Efficient Construction of Semilinear Representations of Languages Accepted by Unary Nondeterministic Finite Automata

Zdeněk Sawa[*]
Center for Applied Cybernetics, Department of Computer Science
Technical University of Ostrava
17. listopadu 15, Ostrava-Poruba, 708 33, Czech Republic
zdenek.sawa@vsb.cz

Abstract. In languages over a unary alphabet, i.e., an alphabet with only one letter, words can be identified with their lengths. It is well known that each regular language over a unary alphabet can be represented as the union of a finite number of arithmetic progressions. Given a nondeterministic finite automaton (NFA) working over a unary alphabet (a unary NFA), the arithmetic progressions representing the language accepted by the automaton can be easily computed by the determinization of the given NFA. However, the number of the arithmetic progressions computed in this way can be exponential with respect to the size of the original automaton. Chrobak (1986) has shown that in fact $O(n^2)$ arithmetic progressions are sufficient for the representation of the language accepted by a unary NFA with n states, and Martinez (2002) has shown how these progressions can be computed in polynomial time. Recently, To (2009) has pointed out that Chrobak's construction and Martinez's algorithm, which is based on it, contain a subtle error and has shown how to correct this error. Geffert (2007) presented an alternative proof of Chrobak's result, also improving some of the bounds. In this paper, a new simpler and more efficient algorithm for the same problem is presented, using some ideas from Geffert (2007). The time complexity of the presented algorithm is $O(n^2(n+m))$ and its space complexity is $O(n+m)$, where n is the number of states and m the number of transitions of a given unary NFA.

1. Introduction

It is well known that Parikh images of regular (and even context-free) languages are semilinear sets [9, 6]. In languages over a *unary* alphabet, i.e., an alphabet with only one letter, words can be identified with

[*] Supported by the Czech Ministry of Education, Grant No. 1M0567, and the Czech Science Foundation – GACR, Grant No. P202/11/0340.

their lengths (i.e., a^n can be identified with n), so the Parikh image of a unary language is just the set of lengths of words of the language, and it can be identified with the language itself. It can be easily shown that each regular unary language can be represented as the union of a finite number of arithmetic progressions of the form $\{c+di \mid i \in \mathbb{N}\}$ where c and d are constants specifying the offset and the period of a progression.

A unary nondeterministic finite automaton (a unary NFA) is an NFA with a one-letter alphabet. Given a unary NFA $\mathcal{A}$, a set of arithmetic progressions representing the language accepted by $\mathcal{A}$ can be computed by determinization of $\mathcal{A}$; however, this straightforward approach can produce an exponential number of progressions, even if the resulting automaton is minimized. Chrobak [1, 2] has shown that this exponential blowup is avoidable and that a language accepted by a unary NFA with n states can be represented as the union of $O(n^2)$ progressions of the form $\{c + di \mid i \in \mathbb{N}\}$ where $c < p(n)$ for some $p(n) \in O(n^2)$ and $0 \leq d \leq n$. The computational complexity of the construction of these progressions was not analyzed in [1], but it can be seen that a naive straightforward implementation would require exponential time. Later, Martinez [7, 8] has shown how the construction described in [1] can be realized in polynomial time. The exact complexity of Martinez's algorithm is $O(kn^4)$ where n is the number of states of the automaton and k the number of strongly connected components of its graph. The result was recently used for example in [5, 4] to obtain more efficient algorithms for some problems in automata theory and the verification of one-counter processes.

Geffert presented in [3] a new alternative proof of Chrobak's result, with improved exact bounds on the number of arithmetic progressions and the sizes of offsets and periods in these progressions.

In [12], To pointed out that Chrobak's construction and Martinez's algorithm (whose correctness relies on correctness of Chrobak's construction) contain a subtle error, and he has shown modifications that correct this error. (In errata [2], Chrobak has corrected some errors from [1] not related to problems discussed here. The error pointed out by To was not corrected in the errata.) The construction in [3] is completely different from the construction in [1], and so it does not contain the above mentioned error. (To [12] does not mention the alternative construction from [3].)

In this paper, we give a simpler and more efficient algorithm for the same problem, i.e., for computing a set of arithmetic progressions representing the language accepted by a given unary NFA. The time complexity of the algorithm is $O(n^2(n + m))$ and its space complexity $O(n + m)$, where n is the number of states and m the number of transitions of the unary NFA. The algorithm and the proof of its correctness use some ideas from [3].

Section 2 gives basic definitions and formulates the main result, Section 3 describes the algorithm and a proof of its correctness, and Section 4 contains a description of an efficient implementation of the algorithm and an analysis of its complexity.

A preliminary version of results presented in this paper has been published on Workshop on Reachability Problems 2010 [10].

Acknowledgement: I would like to thank the anonymous referee for pointing out to the paper of Geffert [3], and for suggesting how ideas from [3] can be used to improve results and simplify proofs from the previous version of the article.

2. Definitions and Main Result

The set of natural numbers $\{0, 1, 2, \ldots\}$ is denoted by $\mathbb{N}$. For $i, j \in \mathbb{N}$ such that $i \leq j$, $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$, and $[i, j)$ denotes the (possibly empty) set $\{i, i+1, \ldots, j-1\}$. Given $c, d \in \mathbb{N}$, an *arithmetic progression* is the set $\{c + d \cdot i \mid i \in \mathbb{N}\}$, denoted $c + d\mathbb{N}$, where c is called the *offset* and d the *period* of the progression.

The following definitions are standard (see e.g. [6]), except that they are specialized to the case where a *unary* alphabet is used, i.e., an alphabet with one letter. In such alphabet, words can be identified with their lengths and the (only) letter of the alphabet is not important. This means that a word a^n can be identified with the number n. Since in the rest of the paper we consider only words over a unary alphabet, words are assumed to be elements of $\mathbb{N}$.

A *unary nondeterministic finite automaton (a unary NFA)* is a tuple $\mathcal{A} = (Q, \delta, I, F)$ where Q is a finite set of *states*, $\delta \subseteq Q \times Q$ is a *transition relation*, and $I, F \subseteq Q$ are sets of *initial* and *final* states, respectively.

Assume a unary NFA $\mathcal{A} = (Q, \delta, I, F)$. A *path* α of length k from q to q', where $q, q' \in Q$, is a sequence of states $q_0, q_1, \ldots, q_k$ from Q such that $q = q_0$, $q' = q_k$, and $(q_{i-1}, q_i) \in \delta$ for each $i \in [1, k]$. Path α is a *cycle* (or *loop*) if $k > 0$ and $q_0 = q_k$. We write $q \xrightarrow{x} q'$ to denote that there exists a path of length x from q to q', and we write $\overset{x}{\overset{\frown}{q}}$ to denote that there exists a cycle of length x in state q, i.e., $q \xrightarrow{x} q$ and $x \geq 1$. We also use notation like $q_1 \xrightarrow{x} \overset{y}{\overset{\frown}{q_2}} \xrightarrow{z} q_3$ to denote that $q_1 \xrightarrow{x} q_2$, $\overset{y}{\overset{\frown}{q_2}}$, and $q_2 \xrightarrow{z} q_3$.

A word $x \in \mathbb{N}$ is *accepted* by $\mathcal{A}$ if $q_0 \xrightarrow{x} q_f$ for some $q_0 \in I$ and $q_f \in F$. The language $L(\mathcal{A})$ accepted by a unary NFA $\mathcal{A}$ is the set of all words accepted by $\mathcal{A}$, i.e.,

$$L(\mathcal{A}) = \{x \in \mathbb{N} \mid \exists q_0 \in I : \exists q_f \in F : q_0 \xrightarrow{x} q_f\}.$$

The paper describes an efficient algorithm for the following problem:

PROBLEM: UNFA-Arith-Progressions

INPUT: A unary NFA $\mathcal{A}$.

OUTPUT: A set $\mathcal{R} = \{(c_1, d_1), (c_2, d_2), \ldots, (c_k, d_k)\}$ of pairs of natural numbers such that

$$L(\mathcal{A}) = \bigcup_{i=1}^{k} (c_i + d_i\mathbb{N}).$$

The main presented result is stated in the following theorem:

Theorem 2.1. There is an algorithm solving UNFA-Arith-Progressions with time complexity $O(n^2(n + m))$ and space complexity $O(n + m)$ where n is the number of states and m the number of transitions of a given unary NFA. Assuming that $n \geq 2$, the algorithm constructs a set $\mathcal{R}$ consisting of two subsets $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $|\mathcal{R}_1| \leq n^2$ and $|\mathcal{R}_2| \leq n$, where:

- each $(c, d) \in \mathcal{R}_1$ satisfies $d = 0$ and $c \in [0, n^2 - 1]$,

- each $(c, d) \in \mathcal{R}_2$ satisfies $d \in [1, n]$ and $c \in [n^2 - d - 1, n^2 - 2]$.

Remark: In the previous theorem and as well as in the rest of the paper we assume that $n \geq 2$. This makes no harm, since the trivial special case $n = 1$ can be easily dealt with separately.

3. $L(\mathcal{A})$ as the Union of Arithmetic Progressions

In this section, we describe the algorithm for UNFA-Arith-Progressions and prove its correctness.

In the rest of the paper, we assume a fixed unary NFA $\mathcal{A} = (Q, \delta, I, F)$ with $|Q| = n$ where $n \geq 2$, given as an input of the problem.

3.1. The Algorithm

As mentioned before, the algorithm computes the resulting set $\mathcal{R}$ as the union of sets $\mathcal{R}_1$ and $\mathcal{R}_2$:

- $\mathcal{R}_1$ is the set of all of pairs $(x, 0)$ where $x \in L(\mathcal{A})$ and $x \in [0, n^2)$, and
- $\mathcal{R}_2$ contains only pairs $(c, d) \subseteq \mathbb{N}^2$ with $d \in [1, n]$ and $c \in [n^2 - d - 1, n^2 - 2]$ where $q_0 \xrightarrow{n-1} \overset{d}{\underset{q}{\circlearrowleft}} \xrightarrow{c-(n-1)} q_f$ for some $q_0 \in I$, $q \in Q$, and $q_f \in F$ (note that $c \geq n - 1$). Only a subset of such pairs is included in $\mathcal{R}_2$. The exact definition of $\mathcal{R}_2$ will be given later.

At first, we can note that if $x \in c + d\mathbb{N}$ for some $(c, d) \in \mathcal{R}$, then $x \in L(\mathcal{A})$, because if $(c, d) \in \mathcal{R}_1$ then $c \in L(\mathcal{A})$ and $d = 0$ by the definition of $\mathcal{R}_1$, and so $c + d\mathbb{N} = \{c\}$ and $x = c$, which implies $x \in L(\mathcal{A})$; and if $(c, d) \in \mathcal{R}_2$ then there are $q_0 \in I$, $q \in Q$, and $q_f \in F$ such that $q_0 \xrightarrow{n-1} \overset{d}{\underset{q}{\circlearrowleft}} \xrightarrow{c-(n-1)} q_f$, and so $x \in L(\mathcal{A})$, because $q_0 \xrightarrow{x} q_f$ for each $x \in c + d\mathbb{N}$ due to the following trivial observation:

Observation 3.1. If $q_1 \xrightarrow{c_1} \overset{d}{\underset{q}{\circlearrowleft}} \xrightarrow{c_2} q_2$ for some $q_1, q, q_2 \in Q$ and $c_1, c_2, d \in \mathbb{N}$ then $q_1 \xrightarrow{x} q_2$ for each $x \in (c_1 + c_2) + d\mathbb{N}$.

Before defining the set $\mathcal{R}_2$, we need some technical definitions. The states of $\mathcal{A}$ can be partitioned into (maximal) *strongly connected components (SCCs)*; recall that states $q, q' \in Q$ belong the same strongly connected component C iff $q \xrightarrow{x} q'$ and $q' \xrightarrow{y} q$ for some $x, y \in \mathbb{N}$. A SCC C is *nontrivial* if it contains at least one transition, i.e., $q \xrightarrow{1} q'$ for some (not necessarily different) states $q, q' \in C$, and it is *trivial* otherwise (C contains exactly one state in this case). For $q \in Q$ we define value $sl(q)$ as the length of the shortest loop that can be done in q, i.e., when $sl(q) = d$ then $\overset{d}{\underset{q}{\circlearrowleft}}$ and we have $d' \geq d$ for each d' such that $\overset{d'}{\underset{q}{\circlearrowleft}}$. If there is no $d \geq 1$ such that $\overset{d}{\underset{q}{\circlearrowleft}}$ then $sl(q)$ is undefined. Note that for each nontrivial SCC C and each $q \in C$ we have $sl(q) \leq |C|$, since all states on the cycle of length $sl(q)$ from q to q belong to C, and no state (except q) is repeated on this cycle.

We call a state q *important* if q belongs to some nontrivial SCC C (which means that $sl(q)$ is defined) and the value $sl(q)$ is minimal in the given SCC C, i.e., for each $q' \in C$ we have $sl(q') \geq sl(q)$. The set of all important states in a given SCC C is denoted $imp(C)$ and the set of all important states of $\mathcal{A}$ is denoted Imp.

The set $\mathcal{R}_2$ is then defined as follows:

- $\mathcal{R}_2$ is the set of all pairs $(c, d) \subseteq \mathbb{N}^2$ where $q_0 \xrightarrow{n-1} \underset{q}{\overset{d}{\circlearrowleft}} \xrightarrow{c'} q_f$ for some $q_0 \in I$, $q \in Imp$, $q_f \in F$, and $c' \in [n^2 - n - d, n^2 - n - 1]$, with $d = sl(q)$ and $c = (n - 1) + c'$ (which means that $c \in [n^2 - d - 1, n^2 - 2]$).

Both $\mathcal{R}_1$ and $\mathcal{R}_2$ can be easily computed in polynomial time. To compute $\mathcal{R}_1$, it is sufficient to test for each $x \in [0, n^2)$ if $x \in L(\mathcal{A})$, and to compute $\mathcal{R}_2$, it is sufficient to test for each $q \in Imp$ (with $sl(q) = d$) and each possible value of $c \in [n^2-d-1, n^2-2]$, if the required conditions from the definition of $\mathcal{R}_2$ are satisfied. All these tests can be done in polynomial time by standard graph algorithms. A more efficient version of the algorithm, together with a more detailed analysis of its complexity, is described in Section 4.

It is obvious from the definitions of $\mathcal{R}_1$ and $\mathcal{R}_2$ that each pair (c, d) from these sets satisfies bounds on c and d specified in Theorem 2.1. It is also obvious that $\mathcal{R}_1$ contains at most n^2 pairs. The fact that $\mathcal{R}_2$ contains at most n pairs follows from the following observations. At first, note that if $(c, d) \in \mathcal{R}_2$ then $d = sl(q)$ for some $q \in Imp$. Moreover, for each such d there are at most d possible values of c such that $(c, d) \in \mathcal{R}_2$ since $c \in [n^2 - d - 1, n^2 - 2]$. Let $C_1, C_2, \ldots, C_k$ be all nontrivial SCCs of $\mathcal{A}$, and let $d_1, d_2, \ldots, d_k$ be the values such that $d_i = sl(q)$ for each $q \in C_i \cap Imp$ (note that d_i is uniquely determined by C_i and that $d_i \leq |C_i|$). Since $(c, d) \in \mathcal{R}_2$ implies that d is one of $d_1, d_2, \ldots, d_k$, we see that $|\mathcal{R}_2| \leq d_1 + d_2 + \cdots + d_k \leq |C_1| + |C_2| + \cdots + |C_k| \leq n$.

To prove the correctness of the algorithm, we need to show that for each $x \in L(\mathcal{A})$ there is some pair $(c, d) \in \mathcal{R}$ such that $x \in c + d\mathbb{N}$. For this we need the following crucial lemma whose proof is postponed to the following subsection:

Lemma 3.1. Let $x \geq n^2$. If $x \in L(\mathcal{A})$ then $x \in c + d\mathbb{N}$ for some $(c, d) \in \mathcal{R}_2$.

For each $x \in L(\mathcal{A})$ such that $x < n^2$ we have $(x, 0) \in \mathcal{R}_1$, and for each $x \in L(\mathcal{A})$ such that $x \geq n^2$ there is some pair $(c, d) \in \mathcal{R}_2$ such that $x \in c + d\mathbb{N}$ by Lemma 3.1. From this and from the previous discussion we obtain the following theorem, which proves the correctness of the algorithm:

Theorem 3.1. Let $x \in \mathbb{N}$. Then $x \in L(\mathcal{A})$ iff $x \in c + d\mathbb{N}$ for some $(c, d) \in \mathcal{R}$.

The rest of this section is devoted to the proof of Lemma 3.1.

3.2. Proof of Lemma 3.1

Lemma 3.1 follows from (a slight modification of) results from [3]. We present a complete proof of Lemma 3.1 here to make the article self-contained. Note that Proposition 3.1, Corollary 3.1, and Proposition 3.2 and their proofs are based on ideas from [3].

Remark: The notion of important states is not used in [3]. Instead, it defines a similar concept of *cardinal* states (with a more complicated definition; in particular, there is at most one cardinal state in each SCC).

The basic idea of the proof of Lemma 3.1 is following: given a path α from some $q_0 \in I$ to some $q_f \in F$ of length $x \geq n^2$, we can transform α into some other path α' from q_0 to q_f of the same length and of the following form:

- α' goes from q_0 to some important state q by $n-1$ steps,
- then α' goes through a cycle of length $d = sl(q)$ several times,
- and then α goes from q to q_f by c' steps, where $c' \in [n^2 - n - d, n^2 - n - 1]$.

This is done by a sequence of transformations, which are described by the following propositions.

Proposition 3.1. Let $q_1, q_2 \in Q$, $x \in \mathbb{N}$, and $d \geq 1$. If $q_1 \xrightarrow{x} q_2$ then $q_1 \xrightarrow{y} q_2$ for some $y \in [0, n \cdot d)$ such that $y \leq x$ and $y \equiv x \pmod{d}$.

Proof:
By induction on x. If $x < nd$ then we can put $y = x$ and we are done. So suppose $x \geq nd$ and let α be a path of length x from q_1 to q_2. The path α can be presented in the form

$$r_0 \xrightarrow{d} r_1 \xrightarrow{d} r_2 \xrightarrow{d} \cdots \xrightarrow{d} r_{n-1} \xrightarrow{d} r_n \xrightarrow{x-nd} q_2$$

where $r_0 = q_1$ and $r_i \xrightarrow{d} r_{i+1}$ for each $i \in [0, n-1]$. The sequence $r_0, r_1, \ldots, r_n$ contains $n+1$ states, and so by the pigeonhole principle there must be some i, j such that $0 \leq i < j \leq n$ and $r_i = r_j$. This means that $q_1 \xrightarrow{id} r_i = r_j \xrightarrow{(n-j)d} r_n \xrightarrow{x-nd} q_2$, i.e., $q_1 \xrightarrow{x'} q_2$ for $x' = x - (j-i)d$. Due to $x' < x$ we can apply the induction hypothesis, by which there is some $y < nd$ such that $q_1 \xrightarrow{y} q_2$ and $y \equiv x' \pmod{d}$. Since $x' \equiv x \pmod{d}$, we have $y \equiv x \pmod{d}$. □

Corollary 3.1. Let $q_1, q_2 \in Q$, $d \geq 1$, and $x \geq nd$. If $\overset{d}{\underset{q_1}{\curvearrowleft}} \xrightarrow{x} q_2$ then $q_1 \xrightarrow{x-d} q_2$.

Proof:
By Proposition 3.1, there is some $y < nd$ such that $q_1 \xrightarrow{y} q_2$ and $y \equiv x \pmod{d}$, which means that $y = x - kd$ for some $k \geq 1$. Together with $\overset{d}{\underset{q_1}{\curvearrowleft}}$ this implies $q_1 \xrightarrow{(k-1)d} q_1 \xrightarrow{y} q_2$, i.e., $q_1 \xrightarrow{x-d} q_2$. □

Proposition 3.2. Let $q_1, q_2 \in Q$, and $x \geq n^2$. If $q_1 \xrightarrow{x} q_2$ then $q_1 \xrightarrow{c} q \xrightarrow{x-c} q_2$ for some $q \in \mathit{Imp}$ and $c \in [0, n)$.

Proof:
Let α be a path from q_1 to q_2 of length x, i.e., a path of the form

$$s_0 \xrightarrow{1} s_1 \xrightarrow{1} \ldots \xrightarrow{1} s_x$$

where $s_0 = q_1$ and $s_x = q_2$. Let s_b be the first state on α that belongs to a nontrivial SCC (i.e., s_i belongs to a trivial SCC for $i \in [0, b)$), and let C be this nontrivial SCC containing s_b. So we have $q_1 \xrightarrow{b} s_b \xrightarrow{x-b} q_2$. Let β be one of the shortest paths from s_b to $\mathit{imp}(C)$, i.e., a path of the form

$$r_0 \xrightarrow{1} r_1 \xrightarrow{1} \ldots \xrightarrow{1} r_k$$

where $r_0 = s_b$, $r_k \in imp(C)$ and $r_i \notin imp(C)$ for $i \in [0, k)$, such that each path from s_b to $imp(C)$ has a length at least k. Note that states $q_1 = s_0, s_1, \dots, s_b = r_0, r_1, \dots, r_k$ are all different since states s_i, where $i \in [0, b)$, belong to separate trivial SCCs, and all states r_i, where $i \in [0, k]$, belong to the nontrivial SCC C and form the shortest path β. This means that $b + k \leq n - 1$, so if we show that $q_1 \xrightarrow{b+k} r_k \xrightarrow{x-(b+k)} q_2$ we are done, since $r_k \in Imp$.

Now we prove that the following holds for $i = 0, 1, \dots, k$:

$$q_1 \xrightarrow{b+i} r_i \xrightarrow{x-(b+i)} q_2 \tag{$*$}$$

The proof proceeds by induction on i. We have already proved $(*)$ for $i = 0$. Lets assume now that $(*)$ holds for some $i \in [0, k)$. We have $r_{i+1} \xrightarrow{\ell} r_i$ for some $\ell \in [1, n-1]$, since r_i and r_{i+1} are in the same SCC C. Consider now the path γ of the form

$$q_1 \xrightarrow{b+i} r_i \xrightarrow{1} r_{i+1} \xrightarrow{\ell} r_i \xrightarrow{x-(b+i)} q_2$$

of length $x + \ell + 1$. Note also that $\overset{\overset{\ell+1}{\curvearrowleft}}{r_{i+1}}$. Together with $r_{i+1} \xrightarrow{\ell+x-(b+i)} q_2$ this allows us, under the assumption that

$$\ell + x - (b + i) \geq n \cdot (\ell + 1) \tag{$**$}$$

to apply Corollary 3.1 to show that $r_{i+1} \xrightarrow{y} q_2$ for $y = \ell + x - (b + i) - (\ell + 1) = x - (b + i + 1)$, which means that

$$q_1 \xrightarrow{b+i+1} r_{i+1} \xrightarrow{x-(b+i+1)} q_2 \,,$$

i.e., that $(*)$ holds for $i + 1$.

To finish the proof, we just need to check that the assumption $(**)$ is actually fulfilled. Note that $(**)$ holds iff $(n - 1) \cdot \ell + n + (b + i) \leq x$. Due to $\ell \leq n - 1$ and $b + i \leq n - 1$, we have

$$(n - 1) \cdot \ell + n + (b + i) \leq (n - 1) \cdot (n - 1) + n + (n - 1) = n^2 \leq x \,,$$

which shows that $(**)$ really holds. □

In the proof of Proposition 3.3, two cases will be considered, either there is a state $q \in Imp$ such that $sl(q) = n$, or not. Note that in the former case the automaton $\mathcal{A}$ consists of the only one SCC containing all n states and that all these states are important, i.e., we have $sl(q) = n$ and $q \in Imp$ for each $q \in Q$. It is rather obvious that in this case $\mathcal{A}$ is a *trivial loop* of length n, i.e., it can be presented (after renaming states appropriately) in a form where $Q = \{q_1, q_2, \dots, q_n\}$ and $\delta = \{(q_i, q_{i+1}) \mid i \in [1, n-1]\} \cup \{(q_n, q_1)\}$.

So we have the following observation:

Observation 3.2. Exactly one of the following cases holds for $\mathcal{A}$:

- $\mathcal{A}$ is not a trivial loop, and then $sl(q) \leq n - 1$ for each $q \in Imp$, or
- $\mathcal{A}$ is a trivial loop of length n, and then $Imp = Q$ and $sl(q) = n$ for each $q \in Q$.

Proposition 3.3. Let $q_1, q_2 \in Q$, and $x \geq n^2$. If $q_1 \xrightarrow{x} q_2$ then $q_1 \xrightarrow{n-1} \overset{d}{\underset{q}{\curvearrowleft}} \xrightarrow{b} q_2$ for some $q \in \mathit{Imp}$, $d = \mathit{sl}(q)$, and $b \in [0, n^2 - n - 1]$ such that $x = (n-1) + k \cdot d + b$ for some $k \geq 0$.

Proof:
We can distinguish two cases depending on whether $\mathcal{A}$ is a trivial loop of length n or not:

a) *$\mathcal{A}$ is not a trivial loop*:

Let us assume that $q_1 \xrightarrow{x} q_2$ for a given $x \geq n^2$. By Proposition 3.2, $q_1 \xrightarrow{c} \overset{d}{\underset{q'}{\curvearrowleft}} \xrightarrow{x-c} q_2$ for some $c \in [0, n-1]$, $q' \in \mathit{Imp}$, and $d = \mathit{sl}(q')$. By Proposition 3.1, there is some $y \in [0, n \cdot d)$ such that $q' \xrightarrow{y} q_2$, $y \leq x - c$, and $y \equiv x - c \pmod{d}$. Since $\mathcal{A}$ is not a trivial loop, Observation 3.2 implies $d \leq n - 1$, and so $y < n \cdot (n-1) = n^2 - n$. We see that there is a path α of length x from q_1 to q_2 that goes from q_1 to q' by c steps, then several times through a cycle β of length d, and then goes from q' to q_2 by y steps, so $x = c + k' \cdot d + y$ for some $k' \geq 0$. From $x \geq n^2$ and $y \leq n^2 - n - 1$ follows that $c + k' \cdot d \geq n + 1$, which together with $c \leq n - 1$ implies that the state visited on α after the first $n - 1$ steps is on the cycle β. Let us denote this state q. Since all states on β belong to the same SCC as q' and all of them are important, we have $q \in \mathit{Imp}$ and $\mathit{sl}(q) = d$. Now we have $q_1 \xrightarrow{n-1} \overset{d}{\underset{q}{\curvearrowleft}} \xrightarrow{x-(n-1)} q_2$. We can apply Proposition 3.1 once more and obtain $q_1 \xrightarrow{n-1} \overset{d}{\underset{q}{\curvearrowleft}} \xrightarrow{b} q_2$ for some $b \geq 0$ such that $b \equiv x - (n-1) \pmod{d}$ and $b < nd \leq n(n-1) = n^2 - n$, which means that $b \in [0, n^2 - n - 1]$ and $x = (n-1) + k \cdot d + b$ for some $k \geq 0$.

b) *$\mathcal{A}$ is a trivial loop (of length n)*:
Let α be the path from q_1 to q_2 of length x (there is only one such path since $\mathcal{A}$ is deterministic) and let q be the state on α reached after first $n - 1$ steps. By Observation 3.2, $q \in \mathit{Imp}$ and $\mathit{sl}(q) = n$. Obviously, α is of the form

$$q_1 \xrightarrow{n-1} q \xrightarrow{n} q \xrightarrow{n} \cdots \xrightarrow{n} q \xrightarrow{b} q_2$$

where $b = (x - (n-1)) \bmod n$. Because $b \leq n - 1$ and because $n - 1 \leq n^2 - n - 1$ holds when $n \geq 2$, we have $b \leq n^2 - n - 1$.

□

Now we can finish the proof of Lemma 3.1:

Proof:
If $x \in L(\mathcal{A})$ then there are some $q_0 \in I$ and $q_f \in F$ such that $q_0 \xrightarrow{x} q_f$. If $x \geq n^2$, then by Proposition 3.3 we have $q_0 \xrightarrow{n-1} \overset{d}{\underset{q}{\curvearrowleft}} \xrightarrow{b} q_f$ for some $q \in \mathit{Imp}$, $d = \mathit{sl}(q)$, and $b \in [0, n^2 - n - 1]$, where $x = (n-1) + k \cdot d + b$ for some $k \geq 0$. This means that $q \xrightarrow{c'} q_f$ for each $c' \in b + d\mathbb{N}$. In particular, there is exactly one such c' in the interval $[n^2 - n - d, n^2 - n - 1]$. □

4. Efficient Implementation

In this section, we describe a more efficient implementation of the algorithm sketched in Subsection 3.1.

To avoid repeated computations during the construction of pairs in sets $\mathcal{R}_1$ and $\mathcal{R}_2$, the algorithm precomputes some information — the decomposition of $\mathcal{A}$ into SCCs, the values $sl(q)$ for each $q \in Q$, the set of important states Imp, and also the following sets:

- $S_i = \{q \in Q \mid \exists q_0 \in I : q_0 \xrightarrow{i} q\}$ for $i \in [0, n^2)$, and
- $T_i = \{q \in Q \mid \exists q_f \in F : q \xrightarrow{i} q_f\}$ for $i \in [0, n^2 - n - 1)$.

Informally, S_i contains all states that can be reached from some initial state by i steps, and T_i contains all states from which some final state can be reached by i steps.

The algorithm also computes the set $Q_{imp} = S_{n-1} \cap Imp$ of important states that can be reached by $n - 1$ steps from some initial state. Then it computes the set $\mathcal{D} = \{sl(q) \mid q \in Q_{imp}\}$, and assigns to each $d \in \mathcal{D}$ the set of those states q of Q_{imp} with $sl(q) = d$.

The algorithm then just checks the conditions from the definitions of $\mathcal{R}_1$ and $\mathcal{R}_2$ from Subsection 3.1:

- to construct $\mathcal{R}_1$, it generates for each $i \in [0, n^2)$ the set S_i, and adds the pair $(i, 0)$ to $\mathcal{R}_1$ iff $S_i \cap F \neq \emptyset$; and
- to construct $\mathcal{R}_2$, it does the following for each $c' \in [n^2 - 2n, n^2 - n - 1]$ and each $d \in \mathcal{D}$, such that $c' \geq n^2 - n - d$:
  it tests if there exists some $q \in Q_{imp}$ with $sl(q) = d$ such that $q \in T_{c'}$, and adds (c, d) with $c = (n - 1) + c'$ to $\mathcal{R}_2$, if there is such q.

Sets S_i and T_i can be stored in memory as bit arrays, so operations like testing if an element is member of a set, adding an element to a set, and so on, can be performed in a constant time.

The sets S_i can be computed as follows: $S_0 = I$ and $S_i = Succ(S_{i-1})$ for $i > 0$, where for a set of states $Q' \subseteq Q$, the set $Succ(Q')$ is defined as

$$Succ(Q') = \{q \in Q \mid \exists q' \in Q' : (q', q) \in \delta\}\,.$$

Similarly, $T_0 = F$, and $T_{i+1} = Pre(T_i)$ for $i \geq 0$, where $Pre(Q') = \{q \in Q \mid \exists q' \in Q' : (q, q') \in \delta\}$.

Let m be the number of transitions of $\mathcal{A}$ (i.e., $|\delta| = m$). Assuming that for each $q \in Q$ we have the sets of its immediate successors and predecessors (i.e., all $q' \in Q$ such that $(q, q') \in \delta$, and all $q' \in Q$ such that $(q', q) \in Q$) represented as lists, the sets $Succ(Q')$ and $Pre(Q')$ for a given $Q' \subseteq Q$ can be easily computed in time $O(n + m)$.

Using this, all the necessary sets S_i and T_i can be computed in time $O(n^2(n + m))$ as described above.

The decomposition of $\mathcal{A}$ into SCCs can be done by Tarjan's algorithm [11] in time $O(n + m)$, and the computation of $sl(q)$ for one Q can be done by the breath-first search in time $O(n + m)$, so it can be done in time $O(n(n + m))$ for all states in Q. The computation of sets Imp, Q_{imp} and partition of states q of Q_{imp} according to values of $sl(q)$ then can be done in time $O(n)$ once the set S_{n-1} has been computed.

When constructing $\mathcal{R}_1$, the test if $S_i \cap F \neq \emptyset$ can be done in time $O(n)$, and so the computation of $\mathcal{R}_1$ (including the computation of all S_i) takes time $O(n^2(n + m))$. When constructing $\mathcal{R}_2$, all computations for one value c' can be done in time $O(n)$ after the set $T_{c'}$ has been computed. So the computation of $\mathcal{R}_2$ takes time $O(n^2)$ if we do not count the time needed for the computation of sets T_i.

From all this we see that the total running time of the algorithm is $O(n^2(n+m))$.

It is not necessary to store all values of the precomputed sets during the computation. In particular, when constructing $\mathcal{R}_1$, the set S_{i-1} can be discarded after S_i has been computed, and for the computation of $\mathcal{R}_2$, we only need the set Q_{imp}, values of $sl(q)$ for $q \in Q_{imp}$, and the partition of states in Q_{imp} according to values of $sl(q)$. Space $O(n)$ is sufficient to store this information. All other sets, that were used only for the computation of this information, can be discarded after they were used. Also information about successors and predecessors of states can be stored in space $O(n+m)$, so the overall space complexity of the algorithm is $O(n+m)$.

Remark: Pairs of numbers produced by the algorithm can be written directly to the output and it is not necessary to store them in the memory, so the space taken by the produced output does not contribute to the space complexity of the algorithm.

This finishes the proof of Theorem 2.1.

References

[1] Chrobak, M.: Finite Automata and Unary Languages, *Theoretical Computer Science*, **47**(2), 1986, 149–158.

[2] Chrobak, M.: Errata to: "Finite Automata and Unary Languages" [Theoret. Comput. Sci. 47 (1986) 149–158], *Theoretical Computer Science*, **302**(1-3), 2003, 497–498.

[3] Geffert, V.: Magic numbers in the state hierarchy of finite automata, *Information and Computation*, **205**(11), 2007, 1652–1670.

[4] Göller, S., Mayr, R., To, A. W.: On the Computational Complexity of Verifying One-Counter Processes, *LICS'09*, IEEE Computer Society, 2009.

[5] Gruber, H., Holzer, M.: Computational Complexity of NFA Minimization for Finite and Unary Languages, *LATA'08*, LNCS 5196, Springer, 2008.

[6] Kozen, D. C.: *Automata and Computability*, Springer-Verlag, 1997.

[7] Martinez, A.: Efficient Computation of Regular Expressions from Unary NFAs, *Descriptional Complexity of Formal Systems (DFCS)*, 2002.

[8] Martinez, A.: *Topics in Formal Languages: String Enumeration, Unary NFAs and State Complexity*, Master Thesis, University of Waterloo, 2002.

[9] Parikh, R. J.: On context-free languages, *Journal of the ACM*, **13**(4), 1966, 570–581.

[10] Sawa, Z.: Efficient Construction of Semilinear Representations of Languages Accepted by Unary NFA, *4th International Workshop on Reachability Problems (RP 2010)*, LNCS 6227, Springer, 2010.

[11] Tarjan, R.: Depth-First Search and Linear Graph Algorithms, *SIAM Journal of Computing*, **1**(2), June 1972, 146–160.

[12] To, A. W.: Unary finite automata vs. arithmetic progressions, *Information Processing Letterss*, **109**(17), 2009, 1010–1014.