigma_1, \ldots, \sigma_n\}$ is a subset of Σ, we simply write Σ' instead of $\sigma_1 \cup \cdots \cup \sigma_n$.

The language $L(E) \subseteq \Sigma^*$ denoted by an $\mathcal{R}$-type regular expression E over Σ is defined in the usual way, see [7]. Note that a regular language $L \subseteq \Sigma^*$ is star-free if and only if it is denoted by some star-free regular expression over Σ.

We recall from [7] that the **star-height** $\mathrm{sh}(E)$ of a regular expression E is defined by

$$\begin{aligned}
\mathrm{sh}(\sigma) &= 0\\
\mathrm{sh}(\emptyset) &= 0\\
\mathrm{sh}(E \cup F) &= \max\{\mathrm{sh}(E), \mathrm{sh}(F)\}\\
\mathrm{sh}(E \cdot F) &= \max\{\mathrm{sh}(E), \mathrm{sh}(F)\}\\
\mathrm{sh}(E^*) &= 1 + \mathrm{sh}(E),
\end{aligned}$$

for all letters $\sigma \in \Sigma$ and regular expressions E, F over Σ.

2.4 Finite automata

Most of our automata-theoretical notations and definitions are adopted from [4].

A **(nondeterministic) finite automaton** (NFA) is represented as a 5-tuple $\mathcal{A} = (Q, \Sigma, \tau, I, F)$, where

- Q is the finite set of states,
- Σ is the input alphabet,

- $\tau : \Sigma \to \mathrm{P}(Q \times Q)$ is the transition function,
- $I \subseteq Q$ is the set of initial states,
- $F \subseteq Q$ is the set of final states.

Note that for each input symbol $\sigma \in \Sigma$, $\tau(\sigma)$ is a binary relation on Q, called the **relation induced by σ in the automaton $\mathcal{A}$**. We prefer the notation $\sigma_{\mathcal{A}}$ to $\tau(\sigma)$. When $u \in \Sigma^*$ is an input word, $u_{\mathcal{A}}$ denotes the **relation induced by u in** $\mathcal{A}$, defined by

$$u_{\mathcal{A}} \quad := \quad \tau(u_1) \circ \cdots \circ \tau(u_{|u|}).$$

Note that $\epsilon_{\mathcal{A}}$ is the identity relation.

The automaton $\mathcal{A}$ can be visualized as a directed graph with vertices Q, and edges labeled by input symbols in Σ. Motivated by this point of view, we shall sometimes denote the relation $u_{\mathcal{A}}$ by $\xrightarrow{u}_{\mathcal{A}}$. Then $q \xrightarrow{u}_{\mathcal{A}} q'$ means that there is a directed u-labeled path from vertex q to vertex q'.

The language $L(\mathcal{A})$ recognized by $\mathcal{A}$ consists of those words $u \in \Sigma^*$ for which there exists a u-labeled path from some initial state to a final state, formally

$$L(\mathcal{A}) \quad = \quad \{u \in \Sigma^* \mid I \xrightarrow{u}_{\mathcal{A}} F\}.$$

When $\mathcal{A}$ is understood, we sometimes omit the subscript in $\xrightarrow{u}_{\mathcal{A}}$ and $u_{\mathcal{A}}$.

We call $\mathcal{A}$ a **deterministic finite automaton** (DFA) if it has at most one initial state, and each relation $\sigma_{\mathcal{A}}$ ($\sigma \in \Sigma$) is a *partial* function $Q \to Q$. A deterministic automaton is called **complete** if it has a unique initial state, and each of its input symbols induces a total function.

The automaton $\mathcal{A}$ is called a **reset automaton** if it has at most one initial state and each input symbol $\sigma \in \Sigma$ induces either the identity function or a partial constant function $Q \to Q$.

A state q of $\mathcal{A}$ is called **accessible** (respectively, **coaccessible**) if there exists some input word $u \in \Sigma^*$ with $I \xrightarrow{u}_{\mathcal{A}} \{q\}$ (respectively, $\{q\} \xrightarrow{u}_{\mathcal{A}} F$). Note that each initial state is accessible and each final state is coaccessible. A **biaccessible** state is one which is both accessible and coaccessible. Two states $q, q' \in Q$ are called **equivalent**, denoted $q \approx_{\mathcal{A}} q'$, if

$$\{q\} \xrightarrow{u}_{\mathcal{A}} F \quad \Longleftrightarrow \quad \{q'\} \xrightarrow{u}_{\mathcal{A}} F,$$

for all input words $u \in \Sigma^*$. Suppose that $\mathcal{A}$ is a DFA. Then $\mathcal{A}$ is called

minimal if all of its states are biaccessible, and it has no different equivalent states,

aperiodic if there exists an integer $k \geq 0$ such that $(u^k)_{\mathcal{A}} = (u^{k+1})_{\mathcal{A}}$, for all $u \in \Sigma^*$.

Observe that if $\mathcal{A}$ is a reset automaton then $(u^2)_{\mathcal{A}} = (u^3)_{\mathcal{A}}$, and if $\mathcal{A}$ is a complete reset automaton then $u_{\mathcal{A}} = (u^2)_{\mathcal{A}}$, for all words $u \in \Sigma^*$.

REMARK 2.1 *It is well known (see [4]) that a deterministic automaton $\mathcal{A} = (Q, \Sigma, \tau, I, F)$ is aperiodic if and only if it satisfies the implication*

$$q \xrightarrow{u^k}_{\mathcal{A}} q \implies q \xrightarrow{u}_{\mathcal{A}} q,$$

for all states $q \in Q$, input words $u \in \Sigma^+$ and integers $k \geq 2$.

Suppose that $n \geq 1$, and $\mathcal{A}_i = (Q_i, \Sigma, \tau_i, I_i, F_i)$ is an NFA, for each $i \in [n]$. Then the **product** of the A_i's is the NFA

$$\prod_{i\in[n]} \mathcal{A}_i = (\prod_{i\in[n]} Q_i, \Sigma, \tau, \prod_{i\in[n]} I_i, \prod_{i\in[n]} F_i),$$

where

$$\tau(\sigma) = \{((q_1, \dots, q_n), (r_1, \dots, r_n)) \mid \forall i \in [n]\ (q_i, r_i) \in \tau_i(\sigma)\},$$

for all $\sigma \in \Sigma$. It is easy to see that

$$L(\prod_{i\in[n]} \mathcal{A}_i) = \bigcap_{i\in[n]} L(\mathcal{A}_i).$$

2.5 Turing machines

A **deterministic Turing machine** (DTM) with a single one-way infinite tape is a system $\mathcal{M} = (Q, \Gamma, \Sigma, \delta, q_0, q_f)$, where

- Q is the finite set of states,
- Γ is the tape alphabet containing the special "blank" symbol $\flat$,
- $\Sigma \subseteq \Gamma$ is the input alphabet, $\flat \notin \Sigma$,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$ is the partial transition function,
- $q_0 \in Q$ is the initial state,
- $q_f \in Q$ is the final state.

We say the machine $\mathcal{M}$ is in the **configuration** (q, i, u) for a state $q \in Q$, integer $i \in \omega$ and infinite word $u \in \Gamma^\omega$ if in state q it scans the ith tape cell and the content of the tape is u. We define a binary relation $\vdash_{\mathcal{M}}$ on the set $Q \times \omega \times \Gamma^\omega$ of configurations by

$$(q, i, u) \vdash_{\mathcal{M}} (r, j, v) \iff \delta(q, u_i) = (r, v_i, j - i) \wedge \forall t \in \omega\ (t \neq i \implies v_t = u_t).$$

Note that $\vdash_{\mathcal{M}}$ is a partial function. The machine $\mathcal{M}$ **accepts** an input word $u \in \Sigma^*$ if

$$(q_0, 1, u\flat^\omega) \vdash^*_{\mathcal{M}} (q_f, 1, \flat^\omega),$$

otherwise $\mathcal{M}$ **rejects** u. The **language** $L(\mathcal{M}) \subseteq \Sigma^*$ **recognized by** $\mathcal{M}$ consists of those words $u \in \Sigma^*$ which are accepted by $\mathcal{M}$. Thus, for each word $u \in L(\mathcal{M})$, there exists a shortest sequence $(q_1, i_1, w_1), \ldots, (q_k, i_k, w_k)$ of configurations such that

$$\begin{aligned}(q_1, i_1, w_1) &= (q_0, 1, u\mathfrak{b}^\omega), \\ (q_k, i_k, w_k) &= (q_f, 1, \mathfrak{b}^\omega), \qquad \text{and} \\ (q_t, i_t, w_t) &\vdash_{\mathcal{M}} (q_{t+1}, i_{t+1}, w_{t+1}),\end{aligned}$$

for all $t \in [k-1]$. Then we define

$$\mathrm{SPACE}_{\mathcal{M}}(u) := \max_{t \in [k]} i_t.$$

Suppose that $S : \mathbf{N} \to \mathbf{N}$ is a (space constructible) function. The machine $\mathcal{M}$ is said to have **space complexity** S if $\mathrm{SPACE}_{\mathcal{M}}(u) \leq S(|u|)$, for all words $u \in L(\mathcal{M})$. The language class **PSPACE** consists of those languages which are recognized by some Turing machine $\mathcal{M}$ having space-complexity p, for some polynomial function $p : \mathbf{N} \to \mathbf{N}$.

We assume the reader is familiar with the concept of logspace-reducibility (see [2], for example).

Suppose L and L' are languages. In this paper, $L \leq_{log} L'$ stands for "L is logspace-reducible to L'". The language L is called **PSPACE-hard** with respect to logspace-reductions, written **PSPACE** $\leq_{log} L$, if every language in **PSPACE** is logspace-reducible to L. Lastly, L is called **PSPACE-complete** with respect to logspace-reductions if $L \in$ **PSPACE** and **PSPACE** $\leq_{log} L$.

3 Problems

We are interested in the computational complexity of the following decision problems:

1. The automata intersection problem (**AIP**):

   INPUT: A sequence $\mathcal{A}_1, \ldots, \mathcal{A}_n$ $(n \geq 2)$ of nondeterministic finite automata with a common input alphabet.

   QUESTION: Does $\bigcap_{i \in [n]} L(\mathcal{A}_i) \neq \emptyset$ hold?

2. A restricted version of the automata intersection problem (**AIP**$_R$):

   INPUT: A sequence $\mathcal{A}_1, \ldots, \mathcal{A}_n$ $(n \geq 2)$ of minimal reset automata with a common input alphabet.

   QUESTION: Does $\bigcap_{i \in [n]} L(\mathcal{A}_i) \neq \emptyset$ hold?

3. Automaton star-freeness (**ASF**):

INPUT: A nondeterministic finite automaton $\mathcal{A}$.

QUESTION: Does $\mathcal{A}$ recognize a star-free language?

4. A restricted version of automaton star-freeness ($\mathbf{ASF}_R$):

   INPUT: A minimal DFA $\mathcal{A}$ with input alphabet {0,1}.

   QUESTION: Does $\mathcal{A}$ recognize a star-free language?

5. Regular expression star-freeness (**RSF**):

   INPUT: A regular expression E.

   QUESTION: Does E denote a star-free language?

6. A restricted version of regular expression star-freeness ($\mathbf{RSF}_R$):

   INPUT: A regular expression E of star-height 2 over the alphabet $\{0,1\}$.

   QUESTION: Does E denote a star-free language?

Assuming some efficient encoding of automata and regular expressions (see [5]) with words over a fixed finite alphabet, all these problems can be considered as languages. We are going to prove

PROPOSITION 3.1 *The problems* **AIP**, $\mathbf{AIP}_R$, **ASF**, $\mathbf{ASF}_R$, **RSF** *and* $\mathbf{RSF}_R$ *are* **PSPACE**-*complete with respect to logspace reductions.*

4 Constructions

In this section we present the constructions of automata and regular expressions which are needed to show that the restricted problems $\mathbf{AIP}_R$, $\mathbf{ASF}_R$ and $\mathbf{RSF}_R$ are **PSPACE**-hard. The first construction shows how can one replace a deterministic Turing machine with a sequence of reset automata.

CONSTRUCTION 4.1 **Input:** A polynomial function $p : \mathbf{N} \to \mathbf{N}$, a DTM $\mathcal{M} = (Q, \Gamma, \Sigma, \delta, q_0, q_f)$ of space-complexity p, and an input word $u \in \Sigma^n$, $n \geq 0$.
Output: A sequence $\mathcal{S}, \mathcal{P}, \mathcal{A}_1, \ldots, \mathcal{A}_m$ of reset automata, where $m = \max\{p(n), 1\}$, and

$$u \in L(\mathcal{M}) \iff L(\mathcal{S}) \cap L(\mathcal{P}) \cap \bigcap_{i \in [m]} L(\mathcal{A}_i) \neq \emptyset. \qquad (4)$$

Description: Let

$$\begin{aligned} \mathcal{S} &= (Q, A, \tau_S, \{q_0\}, \{q_f\}) \\ \mathcal{P} &= ([m], A, \tau_{\mathcal{P}}, \{1\}, \{1\}), \end{aligned}$$

and for each $i \in [m]$,

$$\mathcal{A}_i = (\Gamma, A, \tau_i, \{(u\flat^\omega)_i\}, \{\flat\}),$$

where

$$A = \{\langle q,k,\gamma\rangle \mid q \in Q,\ k \in [m],\ \gamma \in \Gamma\}$$

and the transition functions $\tau_{\mathcal{S}}, \tau_{\mathcal{P}}, \tau_1, \ldots, \tau_m$ are defined as follows.

Suppose that $a = \langle q,k,\gamma\rangle$ is an element of A. If $\delta(q,\gamma)$ is undefined then

$$\tau_{\mathcal{S}}(a) = \tau_{\mathcal{P}}(a) = \tau_1(a) = \cdots = \tau_m(a) = \emptyset,$$

and if $\delta(q,\gamma)$ is defined, say $\delta(q,\gamma) = (r,\gamma',t)$, then

$$\begin{aligned} \tau_{\mathcal{S}}(a) &= \{(q,r)\} \\ \tau_{\mathcal{P}}(a) &= \begin{cases} \{(k,k+t)\} & \text{if } k+t \in [m], \\ \emptyset & \text{if } k+t \notin [m], \end{cases} \\ \tau_i(a) &= \begin{cases} \{(\gamma,\gamma')\} & \text{if } k = i \\ \{(\sigma,\sigma) \mid \sigma \in \Gamma\} & \text{if } k \neq i. \end{cases} \end{aligned}$$

Proof. The intuition is that the automata $\mathcal{S}, \mathcal{P}, \mathcal{A}_1, \ldots, \mathcal{A}_m$ together "simulate" the computation of $\mathcal{M}$ on the input word u, such that $\mathcal{S}$ knows the current state of $\mathcal{M}$, $\mathcal{P}$ knows the position of the read-write head, and each $\mathcal{A}_i$ ($i \in [m]$) knows the content of the ith tape-cell. An input symbol $\langle q,k,\gamma\rangle \in A$ corresponds to the statement "the current state of $\mathcal{M}$ is q, the position of the read-write head is k, and the content of the kth tape-cell is γ".

It is easy to see that each one of $\mathcal{S}, \mathcal{P}, \mathcal{A}_1, \ldots, \mathcal{A}_m$ is a reset automaton. (In fact they are even more restricted: for all input symbols $a \in A$, the relation induced by a in each one of the automata $\mathcal{S}, \mathcal{P}, \mathcal{A}_1, \ldots, \mathcal{A}_m$ is either empty, or a singleton, or the identity function.)

Consider the product automaton

$$\mathcal{A} = \mathcal{S} \times \mathcal{P} \times \prod_{i \in [m]} \mathcal{A}_i.$$

We know

$$L(\mathcal{A}) = L(\mathcal{S}) \cap L(\mathcal{P}) \cap \bigcap_{i \in [m]} L(\mathcal{A}_i).$$

Observe that for all $q,r \in Q$, $v,w \in \Gamma^m$, $j,k \in [m]$, and $a \in A$

$$\begin{aligned} &(q,j,v_1,\ldots,v_m) \xrightarrow{a}_{\mathcal{A}} (r,k,w_1,\ldots,w_m) \iff \\ &\qquad a = \langle q,j,v_j\rangle \ \wedge\ \delta(q,v_j) = (r,w_j,k-j) \ \wedge\ \forall t \in [m]\,(t \neq j \Rightarrow w_t = v_t), \end{aligned}$$

and thus

$$(q,j,vb^\omega) \vdash_{\mathcal{M}} (r,k,wb^\omega) \iff \exists a \in A\,(q,j,v_1,\ldots,v_m) \xrightarrow{a}_{\mathcal{A}} (r,k,w_1,\ldots,w_m).$$

It follows that

$$\begin{aligned} u \in L(\mathcal{M}) &\iff (q_0, 1, u\flat^\omega) \vdash^*_{\mathcal{M}} (q_f, 1, \flat^\omega) \\ &\iff \exists v \in A^* \ (q_0, 1, (u\flat^\omega)_1, \dots, (u\flat^\omega)_m) \xrightarrow{v}_{\mathcal{A}} (q_f, 1, \flat, \dots, \flat) \\ &\iff L(\mathcal{A}) \neq \emptyset. \end{aligned}$$

□

Although the automata $\mathcal{S}, \mathcal{P}, \mathcal{A}_1, \dots, \mathcal{A}_m$ constructed above have a very simple structure, they are not always minimal. In the next construction we show an easy way of modifying these automata so that they become minimal. Note that the standard procedure of automata minimization is not suitable for our purposes since it requires linear space.

CONSTRUCTION 4.2 **Input:** A sequence $\mathcal{A}_1, \dots, \mathcal{A}_n$ $(n \geq 2)$ of reset automata of the form $\mathcal{A}_i = (Q_i, \Sigma, \tau_i, \{s_i\}, \{f_i\})$.
Output: A sequence $\mathcal{B}_1, \dots, \mathcal{B}_n$ of minimal reset automata such that

$$\bigcap_{i \in [n]} L(\mathcal{A}_i) \quad = \quad \bigcap_{i \in [n]} L(\mathcal{B}_i). \tag{5}$$

Description: For each $i \in [n]$ let

$$\mathcal{B}_i \quad = \quad (Q_i, \Sigma \cup \Sigma', \tau'_i, \{s_i\}, \{f_i\}),$$

where

$$\begin{aligned} \Sigma' &= \{\langle q, j\rangle \mid q \in Q_j,\ j \in [n]\}, \\ \tau'_i(\sigma) &= \tau_i(\sigma), \\ \tau'_i(\langle q, j\rangle) &= \begin{cases} \{(p, q) \mid p \in Q_i,\ p \neq q\} & \text{if } j = i, \\ \emptyset & \text{if } j \neq i, \end{cases} \end{aligned}$$

for all $\sigma \in \Sigma$, $\langle q, j\rangle \in \Sigma'$.

Proof. For each $j \in [n]$ let Σ'_j denote the set $\{\langle q, j\rangle \mid q \in Q_j\}$. Consider the automaton $\mathcal{B}_i$ for some $i \in [n]$. It is obvious that $\mathcal{B}_i$ is a reset automaton. Since the elements of $\Sigma' \setminus \Sigma'_i$ induce the empty relation in $\mathcal{B}_i$,

$$L(\mathcal{B}_i) \quad \subseteq \quad (\Sigma \cup \Sigma'_i)^*.$$

Moreover, since each input symbol $\sigma \in \Sigma$ induces the same relation in $\mathcal{B}_i$ as in $\mathcal{A}_i$,

$$L(\mathcal{B}_i) \cap \Sigma^* \quad = \quad L(\mathcal{A}_i).$$

These two observations and $n \geq 2$ imply (5). Lastly, for all states $p, q \in Q_i$ we have

$$\begin{aligned} q \neq s_i &\implies s_i \xrightarrow{\langle q, i\rangle} q, \\ q \neq f_i &\implies q \xrightarrow{\langle f_i, i\rangle} f_i, \\ p \neq q \ \wedge\ q \neq f_i &\implies p\langle q, i\rangle\langle f_i, i\rangle = \{f_i\} \ \wedge\ q\langle q, i\rangle\langle f_i, i\rangle = \emptyset, \end{aligned}$$

showing $\mathcal{B}_i$ is minimal. □

The next construction shows that for each reset automaton $\mathcal{A}$ there exists a "short" regular expression denoting the complement of the language recognized by $\mathcal{A}$. This fact plays a key role in proving that the problem $\mathbf{RSF}_R$ is **PSPACE**-hard.

CONSTRUCTION 4.3 **Input:** A reset automaton

$$\mathcal{A} = (Q, \Sigma, \tau, I, F).$$

Output: A regular expression E over the alphabet Σ such that

$$L(E) \quad = \quad \overline{L(\mathcal{A})}. \tag{6}$$

Description: If $I = \emptyset$ then (6) holds for the regular expression $E = \Sigma^*$. From now on we assume that $\mathcal{A}$ has an initial state q_0. Let

$$\begin{aligned} X_q &= \{\sigma \in \Sigma \mid Q\sigma_{\mathcal{A}} = \{q\}\} \\ Y_q &= \{\sigma \in \Sigma \mid q\sigma_{\mathcal{A}} = \{q\}\} \\ Z_q &= \{\sigma \in \Sigma \mid q\sigma_{\mathcal{A}} = \emptyset\}, \end{aligned}$$

for all $q \in Q$. Using these subsets of Σ we define the regular expressions

$$E_q \quad = \quad \begin{cases} \Sigma^* X_q Y_q^* & \text{if } q \neq q_0 \\ \Sigma^* X_q Y_q^* \cup Y_q^* & \text{if } q = q_0, \end{cases}$$

for all $q \in Q$. Lastly, let

$$E \quad = \quad \left(\bigcup_{q \in Q \setminus F} E_q \right) \cup \left(\bigcup_{q \in Q} E_q Z_q \Sigma^* \right).$$

Proof. We claim

$$u \in L(E_q) \quad \Longrightarrow \quad q_0 u \subseteq \{q\} \tag{7}$$

and

$$q_0 u = \{q\} \quad \Longrightarrow \quad u \in L(E_q), \tag{8}$$

for all $q \in Q$, $u \in \Sigma^*$. Then (6) follows since the definition of E expresses the fact that an input word $u \in \Sigma^*$ is rejected by the automaton $\mathcal{A}$ either if $q_0 u = \{q\}$ for some non-final state q, or $q_0 u = \emptyset$.

As (7) is quite obvious, we only prove (8). Suppose that $q_0 u = \{q\}$ for some state $q \in Q$ and input word $u \in \Sigma^n$, $n \geq 0$. Then there exist some states $q_1, \ldots, q_{n-1}$ such that

$$q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} \cdots \xrightarrow{u_{n-1}} q_{n-1} \xrightarrow{u_n} q.$$

If $q_0 = q_1 = \cdots = q_{n-1} = q$ then $u \in Y_q^* \subseteq L(E_q)$. Otherwise let $k \in [n]$ be the largest index for which $q_{k-1} \neq q$, so that

$$q \neq q_{k-1} \xrightarrow{u_k} q \xrightarrow{u_{k+1}} \cdots \xrightarrow{u_{n-1}} q \xrightarrow{u_n} q.$$

Since $\mathcal{A}$ is deterministic it follows that $u_{k+1}, \ldots, u_n \in Y_q$. Moreover, since $q \neq q_{k-1} \xrightarrow{u_k} q$, the relation induced by u_k is not the identity function. Thus, u_k induces a partial constant function with range $\{q\}$, so that $u_k \in X_q$. We see $u \in \Sigma^* X_q Y_q^* \subseteq L(E_q)$. □

The next construction presents the main idea of reducing $\mathbf{AIP}_R$ to $\mathbf{ASF}_R$. The very same idea was used by Cho and Huynh in [3].

CONSTRUCTION 4.4 **Input:** A sequence $\mathcal{B}_1, \ldots, \mathcal{B}_n$ $(n \geq 2)$ of minimal reset automata of the form $\mathcal{B}_i = (Q_i, \Sigma, \tau_i, I_i, F_i)$.
Output: A minimal DFA $\mathcal{C}$ such that

$$\bigcap_{i \in [n]} L(\mathcal{B}_i) = \emptyset \iff L(\mathcal{C}) \text{ is star-free.} \tag{9}$$

Description: If $I_i = \emptyset$ for some $i \in [n]$ then let $\mathcal{C}$ be the minimal DFA with input alphabet $\{0\}$ recognizing the star-free language $\emptyset$. From now on we assume that each automaton $\mathcal{B}_i$ has a unique initial state s_i. Thus $L(\mathcal{B}_i) \neq \emptyset$, for all $i \in [n]$. Let p be the least prime number with $p \geq n$. It is well known (see [6]) that $p < 2n$. For integers $i \in \{n+1, n+2, \ldots, p\}$ let $\mathcal{B}_i = (Q_i, \Sigma, \tau_i, \{s_i\}, F_i)$ be a minimal DFA recognizing the language Σ^*. For the sake of simplicity assume that the sets Q_i $(i \in [p])$ are pairwise disjoint, and that $\# \notin \Sigma$ is a new input symbol. Let $\nu : \mathbf{N} \to [p]$ be the function mapping each integer i to $((i-1) \bmod p) + 1$. Then we define

$$\mathcal{C} := (\bigcup_{i \in [p]} Q_i, \Sigma \cup \{\#\}, \tau, \{s_1\}, \{s_1\}),$$

where

$$\begin{aligned} \tau(\#) &= \bigcup_{i \in [p]} F_i \times \{s_{\nu(i+1)}\} \\ \tau(\sigma) &= \bigcup_{i \in [p]} \tau_i(\sigma), \end{aligned}$$

for all input symbols $\sigma \in \Sigma$. See Figure 1.

Proof. Clearly, $\mathcal{C}$ is a DFA with

$$L(\mathcal{C}) = (L(\mathcal{B}_1)\#L(\mathcal{B}_2)\# \cdots L(\mathcal{B}_n)\#(\Sigma^*\#)^{p-n})^*.$$

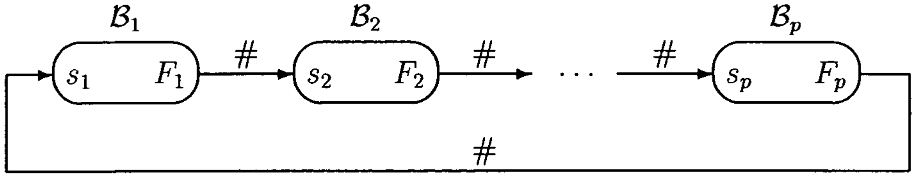

Figure 1: The automaton $\mathcal{C}$

By Schützenberger's theorem, (9) is equivalent to the condition

$$\bigcap_{i\in[n]} L(\mathcal{B}_i) \neq \emptyset \quad \Longleftrightarrow \quad \mathcal{C} \text{ is not aperiodic.} \tag{10}$$

The "$\Longrightarrow$" part of (10) is obvious: if $u \in \Sigma^*$ is a common element of the languages $L(\mathcal{B}_1), \dots, L(\mathcal{B}_n)$ then $s_1 \xrightarrow{(u\#)^p}_{\mathcal{C}} s_1$ and $s_1 \xrightarrow{u\#}_{\mathcal{C}} s_2 \neq s_1$, so that $\mathcal{C}$ is not aperiodic by Remark 2.1. Before we prove the "$\Longleftarrow$" part of (10) observe that if the letter # appears l times in an input word $u \in (\Sigma \cup \{\#\})^*$, and $q \in Q_i$ is a state such that $q(u\#)_{\mathcal{C}} \neq \emptyset$ then $q(u\#)_{\mathcal{C}} = \{s_{\nu(i+l+1)}\}$. Moreover, if $s_j(v\#)_{\mathcal{C}} \neq \emptyset$ for some integer $j \in [p]$ and word $v \in \Sigma^*$ then $v \in L(\mathcal{B}_j)$. Now suppose that $\mathcal{C}$ is not aperiodic, i.e.

$$q \quad \xrightarrow{u^k}_{\mathcal{C}} \quad q, \tag{11}$$

and

$$q \quad \xrightarrow{u}_{\mathcal{C}} \quad q', \tag{12}$$

for some *different* states $q \in Q_i$, $q' \in Q_{i'}$, $i, i' \in [p]$, input word $u \in (\Sigma \cup \{\#\})^+$ and integer $k \geq 2$. Note that by (11) we have $q(u^t)_{\mathcal{C}} \neq \emptyset$, for all $t \geq 0$. Let l be the number of #'s in u, so that u can be written as

$$u^{(0)}\#u^{(1)}\#\dots\#u^{(l)},$$

where $u^{(0)}, \dots, u^{(l)}$ are words in Σ^*. If l were 0 then we would have $i' = i$, $q \xrightarrow{u^k}_{\mathcal{B}_i} q$, and $q \xrightarrow{u}_{\mathcal{B}_i} q' \neq q$, contradicting the fact that $\mathcal{B}_i$ is aperiodic. Thus, $l > 0$.

Let v denote the word $u^{(0)}\# \cdots \#u^{(l-1)}$, so that $u = v\#u^{(l)}$ and

$$q \quad \xrightarrow{v\#}_{\mathcal{C}} \quad s_j,$$

where $j = \nu(i + l)$. By (11) we have

$$q \xrightarrow{u^{k-1}v\#}_{\mathcal{C}} s_i \xrightarrow{u^{(l)}}_{\mathcal{C}} q.$$

If p were a divisor of l then it would follow that $j = i$ and $q \xrightarrow{v\#}_{\mathcal{C}} s_i \xrightarrow{u^{(l)}}_{\mathcal{C}} q$, contradicting (12). Thus p is not a divisor of l.

Let j be an arbitrary element of $[n]$. As p is a prime not dividing l, there exists some integer $t \geq 0$ such that $\nu(i + lt) = j$. For this t we have

$$q \xrightarrow{u^{t-1}v\#}_{\mathcal{C}} s_j.$$

Moreover, since $u^{t-1}v\#u^{(l)}u^{(0)}\#$ is a prefix of u^{t+1} and $q(u^{t+1})_{\mathcal{C}} \neq \emptyset$, it follows that

$$s_j(u^{(l)}u^{(0)}\#)_{\mathcal{C}} = q(u^{t-1}v\#u^{(l)}u^{(0)}\#)_{\mathcal{C}} \neq \emptyset,$$

showing $u^{(l)}u^{(0)} \in L(\mathcal{B}_j)$. Since $j \in [n]$ was arbitrary,

$$u^{(l)}u^{(0)} \in \bigcap_{j\in[n]} L(\mathcal{B}_j).$$

In order to prove $\mathcal{C}$ is minimal suppose that $q \in Q_j$ and $r \in Q_k$ are two different states of the automaton $\mathcal{C}$. For each $i \in [p]$ choose an arbitrary word $v^{(i)} \in L(\mathcal{B}_i)$. Since q is a biaccessible state of $\mathcal{B}_j$, there exist words $v, w \in \Sigma^*$ with

$$s_j \xrightarrow{v}_{\mathcal{B}_j} q \xrightarrow{w}_{\mathcal{B}_j} F_j.$$

Then

$$s_1 \xrightarrow{v^{(1)}\#\cdots\#v^{(j-1)}\#v}_{\mathcal{C}} q \xrightarrow{w\#v^{(j+1)}\#\cdots\#v^{(p)}\#}_{\mathcal{C}} s_1,$$

showing q is a biaccessible state of $\mathcal{C}$. If $j \neq k$, say $j < k$, then

$$\begin{aligned} q(w\#v^{(j+1)}\#\cdots\#v^{(p)}\#)_{\mathcal{C}} &= \{s_1\} \quad \text{and} \\ r(w\#v^{(j+1)}\#\cdots\#v^{(p)}\#)_{\mathcal{C}} &\subseteq \{s_{k-j+1}\}. \end{aligned}$$

Lastly, suppose that $j = k$. Since $\mathcal{B}_j$ is minimal, there exists some word $x \in \Sigma^*$ such that exactly one of the sets $qx_{\mathcal{B}_j} \cap F_j$ and $rx_{\mathcal{B}_j} \cap F_j$ is empty, say $rx_{\mathcal{B}_j} \cap F_j = \emptyset$. Then $q(x\#)_{\mathcal{C}} = \{s_{\nu(j+1)}\}$, $r(x\#)_{\mathcal{C}} = \emptyset$ and we have

$$\begin{aligned} q(x\#v^{(j+1)}\#\cdots\#v^{(p)}\#)_{\mathcal{C}} &= \{s_1\} \quad \text{and} \\ r(x\#v^{(j+1)}\#\cdots\#v^{(p)}\#)_{\mathcal{C}} &= \emptyset. \end{aligned}$$

□

The last construction gives the second part of the reduction $\mathbf{AIP}_R \leq_{log} \mathbf{ASF}_R$.

CONSTRUCTION 4.5 **Input:** A minimal DFA $\mathcal{C} = (Q, \Sigma, \tau, I, F)$.
Output: A minimal DFA $\mathcal{C}'$ with input alphabet $\{0, 1\}$ such that

$$L(\mathcal{C}) \text{ is star-free} \iff L(\mathcal{C}') \text{ is star-free.} \qquad (13)$$

Description: Let $\psi : \Sigma^* \to \{0,1\}^*$ be an injective homomorphism satisfying the conditions of Lemma 2.1. In particular, the image $\psi(\sigma)$ of each symbol $\sigma \in \Sigma$ is a word in $\{0,1\}^{2l}$, where $l = \lceil \log_2(|\Sigma| + 1) \rceil$. For each state $q \in Q$ let

$$S_q \quad := \quad \{\psi(\sigma)q' \mid \sigma \in \Sigma,\ q' \in Q,\ q \xrightarrow{\sigma}_C q'\},$$

so that S_q is a set of words over the alphabet $\{0,1\} \cup Q$, more precisely, $S_q \subseteq \{0,1\}^{2l}Q$. When S is a set of words and u is a word over the same alphabet, $u\backslash S$ denotes the set $\{v \mid uv \in S\}$. For each integer $j \in [2l-1]$ let

$$Q'_j \quad := \quad \{u\backslash S_q \mid q \in Q,\ u \in \{0,1\}^j\} \setminus \{\emptyset\}.$$

Thus each element of Q'_j is a nonempty subset of $\{0,1\}^{2l-j}Q$. Now let

$$C' \quad := \quad (Q \cup Q', \{0,1\}, \tau', I, F),$$

where

$$Q' \quad = \quad \bigcup_{j \in [2l-1]} Q'_j,$$

and τ' is defined such that

$$qx_{C'} \quad = \quad \begin{cases} \{x\backslash S_q\} & \text{if } x\backslash S_q \neq \emptyset, \\ \emptyset & \text{otherwise,} \end{cases}$$

$$Sx_{C'} \quad = \quad \begin{cases} \{q'\} & \text{if } x\backslash S = \{q'\}, \text{ for some } q' \in Q, \\ \emptyset & \text{if } x\backslash S = \emptyset, \\ \{x\backslash S\} & \text{otherwise,} \end{cases}$$

for all $q \in Q$, $S \in Q'$, $x \in \{0,1\}$.

Proof. Let us denote Q by Q'_0. It is easy to see that C' is a DFA satisfying

$$q \xrightarrow{u}_{C'} q' \iff \exists v \in \Sigma^*\ u = \psi(v) \wedge q \xrightarrow{v}_C q', \tag{14}$$

and

$$Q'_i \xrightarrow{u}_{C'} Q'_j \implies |u| \equiv j - i \pmod{2l}, \tag{15}$$

for all $q, q' \in Q$, $u \in \{0,1\}^*$, $0 \leq i,j < 2l$. It follows in particular that $L(C') = \psi(L(C))$, so that (13) holds by Lemma 2.1.

In order to prove C' is minimal suppose that $s \in Q'_i$ and $s' \in Q'_j$ $(0 \leq i,j < 2l)$ are two different states of C'. It is clear from the description of C' that there exist words $v \in \{0,1\}^i$, $v' \in \{0,1\}^{2l-i}$, and states $q, q' \in Q$ such that $q \xrightarrow{v}_{C'} s \xrightarrow{v'}_{C'} q'$. Since C is minimal, there exist words $u, u' \in \Sigma^*$ with $I \xrightarrow{u}_C q$ and $q' \xrightarrow{u'}_C F$. By (14) we have

$$I \xrightarrow{\psi(u)}_{C'} q \xrightarrow{v}_{C'} s \xrightarrow{v'}_{C'} q' \xrightarrow{\psi(u')}_{C'} F,$$

showing s is a biaccessible state of $\mathcal{C}'$. If $i \neq j$ then s and s' are not equivalent by (15), so suppose $i = j$. If $i = j = 0$ then s and s' are two different elements of Q, and since $\mathcal{C}$ is minimal there exists a word $w \in \Sigma^*$ such that exactly one of the two sets $sw_{\mathcal{C}} \cap F = s\psi(w)_{\mathcal{C}'} \cap F$ and $s'w_{\mathcal{C}} \cap F = s'\psi(w)_{\mathcal{C}'} \cap F$ is empty. Lastly suppose that $i = j \in [2l-1]$. Then s and s' are two different subsets of the set $\{0,1\}^{2l-i}Q$, say $s \not\subseteq s'$. Let uq be an arbitrary element in s which is not in s', where $u \in \{0,1\}^{2l-i}$ and $q \in Q$. There are two possibilities: either $s'u_{\mathcal{C}'} = \emptyset$ or $s'u_{\mathcal{C}'} = \{q'\}$ for some state $q' \in Q$, $q' \neq q$. In the first case we have $su\psi(v)_{\mathcal{C}'} \cap F \neq \emptyset$ and $s'u\psi(v)_{\mathcal{C}'} \cap F = \emptyset$, where $v \in \Sigma^*$ is an arbitrary word with $q \xrightarrow{v}_{\mathcal{C}} F$. (Such a v exists since q is a coaccessible state of $\mathcal{C}$.) The second case can be handled similarly to the case $i = j = 0$. □

5 Results

THEOREM 5.1 *The problems* **AIP** *and* **AIP**$_R$ *are* **PSPACE**-*complete with respect to logspace reductions.*

Proof. We show

$$\textbf{PSPACE} \leq_{log} \textbf{AIP}_R \leq_{log} \textbf{AIP} \in \textbf{PSPACE}.$$

Suppose that $L \subseteq \Sigma^*$ is a language in **PSPACE**. Then there exists a polynomial function $p : \mathbf{N} \to \mathbf{N}$ and a deterministic Turing machine $\mathcal{M}$ of space-complexity p such that $L(\mathcal{M}) = L$. Applying Construction 4.1 followed by Construction 4.2 to $\mathcal{M}$ and an input word $u \in \Sigma^*$, we obtain a list $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of minimal reset automata such that

$$u \in L \iff \bigcap_{i\in[n]} L(\mathcal{A}_i) \neq \emptyset.$$

Since both constructions can be carried out by a logspace-bounded Turing machine, **PSPACE** $\leq_{log}$ **AIP**$_R$. The claim **AIP**$_R$ $\leq_{log}$ **AIP** is trivial. In order to prove **AIP** $\in$ **PSPACE** suppose that $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are NFA's with a common input alphabet Σ, say $\mathcal{A}_i = (Q_i, \Sigma, \tau_i, I_i, F_i)$. The following nondeterministic PASCAL-style program accepts the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ if and only if $\bigcap_{i\in[n]} L(\mathcal{A}_i) \neq \emptyset$:

```
function Solve_AIP($\mathcal{A}_1, \ldots \mathcal{A}_n$ : NFA):boolean;
var
    $S_1, \ldots, S_n$ : set of state;
    $\sigma$ : input symbol;
begin
    $S_1 := I_1$;
      ⋮
    $S_n := I_n$;
    while $S_1 \cap F_1 = \emptyset$ or $\cdots$ or $S_n \cap F_n = \emptyset$ do
    begin
        guess $\sigma \in \Sigma$;
        $S_1 := S_1\sigma_{\mathcal{A}_1}$;
            $\vdots$
        $S_n := S_n\sigma_{\mathcal{A}_n}$;
    end;
    Solve_AIP:=true;
end;
```

The space complexity of the program is linear. It follows by Savitch's theorem that **AIP** $\in$ **PSPACE**. □

THEOREM 5.2 *The problems* **ASF** *and* $\mathbf{ASF}_R$ *are* **PSPACE**-*complete with respect to logspace reductions.*

Proof. We show

$$\overline{\mathbf{AIP}_R} \leq_{log} \mathbf{ASF}_R \leq_{log} \mathbf{ASF} \in \mathbf{PSPACE}.$$

Suppose that $\mathcal{B}_1, \ldots, \mathcal{B}_n$ $(n \geq 2)$ are minimal reset automata with a common input alphabet. Applying Construction 4.4 followed by Construction 4.5 to $\mathcal{B}_1, \ldots, \mathcal{B}_n$, we obtain a minimal DFA $\mathcal{C}'$ with input alphabet $\{0, 1\}$ such that

$$\bigcap_{i \in [n]} L(\mathcal{B}_i) = \emptyset \quad \Longleftrightarrow \quad L(\mathcal{C}') \text{ is star-free.}$$

Since both constructions can be carried out by a logspace-bounded Turing machine, $\overline{\mathbf{AIP}_R} \leq_{log} \mathbf{ASF}_R$. The claim $\mathbf{ASF}_R \leq_{log} \mathbf{ASF}$ is trivial. In order to prove **ASF** $\in$ **PSPACE** suppose that $\mathcal{A} = (Q, \Sigma, \tau, I, F)$ is an NFA. By Schützenberger's theorem, $L(\mathcal{A})$ is star-free if and only if the minimal DFA recognizing $L(\mathcal{A})$ is aperiodic. Recall that the power automaton of $\mathcal{A}$ is the deterministic automaton

$$\mathrm{P}(\mathcal{A}) \;=\; (\mathrm{P}(Q), \Sigma, \tau', \{I\}, F'),$$

where

$$\begin{aligned} F' &= \{S \in \mathrm{P}(Q) \mid S \cap F \neq \emptyset\}, \\ \tau'(\sigma) &= \{(S, S\sigma_{\mathcal{A}}) \mid S \in \mathrm{P}(Q)\}, \end{aligned}$$

for all $\sigma \in \Sigma$. The minimal DFA recognizing $L(\mathcal{A})$ is obtained from $\mathrm{P}(\mathcal{A})$ by deleting those states which are not biaccessible, and then identifying the equivalent states. It follows that $L(\mathcal{A})$ is star-free if and only if there exists some input word $u \in \Sigma^*$, *accessible* state S of $\mathrm{P}(\mathcal{A})$ and integer $k \geq 2$ such that $S \approx_{\mathrm{P}(\mathcal{A})} S(u^k)_{\mathcal{A}} = S(u_{\mathcal{A}})^k$ and $S \not\approx_{\mathrm{P}(\mathcal{A})} Su_{\mathcal{A}}$. The following nondeterministic procedure decides if $S \not\approx_{\mathrm{P}(\mathcal{A})} S'$ holds for two states S, S' of $\mathrm{P}(\mathcal{A})$:

```
function Not_Equiv($S, S'$ : set of state):boolean;
var
    $\sigma$ : input symbol;
begin
    while ($S \cap F = \emptyset$ and $S' \cap F = \emptyset$) or
          ($S \cap F \neq \emptyset$ and $S' \cap F \neq \emptyset$) do
    begin
        guess $\sigma \in \Sigma$;
        $S := S\sigma_{\mathcal{A}}$;
        $S' := S'\sigma_{\mathcal{A}}$;
    end;
    Not_Equiv:=true;
end;
```

By Savitch's theorem we obtain a deterministic polynomial-space program Equiv which decides if two states of $\mathrm{P}(\mathcal{A})$ are equivalent. The following nondeterministic program uses Equiv as a subroutine to decide if $L(\mathcal{A})$ is not star-free:

```
function Not_ASF($\mathcal{A}$ : NFA):boolean;
var
    $\sigma$ : input symbol;
    $S, S'$ : set of state;
    $\rho$ : relation;
    $halt$ : boolean;
begin
    $S := I$;
    repeat
        guess $\sigma \in \Sigma$;
        $S := S\sigma_{\mathcal{A}}$;
        guess $halt$;
    until $halt$;
    $\rho := \epsilon_{\mathcal{A}}$;
    repeat
        guess $\sigma \in \Sigma$;
        $\rho := \rho \circ \sigma_{\mathcal{A}}$;
        guess $halt$;
    until $halt$;
    $S' := S\rho$;
    if Equiv($S, S'$) then
        Not_ASF:=false
    else begin
        repeat
            $S' := S'\rho$;
        until Equiv($S, S'$);
        Not_ASF:=true;
    end;
```

end;

By Savitch's theorem and the fact that the language class **PSPACE** is closed under complementation it follows that **ASF** $\in$ **PSPACE**. □

THEOREM 5.3 *The problems* **RSF** *and* $\mathbf{RSF}_R$ *are* **PSPACE**-*complete with respect to logspace reductions.*

Proof. We show

$$\overline{\mathbf{AIP}_R} \leq_{log} \mathbf{RSF}_R \leq_{log} \mathbf{RSF} \leq_{log} \mathbf{ASF}.$$

The claim $\mathbf{RSF}_R \leq_{log}$ **RSF** is trivial, and it is also easy to see that **RSF** $\leq_{log}$ **ASF**: given a regular expression E, a logspace-bounded Turing machine can construct a *nondeterministic* automaton $\mathcal{A}$ such that $L(E) = L(\mathcal{A})$.

Suppose that $\mathcal{B}_1, \ldots, \mathcal{B}_n$ ($n \geq 2$) are minimal reset automata with a common input alphabet Σ. Let $\mathcal{C}$ be the result of Construction 4.4 applied to the automata $\mathcal{B}_1, \ldots, \mathcal{B}_n$. Then $\mathcal{C}$ is a minimal DFA with input alphabet $\Sigma \cup \{\#\}$ such that

$$\bigcap_{i \in [n]} L(\mathcal{B}_i) = \emptyset \iff L(\mathcal{C}) \text{ is star-free.}$$

Applying Construction 4.3 to each one of the automata $\mathcal{B}_1, \ldots, \mathcal{B}_n$ we get regular expressions $E_1, \ldots, E_n$ such that

$$L(E_i) = \overline{L(\mathcal{B}_i)},$$

for all $i \in [n]$. Recall that

$$L(\mathcal{C}) = \left(L(\mathcal{B}_1)\#L(\mathcal{B}_2)\# \cdots L(\mathcal{B}_n)\#(\Sigma^*\#)^{p-n}\right)^*,$$

where p is the least prime number with $p \geq n$. It follows that a word $v = v^{(0)}\#v^{(1)}\# \cdots v^{(k-1)}\#v^{(k)}$ ($k \geq 0$, $v^{(0)}, \ldots, v^{(k)} \in \Sigma^*$) belongs to $L(\mathcal{C})$ if and only if $v^{(k)} = \epsilon$, k is a multiple of p, and $v^{(i)} \in L(\mathcal{B}_{(i \bmod p)+1})$, for all $i < k$ with $i \bmod p < n$. The languages denoted by the regular expressions

$$\begin{aligned}
F_1 &= (\Sigma \cup \#)^*\Sigma \\
F_2 &= ((\Sigma^*\#)^p)^* \left(\bigcup_{i \in [p-1]} (\Sigma^*\#)^i \right) \Sigma^* \\
F_3 &= ((\Sigma^*\#)^p)^* \left(\bigcup_{i \in [n]} (\Sigma^*\#)^{i-1} E_i \# \right) (\Sigma \cup \#)^*
\end{aligned}$$

consist of those words $v = v^{(0)}\#v^{(1)}\# \cdots v^{(k-1)}\#v^{(k)}$ for which

1. $v^{(k)} \neq \epsilon$,
2. k is not a multiple of p,
3. $v^{(i)} \notin L(\mathcal{B}_{(i \bmod p)+1})$ for some $i < k$ with $i \bmod p < n$,

respectively. Thus, the regular expression $E := F_1 \cup F_2 \cup F_3$ denotes the complement of the language $L(\mathcal{C})$. Let $\psi : (\Sigma \cup \{\#\})^* \to \{0,1\}^*$ be a homomorphism satisfying the conditions of Lemma 2.1. Let E' be the regular expression obtained from E by replacing each occurence of every letter $x \in \Sigma \cup \{\#\}$ by the word $\psi(x) \in \{0,1\}^*$. Then E' is a regular expression over the alphabet $\{0,1\}$ having star-height 2. Moreover, $L(E') = \psi(L(E)) = \psi(\overline{L(\mathcal{C})})$, so that

$$L(E') \text{ is star-free} \iff L(\mathcal{C}) \text{ is star-free} \iff \bigcap_{i \in [n]} L(\mathcal{B}_i) = \emptyset.$$

The simple structure of E' assures that it can be constructed by a logspace-bounded Turing machine. □

6 Open problems

The above results suggest that the following questions may be interesting.

1. What is the complexity of deciding whether $\bigcap_{i \in [n]} L(\mathcal{A}_i) \neq \emptyset$, for minimal complete reset automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$?
2. What is the complexity of deciding whether a regular expression of star-height 1 denotes a star-free language?

We conjecture that the answer for the first question is "**NP**-complete".

The second question seems to be harder. However, it is our conjecture that restricting the problem **RSF** to regular expressions of star-height 1 substantially decreases its computational complexity.

7 Acknowledgement

I would like to thank Zoltán Ésik for encouragement and for many useful comments and suggestions. I also thank Stephen L. Bloom and an anonymous referee whose suggestions have been incorporated.

References

[1] L. Bernátsky and Z. Ésik. Semantics of Flowchart Programs and the Free Conway Theories. Submitted for publication to RAIRO Theoretical Informatics and Applications.

[2] D. P. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity.* Prentice Hall, 1994.

[3] Sang Cho and Dung T. Huynh. Finite-automaton aperiodicity is PSPACE-complete. *Theoretical Computer Science*, 88:99–116, 1991.

[4] Samuel Eilenberg. *Automata, Languages and Machines.* Academic Press, New York and London, 1974.

[5] Michael R. Garey and David S. Johnson. *Computers and intractability, A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[6] G.H. Hardy and E.M. Wright. *An Introduction to the Theory of Numbers.* Oxford University Press, London, 3rd edition, 1954.

[7] A. Salomaa. *Theory of Automata.* Pergamon Press, 1969.

[8] M. P. Schützenberger. On Finite Monoids Having Only Trivial Subgroups. *Information and Control*, 8:190–194, 1965.

[9] Jacques Stern. Complexity of Some Problems from the Theory of Automata. *Information and Control*, 66:163–176, 1985.

[10] H. Straubing. *Finite Automata, Formal Languages and Circuit Complexity.* Birkhäuser, 1994.

A Appendix

Proof of Lemma 2.1. First of all note that $n \leq 2^l - 1$, so that l bits are sufficient to represent the number n in binary. Let $\psi : \Sigma^* \to \{0,1\}^*$ be the homomorphism mapping each letter $\sigma_i \in \Sigma$ ($i \in [n]$) to the $2l$-bit binary representation of i. Then ψ is injective and satisfies (1).

Proof of (2). Suppose that (2) is not true. Then there exist some words $u, v, w \in \{0,1\}^*$ such that $uv^{2l}w \in \psi(\Sigma^*)$, but $2l$ is not a divisor of $|v|$. Let us denote $|v|$ by m. Then $m > 0$ and $\gcd(2l, m) < 2l$. Since none of the integers $l+1, l+2, \ldots, 2l-1$ is a divisor of $2l$,

$$\gcd(2l, m) \quad \leq \quad l. \tag{16}$$

Moreover, since no word in $\psi(\Sigma^*)$ may contain 0^{2l} as a subword, the letter 1 occurs in the word v^{2l}, and thus in v. Let $j \in [m]$ be an integer such that the jth letter of v is 1. If $i \in [2lm]$ is an integer satisfying

$$i \quad \equiv \quad j \pmod m \tag{17}$$

then the ith letter of v^{2l} is 1. By (1) it follows that if $1 \leq i \leq |uv^{2l}w|$ is an integer such that $(i-1) \bmod 2l < l$, then the ith letter of $uv^{2l}w$ is 0. In other words, if $i \in [2lm]$ is an integer satisfying

$$i \equiv t - |u| \pmod{2l} \tag{18}$$

for some $t \in [l]$, then the ith letter of v^{2l} is 0. The diophantic system (17,18) is solvable in the variable i if and only if

$$t - |u| \equiv j \pmod{\gcd(2l, m)}, \tag{19}$$

and in this case every solution can be written in the form

$$i = i_0 + h \cdot \operatorname{lcm}(2l, m),$$

where i_0 is a fixed solution and h is an integer. Let t be the unique element of $[\gcd(2l,m)]$ satisfying (19). Then $t \in [l]$, by (16). For this t there exists a unique integer $i \in [\operatorname{lcm}(2l,m)] \subseteq [2lm]$ such that both (17) and (18) hold. But then we have the contradiction that the ith letter of v^{2l} is equal to both 0 and 1. This contradiction was caused by the assumption that (2) fails.

Proof of (3). Suppose that $L \subseteq \Sigma^*$ is a language and $\psi(L) \subseteq \{0,1\}^*$ is star-free. Then L is regular and there exists an integer $k \geq 0$ such that for all words $u, v, w \in \Sigma^*$,

$$\begin{aligned} uv^k w \in L &\iff \psi(u)\psi(v)^k\psi(w) \in \psi(L) \\ &\iff \psi(u)\psi(v)^{k+1}\psi(w) \in \psi(L) \\ &\iff uv^{k+1}w \in L, \end{aligned}$$

showing L is star-free. Thus, for this direction no special property of the homomorphism ψ is needed other than its injectivity.

For the converse direction, suppose that $L \subseteq \Sigma^*$ is a star-free language. Then there exists an integer $k \geq 0$ such that

$$xy^k z \in L \iff xy^{k+1}z \in L, \tag{20}$$

for all words $x, y, z \in \Sigma^*$. Let m be the maximum of $2l$ and $k+1$. Suppose that $uv^m w \in \psi(L)$, for some $u, v, w \in \{0,1\}^*$. We want to show that $uv^{m+1}w \in \psi(L)$. This is obvius if $|v| = 0$, so suppose that $|v| > 0$. By (2) it follows that $|v|$ is a multiple of $2l$, so that $|v| \geq 2l$. Let α be the shortest prefix of v such that the length of the word $u\alpha$ is a multiple of $2l$. Then v can be written as $\alpha\beta$, for some word $\beta \in \{0,1\}^*$. Since $uv^m w = u\alpha(\beta\alpha)^{m-1}\beta w \in \psi(L)$ and the length of the words $u\alpha$, $\beta\alpha$ and βw are multiples of $2l$, there exist words $x, y, z \in \Sigma^*$ such that $\psi(x) = u\alpha$, $\psi(y) = \beta\alpha$, $\psi(z) = \beta w$ and $xy^{m-1}z \in L$. Since $m - 1 \geq k$, it follows by (20) that $xy^m z \in L$. Thus,

$$\psi(xy^m z) = u\alpha(\beta\alpha)^m\beta w = uv^{m+1}w \in \psi(L).$$

The implication $uv^{m+1}w \in \psi(L) \Longrightarrow uv^m w \in \psi(L)$ is proved in a similar way. □

Received January, 1996