On the Complexity of Flanked Finite State Automata

Florent Avellaneda, Silvano Dal Zilio, Jean-Baptiste Raclet

On the Complexity of Flanked Finite State Automata

Florent Avellaneda[1,2,3], Silvano Dal Zilio[1,3], and Jean-Baptiste Raclet[2,3]

[1]CNRS, LAAS, F-31400 Toulouse, France
[2]IRIT, F-31400 Toulouse, France
[3]Univ de Toulouse, F-31400 Toulouse, France

We define a new subclass of nondeterministic finite automata for prefix-closed languages called *Flanked Finite Automata* (FFA). We show that this class enjoys good complexity properties while preserving the succinctness of nondeterministic automata. In particular, we show that the universality problem for FFA is in linear time and that language inclusion can be checked in polynomial time. A useful application of FFA is to provide an efficient way to compute the quotient and inclusion of regular languages without the need to use the powerset construction. These operations are the building blocks of several verification algorithms.

1 Introduction

While the problems of checking universality or language inclusion are known to be computationally easy for Deterministic Finite Automata (DFA), they are PSPACE-complete for Nondeterministic Finite Automata (NFA). On the other hand, the size of a NFA can be exponentially smaller than the size of an equivalent minimal DFA. This gap in complexity between the two models can be problematic in practice. This is for example the case when using finite state automata for system verification, where we need to manipulate very large number of states.

Several work have addressed this problem by trying to find classes of finite automata that retain the same complexity than DFA on some operations while still being more succinct than the minimal DFA. A good survey on the notion of determinism for automata is for example [4]. One such example is the class of Unambiguous Finite Automata (UFA) [9, 10]. Informally, a UFA is a finite state automaton such that, if a word is

accepted, then there is a unique run which witnesses this fact, that is a unique sequence of states visited when accepting the word. Like with DFA, the problems of universality and inclusion for UFA is in polynomial-time.

In this paper, we restrict our study to automaton accepting prefix closed languages. More precisely, we assume that all the states of the automaton are final (which corresponds exactly to the class of prefix-closed regular languages). This restriction is very common when using NFA for the purpose of system verification. For instance, Kripke structures used in model-checking algorithms are often interpreted as finite state automaton where all states are final. It is easy to see that, with this restriction on prefix-closed language, an UFA is necessarily deterministic. Therefore new classes of NFA, with the same nice complexity properties than UFA, are needed in this context. We can also note that the classical complexity results on NFA are still valid when we restrict to automata accepting prefix-closed language. For instance, given a NFA $\mathcal{A}$ with all its states final, checking the universality of $\mathcal{A}$ is PSPACE-hard [7]. Likewise for the minimization problem. Indeed there are examples of NFA with n states, all finals, such that the minimal equivalent DFA has 2^n states [7, Sect. 7]. We provide such an example in Sect. 5 of this paper. Therefore this restriction does not intrinsically change the difficulty of our task.

We define a new class of finite state automaton called *Flanked Finite Automata* (FFA) that has complexity properties similar to that of UFA but for prefix-closed language. Informally, a FFA includes extra-information that can be used to check efficiently if a word is not accepted by the automaton. In Sect. 3, we show that the universality problem for FFA is in linear-time while testing the language inclusion between two FFA $\mathcal{A}$ and $\mathcal{B}$ is in time $O(|\mathcal{A}|.|\mathcal{B}|)$, where $|\mathcal{A}|$ denotes the size of the automaton $\mathcal{A}$ in number of states. In Sect. 4, we define several operations on FFA. In particular we describe how to compute a flanked automata for the intersection, union and quotient of two languages defined by FFA. Finally, before concluding, we give an example of (a family of) regular languages that can be accepted by FFA which are exponentially more succinct than their equivalent minimal DFA.

Our main motivation for introducing this new class of NFA was to provide an efficient way to compute the quotient of two regular languages L_1 and L_2. This operation, denoted L_1/L_2 and defined in Sect. 4, is central in several automata-based verification problems that arise in applications ranging from the synthesis of discrete controller to the modular verification of component-based systems. For example, it has been used in the definition of contract-based specification theories [3, 2] or as a key operation for solving language equations [11]. With our approach, it is possible to construct the quotient of two flanked automaton, $\mathcal{A}_1$ and $\mathcal{A}_2$, using less than $|\mathcal{A}_1|.|\mathcal{A}_2| + 1$ states; moreover the resulting automata is still flanked. We believe that this work provides the first algorithm for computing the quotient of two languages without resorting to the powerset construction on the underlying automata, that is without determinizing them.

2 Notations and Definitions

A finite automaton is a quintuple $\mathcal{A} = (Q, \Sigma, E, I)$ where: Q is a finite set of states; Σ is the alphabet of $\mathcal{A}$ (that is a finite set of symbols); $E \subseteq Q \times \Sigma \times Q$ is the transition relation; and $I \subseteq Q$ is the set of initial states. In the remainder of this text, we always assume that every states of an automaton is final, hence we do not need a distinguished subset of accepting states. Without loss of generality, we also assume that every state in Q is reachable in $\mathcal{A}$ from I following a sequence of transitions in E.

For every word $u \in \Sigma^*$ we denote $\mathcal{A}(u)$ the subset of states in Q that can be reached when trying to accept the word u from an initial state in the automaton. We can define the set $\mathcal{A}(u)$ by induction on the word u. We assume that ϵ is the empty word and we use the notation $u\,a$ for the word obtained form u by concatenating the symbol $a \in \Sigma$; then:

$$\begin{array}{lcl} \mathcal{A}(\epsilon) & = & I \\ \mathcal{A}(u\,a) & = & \{q' \in Q \mid \exists q \in \mathcal{A}(u).(q, a, q') \in E\} \end{array}$$

By extension, we say that a word u is accepted by $\mathcal{A}$, denoted $u \in \mathcal{A}$, if the set $\mathcal{A}(u)$ is not empty.

Definition 1. *A Flanked Finite Automaton (FFA) is a pair $(\mathcal{A}, F)$ where $\mathcal{A} = (Q, \Sigma, E, I)$ is a finite automaton and $F : Q \times \Sigma$ is a "flanking function", that associates symbols of Σ to states of $\mathcal{A}$. We also require the following relation between $\mathcal{A}$ and F:*

$$\forall u \in \Sigma^*, a \in \Sigma.\big((u \in \mathcal{A} \wedge u\,a \notin \mathcal{A}) \Leftrightarrow \exists q \in \mathcal{A}(u).(q, a) \in F \big) \qquad (\mathrm{F}\star)$$

We will often use the notation $q \xrightarrow{a} q'$ when $(q, a, q') \in E$, that is when there is a transition from q to q' with symbol a in $\mathcal{A}$. Likewise, we use the notation $q \overset{a}{\nrightarrow}$ when $(q, a) \in F$.

With our condition that every state of an automaton is final, the relation $q \xrightarrow{a} q'$ states that every word u "reaching" q in $\mathcal{A}$ can be extended by the symbol a; meaning that $u\,a$ is also accepted by $\mathcal{A}$. Conversely, the relation $q \overset{a}{\nrightarrow}$ states that the word $u\,a$ is not accepted. Therefore, in a FFA $(\mathcal{A}, F)$, when $q \in \mathcal{A}(u)$ and $(q, a) \in F$, then we know that the word u cannot be extended with a. In other words, the flanking function gives information on the "frontier" of a prefix-closed language—the extreme limit over which words are no longer accepted by the automaton—hence the use of the noun *flank* to describe this class.

In the rest of the paper, we simply say that the pair $(\mathcal{A}, F)$ is *flanked* when condition (F$\star$) is met. We also say that the automaton $\mathcal{A}$ is *flankable* if there exist a flanking function F such that $(\mathcal{A}, F)$ is flanked.

2.1 Testing if a Pair $(\mathcal{A}, F)$ is Flanked

We can use the traditional Rabin-Scott powerset construction to test whether F flanks the automaton $\mathcal{A} = (Q, \Sigma, E, I)$. We build from $\mathcal{A}$ the "powerset automaton" $\wp(\mathcal{A})$, a DFA with alphabet Σ and with states in 2^Q (also called classes) that are the sets of states in Q reached after accepting a given word prefix; that is all the sets of the form

Figure 1: An example of non-flankable NFA (left) and its associated Rabin-Scott powerset construction (right).

$\mathcal{A}(u)$. The initial state of $\wp(\mathcal{A})$ is the class $\mathcal{A}(\epsilon) = I$. Finally, we have that $C \xrightarrow{a} C'$ in $\wp(\mathcal{A})$ if and only if there is $q \in C$ and $q' \in C'$ such that $q \xrightarrow{a} q'$.

Let $F^{-1}(a)$ be the set $\{q \mid q \overset{a}{\nrightarrow}\}$ of states that "forbids" the symbol a after a word accepted by $\mathcal{A}$. Then the pair $(\mathcal{A}, F)$ is flanked if, for every possible symbol $a \in \Sigma$ and for every reachable class $C \in \wp(\mathcal{A})$ we have: $C \cap F^{-1}(a) \neq \emptyset$ if and only if there are no class C' such that $C \xrightarrow{a} C'$.

This construction shows that checking if a pair $(\mathcal{A}, F)$ is flanked should be a costly operation, that is, it should be as complex as exploring a deterministic automaton equivalent to $\mathcal{A}$. In Sect. 3 we prove that this problem is actually PSPACE-complete.

2.2 Testing if a NFA is Flankable

It is easy to show that the class of FFA includes the class of deterministic finite state automaton; meaning that every DFA is flankable. If an automaton $\mathcal{A}$ is deterministic, then it is enough to choose the "flanking function" F such that, for every state q in Q, we have $q \overset{a}{\nrightarrow}$ if and only if there are no transitions of the form $q \xrightarrow{a} q'$ in $\mathcal{A}$. DFA are a proper subset of FFA; indeed we give examples of NFA that are flankable in Sect. 5.

On the other way, if an automaton is not deterministic, then in some cases it is not possible to define a suitable flanking function F. For example, consider the automaton from Fig. 1 and assume, by contradiction, that we can define a flankable function F for this automaton. The word b is accepted by $\mathcal{A}$ but the word $b\,b$ is not, so by definition of FFA (see eq. (F$\star$)), there must be a state $q \in \mathcal{A}(b)$ such that $q \overset{b}{\nrightarrow}$. Hence, because q_1 is the only state in $\mathcal{A}(b)$, we should necessarily have $q_1 \overset{b}{\nrightarrow}$. However, this contradicts the fact that the word $a\,b$ is in $\mathcal{A}$, since q_1 is also in $\mathcal{A}(a)$.

More generally, it is possible to define a necessary and sufficient condition for the existence of a flanking function; this leads to an algorithm for testing if an automaton $\mathcal{A}$ is flankable. Let $\mathcal{A}^{-1}(a)$ denotes the set of states reachable by words that can be extended by the symbol a (remember that we consider prefix-closed languages):

$$\mathcal{A}^{-1}(a) \;=\; \bigcup\{\mathcal{A}(u) \mid u\,a \in \mathcal{A}\}$$

It is possible to find a flanking function F for the automaton $\mathcal{A}$ if and only if, for every word $u \in \mathcal{A}$ such that $u\,a \notin \mathcal{A}$ then the set $\mathcal{A}(u) \setminus \mathcal{A}^{-1}(a)$ is not empty. Indeed, in

this case, it is possible to choose F such that $(q, a) \in F$ as soon as there exists a word u with $q \in \mathcal{A}(u) \setminus \mathcal{A}^{-1}(a)$.

Conversely, an automaton $\mathcal{A}$ is not flankable if we can find a word $u \in \mathcal{A}$ such that $u\,a \notin \mathcal{A}$ and $\mathcal{A}(u) \subseteq \mathcal{A}^{-1}(a)$. For example, for the automaton in Fig. 1, we have $\mathcal{A}^{-1}(b) = \{q_0, q_1, q_2\}$ while $b\,b \notin \mathcal{A}$ and $\mathcal{A}(b) = \{q_1\}$.

This condition can also be checked using the powerset construction. Indeed, we can compute the set $\mathcal{A}^{-1}(a)$ by taking the union of the classes in the powerset automaton $\wp(\mathcal{A})$ that are the source of an a transition. Then it is enough to test this set for inclusion against all the classes that have no outgoing transitions labeled with a in $\wp(\mathcal{A})$.

3 Complexity Results for Basic Problems

In this section we give some results on the complexity of basic operations over FFA.

Theorem 1. *The universality problem for FFA is decidable in linear time.*

Proof. We consider a FFA $(\mathcal{A}, F)$ with $\mathcal{A} = (Q, \Sigma, E, I)$ and we want to check that every word $u \in \Sigma^*$ is accepted by $\mathcal{A}$. We assume that Q and I are not empty and that every state is reachable in $\mathcal{A}$. We also assume that the function F is "encoded" a mapping from Q to sequences of symbols in Σ.

We start by proving that $\mathcal{A}$ is universal if and only if the relation F is empty; meaning that for all states $q \in Q$ it is not possible to find a symbol $a \in \Sigma$ such that $q \xrightarrow{a}$. As a consequence, all words reaching a state q in $\mathcal{A}$ can always be extended by any symbol of Σ.

$\mathcal{A}$ universal implies F empty. If the automaton $\mathcal{A}$ is universal then every word $u \in \Sigma^*$ is accepted by $\mathcal{A}$ and can be extended by any symbol $a \in \Sigma$. Hence, by definition of FFA (see eq. (F$\star$)) we have that $(q, a) \notin F$ for all symbol a in Σ. Hence F is the empty relation over $Q \times \Sigma$.

$\mathcal{A}$ not universal implies F not empty. Assume that u is the shortest word not accepted by $\mathcal{A}$. We have that $u \neq \epsilon$, since I is not empty. Hence there exist a word v such that $u = v\,a$ and v is accepted. Again, by definition of FFA (see eq. (F$\star$)), there must be a state $q \in \mathcal{A}(v)$ such that $q \xrightarrow{a}$; and therefore F is not empty.

As a consequence, to test whether $\mathcal{A}$ is universal, it is enough to check whether there is a state $q \in Q$ that is mapped to a non-empty set of symbols in F. Note that, given a different encoding of F, this operation could be performed in constant time. □

We can use this result to settle the complexity of testing if an automaton is flankable.

Theorem 2. *Given an automaton $\mathcal{A} = (Q, \Sigma, E, I)$ and a relation $F \in Q \times \Sigma$, the problem of testing if $(\mathcal{A}, F)$ is a flanked automaton is PSPACE-complete when there is at least two symbols in Σ.*

Proof. We can define a simple nondeterministic algorithm for testing is $(\mathcal{A}, F)$ is flanked. We recall that the function $F^{-1}(a)$ stands for the set $\{q \mid q \overset{a}{\not\to}\}$ of states that "forbids" the symbol a. As stated in Sect. 2.1, to test if $(\mathcal{A}, F)$ is flanked, we need, for every symbol $a \in \Sigma$, to explore the classes C in the powerset automaton of $\mathcal{A}$ and test whether $C \xrightarrow{a} C'$ in $\wp(\mathcal{A})$ and whether $C \cap F^{-1}(A) = \emptyset$ or not. These tests can be performed using $|Q|$ bits since every class C and every set $F^{-1}(a)$ is a subset of Q. Moreover there are at most $2^{|Q|}$ classes in $\wp(\mathcal{A})$. Hence, using Savitch's theorem, the problem is in PSPACE.

On the other way, we can reduce the problem of testing the universality of a NFA $\mathcal{A}$ to the problem of testing if a pair $(\mathcal{A}, \emptyset)$, where $\emptyset$ is the "empty" flanking function over $Q \times \Sigma$. The universality problem is known to be PSPACE-hard when the alphabet Σ is of size at least 2, even if all the states of $\mathcal{A}$ are final [7]. Indeed, to test if $\mathcal{A}$ is universal, we showed in the proof of the previous theorem, that it is enough to check that $(\mathcal{A}, \emptyset)$ is flanked. Hence our problem is also PSPACE-hard. □

To conclude this section, we prove that the complexity of checking language inclusion between a NFA and a FFA is in polynomial time, therefore proving that our new class of automata as the same nice complexity properties than those of UFA. We say that the language of $\mathcal{A}_1$ is included in $\mathcal{A}_2$, simply denoted $\mathcal{A}_1 \subseteq \mathcal{A}_2$, if all the words accepted by $\mathcal{A}_1$ are also accepted by $\mathcal{A}_2$.

Theorem 3. *Given a NFA $\mathcal{A}_1$ and a FFA $(\mathcal{A}_2, F_2)$, we can check whether $\mathcal{A}_1 \subseteq \mathcal{A}_2$ in polynomial time.*

Proof. Without loss of generality, we can assume that $\mathcal{A}_1 = (Q_1, \Sigma, E_1, I_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, E_2, I_2)$ are two NFA over the same alphabet Σ. We define a variant of the classical product construction between $\mathcal{A}_1$ and $\mathcal{A}_2$ that also takes into account the "pseudo-transitions" $q \overset{a}{\not\to}$ defined by the flanking functions.

We define the product of $\mathcal{A}_1$ and $(\mathcal{A}_2, F_2)$ as the NFA $\mathcal{A} = (Q, \Sigma, E, I)$ such that $I = I_1 \times I_2$ and $Q = (Q_1 \times Q_2) \cup \{\bot\}$. The extra state $\bot$ will be used to detect an "error condition", that is a word that is accepted by $\mathcal{A}_1$ and not by $\mathcal{A}_2$. The transition relation of $\mathcal{A}$ is such that:

- if $q_1 \xrightarrow{a} q_1'$ in $\mathcal{A}_1$ and $q_2 \xrightarrow{a} q_2'$ in $\mathcal{A}_2$ then $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ in $\mathcal{A}$;
- if $q_1 \xrightarrow{a} q_1'$ in $\mathcal{A}_1$ and $q_2 \overset{a}{\not\to}$ in $\mathcal{A}_2$ then $(q_1, q_2) \xrightarrow{a} \bot$ in $\mathcal{A}$

We can show that the language of $\mathcal{A}_1$ is included in the language of $\mathcal{A}_2$ if and only if the state $\bot$ is not reachable in $\mathcal{A}$. Actually, we show that any word u such that $\bot \in \mathcal{A}(u)$ is a word accepted by $\mathcal{A}_1$ and not by $\mathcal{A}_2$.

We prove the first implication. Assume that every word u accepted by $\mathcal{A}_1$ is accepted by $\mathcal{A}_2$. Hence we can prove by induction on the size of u that $\mathcal{A}(u) \subseteq Q_1 \times Q_2$. On the other way, if u is not accepted by $\mathcal{A}_1$ then u is not accepted by $\mathcal{A}$ (there are no transitions in this case). Hence, for all words in Σ^*, the set $\mathcal{A}(u)$ does not contain $\bot$.

For the other direction, assume that there is a word u such that $\bot \in \mathcal{A}(u)$. The word u cannot be ϵ since $\mathcal{A}(\epsilon) = I_1 \times I_2 \not\ni \bot$. Therefore u is of the form $v\,a$. Since there are no transitions from $\bot$ in $\mathcal{A}$, there must be a pair $(q_1, q_2) \in Q_1 \times Q_2$ such that $q_1 \in \mathcal{A}_1(v)$;

$q_2 \in \mathcal{A}_2(v)$; $q_1 \xrightarrow{a} q_1'$ in $\mathcal{A}_1$ and $q_2 \xrightarrow{a}$ in $\mathcal{A}_2$. By property (F$\star$), since $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$ are both flanked, we have that $v\,a \in \mathcal{A}_1$ and $v\,a \notin \mathcal{A}_2$, as required.

We cannot generate more than $|Q_1|.|Q_2|$ reachable states in $\mathcal{A}$ before finding the error $\perp$ (or stopping the construction). Hence this algorithm is solvable in polynomial time. □

4 Closure Properties of Flanked Automata

In this section, we study how to compute the composition of flanked automata. We prove that the class of FFA is closed by language intersection and by the "intersection adjunct", also called quotient. On a negative side, we show that the class is not closed by non-injective relabeling.

We consider the problem of computing a flanked automaton accepting the intersection of two prefix-closed, regular languages. More precisely, given two FFA $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$, we want to compute a FFA $(\mathcal{A}, F)$ that recognizes the set of words accepted by both $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted simply $\mathcal{A}_1 \cap \mathcal{A}_2$.

Theorem 4. *Given two FFA $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$, we can compute a FFA $(\mathcal{A}, F)$ for the language $\mathcal{A}_1 \cap \mathcal{A}_2$ in polynomial time. The NFA $\mathcal{A}$ has size less than $|\mathcal{A}_1|.|\mathcal{A}_2|$.*

Proof. We define a classical product construction between $\mathcal{A}_1$ and $\mathcal{A}_2$ and show how to extend this composition on the flanking functions. We assume that $\mathcal{A}_i$ is an automaton (Q_i, Σ, E_i, I_i) for $i \in \{1, 2\}$.

The automaton $\mathcal{A} = (Q, \Sigma, E, I)$ is defined as the synchronous product of $\mathcal{A}_1$ and $\mathcal{A}_2$, that is: $Q = Q_1 \times Q_2$; $I = I_1 \times I_2$; and the transition relation is such that $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ in $\mathcal{A}$ if both $q_1 \xrightarrow{a} q_1'$ in $\mathcal{A}_1$ and $q_2 \xrightarrow{a} q_2'$ in $\mathcal{A}_2$. It is a standard result that $\mathcal{A}$ accepts the language $\mathcal{A}_1 \cap \mathcal{A}_2$.

The *flanking function* F is defined as follows: for each accessible state $(q_1, q_2) \in Q$, we have $(q_1, q_2) \xrightarrow{a}$ if and only if $q_1 \xrightarrow{a}$ in $\mathcal{A}_1$ or $q_2 \xrightarrow{a}$ in $\mathcal{A}_2$. What is left to prove is that $(\mathcal{A}, F)$ is flanked, that is, we show that condition (F$\star$) is correct:

- assume u is accepted by $\mathcal{A}$ and $u\,a$ is not; then there is a state $q = (q_1, q_2)$ in $\mathcal{A}$ such that $q \in \mathcal{A}(u)$ and $(q, a) \in F$. By definition of $\mathcal{A}$, we have that u is accepted by both $\mathcal{A}_1$ or $\mathcal{A}_2$, while the word $u\,a$ is not accepted by at least one of them. Assume that $u\,a$ is not accepted by $\mathcal{A}_1$. Since F_1 is a flanking function for $\mathcal{A}_1$, we have by equation (F$\star$) that $(q_1, a) \in F_1$; and therefore $(q, a) \in F$, as required.

- assume there is a reachable state $q = (q_1, q_2)$ in $\mathcal{A}$ such that $q \in \mathcal{A}(u)$ and $(q, a) \in F$; then u is accepted by $\mathcal{A}$. We show, by contradiction, that $u\,a$ cannot be accepted by $\mathcal{A}$, that is $u\,a \notin \mathcal{A}_1 \cap \mathcal{A}_2$. Indeed, if so, then $u\,a$ will be accepted both by $\mathcal{A}_1$ and $\mathcal{A}_2$ and therefore we will have $(q_1, a) \notin F_1$ and $(q_2, a) \notin F_2$, which contradicts the fact that $(q, a) \in F$.

□

Next we consider the adjunct of the intersection operation, denoted $\mathcal{A}_1/\mathcal{A}_2$. This operation, also called *quotient*, is defined as the biggest prefix-closed language X such that $\mathcal{A}_2 \cap X \subseteq \mathcal{A}_1$. Informally, X is the solution to the following question: what is the biggest set of words x such that x is either accepted by $\mathcal{A}_1$ or not accepted by $\mathcal{A}_2$. Therefore the language $\mathcal{A}_1/\mathcal{A}_2$ is always defined (and not empty), since it contains at least the empty word ϵ. Actually, the quotient can be interpreted as the biggest prefix-closed language included in the set $L_1 \cup \bar{L_2}$, where L_1 is the language accepted by $\mathcal{A}_1$ and $\bar{L_2}$ is the complement of the language of $\mathcal{A}_2$. The quotient operation can also be defined by the following two axioms:

$$\text{(Ax1)} \quad \mathcal{A}_2 \cap (\mathcal{A}_1/\mathcal{A}_2) \subseteq \mathcal{A}_1 \qquad\qquad \text{(Ax2)} \quad \forall X.\, \mathcal{A}_2 \cap X \subseteq \mathcal{A}_1 \Rightarrow X \subseteq \mathcal{A}_1/\mathcal{A}_2$$

The quotient operation is useful when trying to solve *language equations problems* [11] and has applications in the domain of system verification and synthesis. For instance, we can find a similar operation in the contract framework of Benveniste et al. [3] or in the contract framework of Bauer et al. [2].

Our results on FFA can be use for the simplest instantiation of these frameworks, that considers a simple trace-based semantics where the behavior of systems is given as a regular set of words; composition is language intersection; and implementation refinement is language inclusion. Our work was motivated by the fact that there are no known effective methods to compute the quotient. Indeed, to the best of our knowledge, all the approaches rely on the determinization of NFA, which is very expensive in practice [8, 11].

Our definitions of quotient could be easily extended to replace language intersection by synchronous product and to take into account the addition of modalities [8].

Theorem 5. *Given two FFA $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$, we can compute a FFA $(\mathcal{A}, F)$ for the quotient language $\mathcal{A}_1/\mathcal{A}_2$ in polynomial time. The NFA $\mathcal{A}$ has size less than $|\mathcal{A}_1|.|\mathcal{A}_2| + 1$*

Proof. Without loss of generality, we can assume that $\mathcal{A}_1 = (Q_1, \Sigma, E_1, I_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, E_2, I_2)$ are two NFA over the same alphabet Σ. Like in the construction for testing language inclusion, we define a variant of the classical product construction between $\mathcal{A}_1$ and $\mathcal{A}_2$ that also takes into account the flanking functions.

We define the product of $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$ as the NFA $\mathcal{A} = (Q, \Sigma, E, I)$ such that $I = I_1 \times I_2$ and $Q = (Q_1 \times Q_2) \cup \{\top\}$. The extra state $\top$ will be used as a sink state from which every suffix can be accepted. The transition relation of $\mathcal{A}$ is such that:

- if $q_1 \xrightarrow{a} q_1'$ in $\mathcal{A}_1$ and $q_2 \xrightarrow{a} q_2'$ in $\mathcal{A}_2$ then $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ in $\mathcal{A}$;
- if $q_2 \overset{a}{\not\to}$ in $\mathcal{A}_2$ then $(q_1, q_2) \xrightarrow{a} \top$ in $\mathcal{A}$ for all state $q_1 \in Q_1$
- $\top \xrightarrow{a} \top$ for every $a \in \Sigma$

Note that we do not have a transition rule for the case where $q_1 \overset{a}{\not\to}$ in $\mathcal{A}_1$ and $q_2 \xrightarrow{a} q_2'$; this models the fact that a word "that can be extended" in $\mathcal{A}_2$ but not in $\mathcal{A}_1$ cannot be

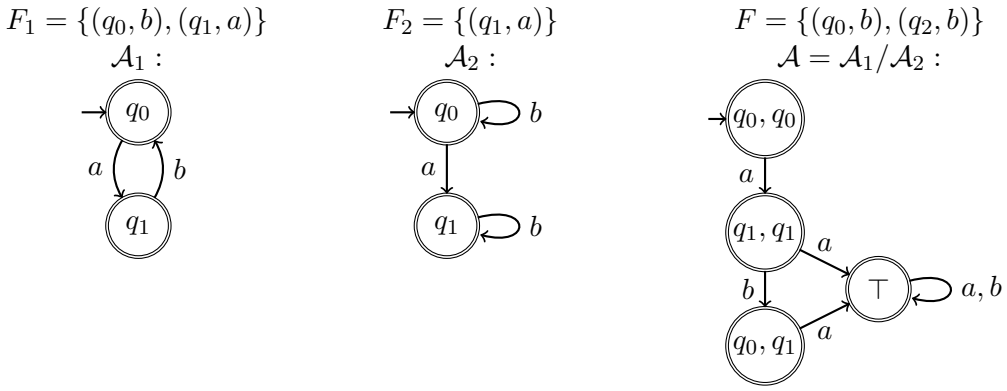

Figure 2: Construction for the quotient of two FFA $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$.

in the quotient $\mathcal{A}_1/\mathcal{A}_2$. It is not difficult to show that $\mathcal{A}$ accepts the language $\mathcal{A}_1/\mathcal{A}_2$. We give an example of the construction in Figure 2.

Next we show that $\mathcal{A}$ is flankable and define a suitable flanking function. Let F be the relation in $Q \times \Sigma$ such that $(q_1, q_2) \stackrel{a}{\not\to}$ if and only if $q_1 \stackrel{a}{\not\to}$ in F_1 and $q_2 \stackrel{a}{\to} q_2'$ in $\mathcal{A}_2$. That is, the symbol a is forbidden exactly in the case that was ruled out in the transition relation of $\mathcal{A}$. What is left to prove is that $(\mathcal{A}, F)$ is flanked, that is, we show that condition (F$\star$) is correct:

- Assume u is accepted by $\mathcal{A}$ and $u\,a$ is not. Since $u\,a$ is not accepted, it must be the case that $q \neq \top$. Therefore there is a state $q = (q_1, q_2)$ in $\mathcal{A}$ such that $q_1 \in \mathcal{A}_1(u)$ and $q_2 \in \mathcal{A}_2(u)$. Also, since there are no transition with label a from q, then necessarily $q_1 \stackrel{a}{\not\to}$ in $\mathcal{A}_1$ and $q_2 \stackrel{a}{\to} q_2'$. This is exactly the case where $(q, a) \in F$, as required.

- Assume there is a reachable state q in $\mathcal{A}$ such that $q \in \mathcal{A}(u)$ and $(q, a) \in F$. Since $(q, a) \in F$, we have $q \neq \top$ and therefore $q = (q_1, q_2)$ with $q_1 \in \mathcal{A}_1(u)$ and $q_2 \in \mathcal{A}_2(u)$. Hence u is accepted by $\mathcal{A}$. Next, we show by contradiction that $u\,a$ cannot be accepted by $\mathcal{A}$. Indeed, if it was the case then $u\,a \in \mathcal{A}_2$ and $u\,a \notin \mathcal{A}_2$. However, if $u\,a \in \mathcal{A}_2$ then, $(q_2, a) \notin F_2$ and so, by construction, $((q_1, q_2), a) \notin F$.

□

We give an example of the construction of the "quotient" FFA in Fig. 2. If we look more closely at the construction used in Theorem 5, that defines an automaton for the quotient of two FFA $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$, we see that the flanking function F_1 is used only to compute the flanking function of the result. Therefore, as a corollary, it is not difficult to prove that we can use the same construction to build a quotient automaton for $\mathcal{A}_1/\mathcal{A}_2$ from an arbitrary NFA $\mathcal{A}_1$ and a FFA $(\mathcal{A}_2, F_2)$. However the resulting automaton may not be flankable.

We can also prove that flankability is preserved by language union: given two FFA $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$, we can compute a FFA $(\mathcal{A}, F)$ that recognizes the set of words

accepted either by $\mathcal{A}_1$ or by $\mathcal{A}_2$, denoted $\mathcal{A}_1 \cup \mathcal{A}_2$. Operations corresponding to the adjunct of the union or the to Kleene star closure are not interesting in the context of automaton where every state is final and therefore they are not studied in this paper.

Theorem 6. *Given two FFA $(\mathcal{A}_1, F_1)$ and $(\mathcal{A}_2, F_2)$, we can compute a FFA $(\mathcal{A}, F)$ for the language $\mathcal{A}_1 \cup \mathcal{A}_2$ in polynomial time. The NFA $\mathcal{A}$ has size less than $(|\mathcal{A}_1|+1).(|\mathcal{A}_2|+1)$.*

Proof. Like for language intersection and language inclusion, we base our construction on a variant of the classical product construction between $\mathcal{A}_1$ and $\mathcal{A}_2$ and show how to extend this composition on the flanking functions. We assume that $\mathcal{A}_i$ is an automaton (Q_i, Σ, E_i, I_i) for $i \in \{1, 2\}$ and that both automaton have the same alphabet.

We consider a special state symbol $\top$ not in $Q_1 \cup Q_2$. This state will be used in $\mathcal{A}$ when we start accepting words that are not in the intersection of $\mathcal{A}_1$ and $\mathcal{A}_2$. The automaton $\mathcal{A} = (Q, \Sigma, E, I)$ is such that: $Q \subseteq (Q_1 \cup \{\top\}) \times (Q_2 \cup \{\top\})$; $I = I_1 \times I_2$; and the transition relation is such that:

- if $q_1 \xrightarrow{a} q_1'$ in $\mathcal{A}_1$ and $q_2 \xrightarrow{a} q_2'$ in $\mathcal{A}_2$ then $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ in $\mathcal{A}$;
- if $q_1 \xrightarrow{a} q_1'$ in $\mathcal{A}_1$ and $q_2 \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in $\mathcal{A}_2$ then $(q_1, q_2) \xrightarrow{a} (q_1', \top)$ in $\mathcal{A}$;
- if $q_1 \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in $\mathcal{A}_1$ and $q_2 \xrightarrow{a} q_2'$ in $\mathcal{A}_2$ then $(q_1, q_2) \xrightarrow{a} (\top, q_2')$ in $\mathcal{A}$;
- if $q_1 \xrightarrow{a} q_1'$ in $\mathcal{A}_1$ then $(q_1, \top) \xrightarrow{a} (q_1', \top)$ in $\mathcal{A}$;
- if $q_2 \xrightarrow{a} q_2'$ in $\mathcal{A}_2$ then $(\top, q_2) \xrightarrow{a} (\top, q_2')$ in $\mathcal{A}$.

It is not difficult to prove that the NFA $\mathcal{A}$ accepts all the words in $\mathcal{A}_1 \cup \mathcal{A}_2$.

The *flanking function* F is defined as the smallest relation such that, for each accessible state $(q_1, q_2) \in Q_1 \times Q_2$:

- if both $q_1 \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in F_1 and $q_2 \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in F_2 then $(q_1, q_2) \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in F;
- if $q_1 \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in F_1 then $(q_1, \top) \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in F;
- if $q_2 \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in F_2 then $(\top, q_2) \xrightarrow{a}\!\!\!\!\!\!\rightarrow$ in F

We are left to prove that $(\mathcal{A}, F)$ is flanked, that is condition (F$\star$) is correct. The proof is very similar to the one for Theorem 4. □

The two main closure properties given in this section are useful when we want to check language inclusion between the composition of several languages; for example if we need to solve, for X, the equation $\mathcal{A}_1 \cap \cdots \cap \mathcal{A}_n \cap X \subseteq \mathcal{B}$. This is the case, for example, if we need to synthesize a discrete controller, X, that satisfies a given requirement specification $\mathcal{B}$ when put in parallel with components whose behavior is given by $\mathcal{A}_i$ (with $i \in 1..n$). Indeed, even though there may be a small price to pay to "flank" the sub-components of this equation, we can incrementally build a flanked automaton for $\mathcal{A}_1 \cap \cdots \cap \mathcal{A}_n$ and then compute efficiently the quotient $\mathcal{B}/(\mathcal{A}_1 \cap \cdots \cap \mathcal{A}_n)$.

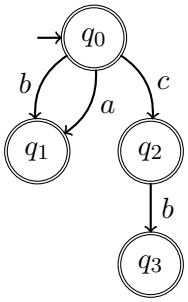

Figure 3: Example of a FFA not flankable after relabeling c to a.

Even though the class of FFA enjoys interesting closure properties, there are operations that, when applied to a FFA, may produce a result that is not flankable. This is for example the case with "(non-injective) relabeling", that is the operation of applying a substitution over the symbols of an automaton. The same can be observed if we consider an erasure operation, in which we can replace all transition on a given symbol by an ϵ-transition. Informally, it appears that the property flankable can be lost when applying an operation that increases the non-determinism of the transition relation.

We can prove this result by exhibiting a simple counterexample, see the automaton in Fig. 3. This automaton with alphabet $\Sigma = \{a, b, c\}$ is deterministic, so we can easily define an associated flanking function. For example we can choose $F = \{(q_1, a), (q_1, b), (q_1, c), (q_2, a), (q_2, c), (q_3, a), (q_3, b), (q_3, c)\}$. However, if we substitute the symbol c with a (that is we apply the non-injective relabeling function $\{a \leftarrow a\}\{b \leftarrow b\}\{c \leftarrow a\}$), we obtain the non-flankable automaton described in Sect. 2.1 (see Fig. 1).

5 Succinctness of Flanked Automata

In this section we show that a flankable automata can be exponentially more succinct than its equivalent minimal DFA. This is done by defining a language over an alphabet of size $2\,n$ that can be accepted by a linear size FFA but that corresponds to a minimal DFA with an exponential number of states. This example is due to Thomas Colcombet [5].

At first sight, this result may seem quite counterintuitive. Indeed, even if a flanked automata is build from a NFA, the combination of the automaton and the flanking function contains enough information to "encode" both a language and its complement. This is what explain the good complexity results on testing language inclusion for example. Therefore we could expect worse results concerning the relative size of a FFA and an equivalent DFA.

Theorem 7. *For every integer n, we can find a FFA $(\mathcal{A}_n, F)$ such that $\mathcal{A}_n$ has $2\,n + 2$ states and that the language of $\mathcal{A}_n$ cannot be accepted by a DFA with less than 2^n states.*

Proof. We consider two alphabets with n symbols: $\Pi_n = \{1, \ldots, n\}$ and $\Theta_n = \{\sharp_1, \sharp_2, \ldots, \sharp_n\}$. We define the language L_n over the alphabet $\Pi_n \cup \Theta_n$ as the smallest set of words such that:

- all words in Π_n^* are in L_n, that is all the words that do not contain a symbol of the kind $\sharp_i$;
- a word of the form $(u\,\sharp_i)$ is in L_n if and only if u is a word of Π_n^* that contains at least one occurrence of the symbol i. That is L_n contains all the words of the form $\Pi_n^* \cdot i \cdot \Pi_n^* \cdot \sharp_i$ for all $i \in 1..n$. We denote L_n^i the regular language consisting of the words of the form $\Pi_n^* \cdot i \cdot \Pi_n^* \cdot \sharp_i$.

Clearly the language L_n is the union of $n+1$ regular languages; $L = \Pi_n^* \cup L_n^1 \cup \cdots \cup L_n^n$. It is also easy to prove that L_n is prefix-closed, since the set of prefixes of the words in L_n^i is exactly Π_n^* for all $i \in 1..n$.

A DFA accepting the language L_n must have at least 2^n different states. Indeed it must be able to record the subset of symbols in Π_n that have already been seen before accepting $\sharp_i$ as a final symbol; to accept a word of the form $u\,\sharp_i$ the DFA must know whether i has been seen in u for all possible $i \in 1..n$.

Next we define a flankable NFA $\mathcal{A}_n = (Q_n, \Pi_n \cup \Theta_n, E_n, \{p\})$ with $2\,n+2$ states that can recognize the language L_n. We give an example of the construction in Fig. 4 for the case $n = 3$. The NFA $\mathcal{A}_n$ has a single initial state, p, and a single sink state (a state without outgoing transitions), r. The set Q_n also contains two states, p_i and q_i, for every symbol i in Π.

The transition relation E_n is the smallest relation that contains the following triplets for all $i \in 1..n$:

- the 3 transitions $p \xrightarrow{i} q_i$; $p_i \xrightarrow{i} q_i$; and $q_i \xrightarrow{i} q_i$;
- for every index $j \neq i$, the 3 transitions $p \xrightarrow{j} p_i$; $p_i \xrightarrow{j} p_i$; and $q_i \xrightarrow{j} q_i$;
- and the transition $q_i \xrightarrow{\sharp_i} r$.

Intuitively, a transition from p to p_i or q_i will select non-deterministically which final symbol $\sharp_i$ is expected at the end of the word (which sub-language L_n^i we try to accept). Once a symbol in Θ has been seen—in one of the transition of the kind $q_i \xrightarrow{\sharp_i} r$—the automaton is stuck on the state r. It is therefore easy to prove that $\mathcal{A}_n$ accepts the union of the languages L_n^i and their prefixes.

Finally, the NFA $\mathcal{A}_n$ is flankable. It is enough to choose, for the flanking function, the smallest relation on $Q \times \Theta_n$ such that $p_i \overset{\sharp_i}{\nrightarrow}$ and $p \overset{\sharp_i}{\nrightarrow}$ for all $i \in 1..n$; and such that $r \overset{a}{\nrightarrow}$ for all the symbols $a \in \Pi_n \cup \Theta_n$. Indeed, it is not possible to accept the symbol $\sharp_i$ from the initial state, p, or from a word that can reach p_i; that is, it is not possible to extend a word without any occurrence of the symbol i with the symbol $\sharp_i$. Also, it is not possible to extend a word that can reach the state r in $\mathcal{A}_n$. It is easy to prove that this cover all the possible words not accepted by $\mathcal{A}_n$. □

6 Conclusion

We define a new subclass of NFA for prefix-closed languages called flanked automata. Intuitively, a FFA $(\mathcal{A}, F)$ is a simple extension of NFA where we add in the relation F

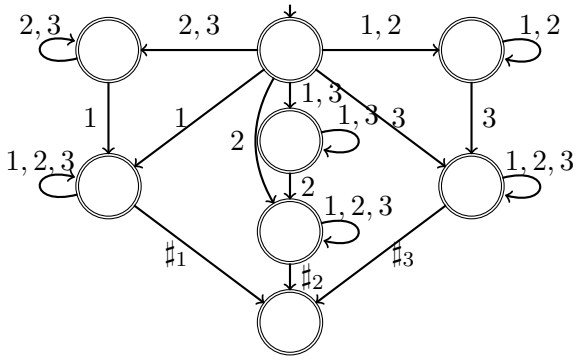

Figure 4: Flankable NFA for the language L_3.

extra information that can be used to check (non-deterministically) whether a word is not accepted by $\mathcal{A}$. Hence a FFA can be used both to test whether a word is in the language associated to $\mathcal{A}$ or in its complement. As a consequence, we obtain good complexity results for several interesting problems: universality, language inclusion, ... This idea of adding extra-information to encode both a language and its complement seems to be new. It is also quite different from existing approaches used to to define subclasses of NFA with good complexity properties, like for example unambiguity [9, 10]. Our work could be extended in several ways.

First, we have implemented all our proposed algorithms and constructions and have found that—for several examples coming from the system verification domain—it was often easy to define a flanking function for a given NFA (even though we showed in Sect. 2.2 that it is not always possible). More experimental work is still needed, and in particular the definition of a good set of benchmarks.

Next, we have used the powerset construction multiple time in our definitions. Most particularly as a way to test if a FFA is flanked or if a NFA is flankable. Other constructions used to check language inclusion or simulation between NFA could be useful in this context like, for example, the antichain-based method [1].

Finally, we still do not know how to compute a "succinct" flanked automaton from a NFA that is not flankable. At the moment, our only solution is to compute a minimal equivalent DFA (since DFA are always flankable). While it could be possible to subsequently simplify the DFA—which is known to be computationally hard [6], even without taking into account the flanked function—it would be interesting to have a more direct construction. This interesting open problem is left for future investigations.

Acknowledgments We thank Denis Kuperberg, Thomas Colcombet, and Jean-Eric Pin for providing their expertise and insight and for suggesting the example that led to the proof of Theorem 7.

References

[1] Parosh Aziz Abdulla, Yu-Fang Chen, Lukas Holik, Richard Mayr, and Tomas Vojnar. When simulation meets antichains. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 6015 of *LNCS*. Springer, 2010.

[2] Sebastian S. Bauer, Alexandre David, Rolf Hennicker, Kim Guldstrand Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. Moving from specifications to contracts in component-based design. In *Fundamental Approaches to Software Engineering*, volume 7212 of *LNCS*, pages 43–58. Springer, 2012.

[3] Albert Benveniste, Benoît Caillaud, Alberto Ferrari, Leonardo Mangeruca, Roberto Passerone, and Christos Sofronis. Multiple viewpoint contract-based specification and design. In *Formal Methods for Components and Objects*, volume 5382 of *LNCS*, pages 200–225. Springer, 2008.

[4] Thomas Colcombet. Forms of Determinism for Automata. In *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14, pages 1–23, 2012.

[5] Thomas Colcombet. Flankable automata may be exponentially more succint than deterministic one. private communication, March 2015.

[6] Tao Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

[7] Jui-Yi Kao, Narad Rampersad, and Jeffrey Shallit. On NFAs where all states are final, initial, or both. *Theoretical Computer Science*, 410(4749):5010–5021, 2009.

[8] Jean-Baptiste Raclet. Residual for component specifications. *Electronic Notes in Theoretical Computer Science*, 215:93–110, 2008. Proceedings of the 4th International Workshop on Formal Aspects of Component Software (FACS 2007).

[9] E. M. Schmidt. *Succinctness of Description of Context-Free, Regular and Unambiguous Languages*. PhD thesis, Cornell University, 1978.

[10] Richard Edwin Stearns and Harry B Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3):598–611, 1985.

[11] T. Villa, A. Petrenko, N. Yevtushenko, A. Mishchenko, and R. Brayton. Component-based design by solving language equations. *Proceedings of the IEEE*, PP(99):1–16, 2015.