Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 229 (2009) 19–35
www.elsevier.com/locate/entcs

Security Policies Enforcement Using Finite Edit Automata

Danièle Beauquier [a,1] Joëlle Cohen[a,2] Ruggero Lanotte[b,3]

[a] *LACL*
University Paris-Est
CRETEIL, FRANCE

[b] *Dipartimento di Scienze della cultura, Politiche e dell'Informazione*
Università dell'Insubria
COMO, ITALIA

Abstract

Edit automata have been introduced by J.Ligatti et al. as a model for security enforcement mechanisms which work at run time. In a distributed interacting system, they play a role of monitor that runs in parallel with a target program and transforms its execution sequence into a sequence that obeys the security property. In this paper we characterize security properties which are enforceable by finite edit automata, i.e. edit automata with a finite set of states. We prove that these properties are a sub-class of ω-regular sets. Moreover given an ω-regular set P, one can decide in time $O(n^2)$ whether P is enforceable by a finite edit automaton (where n is the number of states of the finite automaton recognizing P) and we give an algorithm to synthesize the controller.

Keywords: Controller, finite edit automata, security.

1 Introduction

Security enforcement mechanisms are used to prevent violation of a policy which must guarantee protection of an extensible system and its user. Web browsers which upload and run applets programs or a database that allows users to submit their own queries have to ensure that the behavior of the system is not dangerous. This goal can be reached by means of a *program monitor* which enforces the security policy.

We restrict ourselves to enforcement mechanisms which work at run time in parallel with the program under control.

[1] Email: beauquier@univ-paris12.fr
[2] Email: j.cohen@univ-paris12.fr
[3] Email: ruggero.lanotte@uninsubria.it

doi:10.1016/j.entcs.2009.06.037

Schneider[7,3] defined the first formal model of program monitor and studied what properties are enforceable with respect to this model. Ligatti and al. [4,5,2] propose a more general model based on *edit automata*. The monitor is not only able to interrupt a program execution in the case when it violates the security policy but it also can modify its behavior using suppression and insertion mechanisms.

Our goal is to characterize policies which can be enforced by edit automata having limited capabilities namely a finite memory. In [8] a family of edit automata named Bounded History Automata is introduced and policies enforceable by these automata are characterized, but the framework is different, the input alphabet and the set of states are not necessarily finite. Our results depend crucially on the finiteness of these two parameters.

The next section is devoted to definitions. Section 3 gives general properties of edit automata, in particular the fact that any recursive security policy is enforceable by an edit automaton. In section 4 we study the power of finite edit automata. The behavior of a finite edit automaton is analyzed as well as the structure of the enforced policy. The main result is given in section 5 : security policies enforceable by a finite edit automaton are exactly ∞-regular properties which are memory bounded. It is also proved that if the policy P is given by its finite automaton with n states, one can decide in $O(n^2)$ whether P is enforceable by a finite edit automaton and synthesize the controller in the positive case. We conclude in the last section with an example to illustrate the obtained results.

2 Basic notions

An execution σ is a finite sequence of actions $a_1 a_2 \dots a_n$. With $|\sigma|$ we denote the length n of σ. We use the notation $\mathcal{A}^*$ (resp. $\mathcal{A}^\omega$) to denote the set of all finite length (resp. infinite length) sequences of actions on a system with finite action set $\mathcal{A}$. Let $\mathcal{A}^\infty = \mathcal{A}^* \cup \mathcal{A}^\omega$. The symbol ϵ denotes the empty sequence. We use the notation $\sigma[i]$ to denote the i-th action in the sequence. The notation $\sigma[..i]$ denotes the prefix of σ of length i, and $\sigma[i+1..]$ denotes the corresponding suffix. When τ is a proper prefix of σ we write $\tau < \sigma$ and we write $\tau \leq \sigma$ to denote the fact that $\tau < \sigma$ or $\tau = \sigma$.

An *ultimately periodic sequence* is an infinite sequence of the form uv^ω, where $u, v \in \mathcal{A}^*, v \neq \epsilon$.

For $\sigma \in \mathcal{A}^\infty$ let us denote $\texttt{Pref}(\sigma)$ the set of prefixes of σ, and for a set $P \subset \mathcal{A}^\infty$, $\texttt{Pref}(P)$ denotes the set $\{u \in \mathcal{A}^\infty \mid u \in \texttt{Pref}(\sigma) \text{for some } \sigma \in P\}$. A *security policy* P is a subset of $\mathcal{A}^\infty$ such that $\epsilon \in P$.

We denote $P_{\texttt{fin}}$ the set $P \cap \mathcal{A}^*$ and $P_{\texttt{inf}}$ the set $P \cap \mathcal{A}^\omega$.

For $X \subset \mathcal{A}^*$, the *limit* of X denoted $\overrightarrow{X}$ is the set of infinite sequences which have infinitely many prefixes in X.

An *edit automaton* A is a deterministic finite or countably infinite state machine (Q, i, δ) that is defined with respect to some system with action set $\mathcal{A}$. Q is the set of automaton states, s is the initial state, and, $\delta : Q \times \mathcal{A} \to Q \times (\mathcal{A} \cup \{\epsilon\})$, is the transition function. We require that δ be Turing Machine computable. If

$\delta(q,a) = (q',b)$, and $b \neq \epsilon$, the transition is an insertion step: a is observed on the input but not consumed and b is produced on the output. We will write the transition $q \xrightarrow[b]{\cdot a} q'$.

If $\delta(q,a) = (q',\epsilon)$, the transition is a suppression step: a is read (consumed) on the input and nothing is produced on the output. We will write the transition $q \xrightarrow[\epsilon]{a\cdot} q'$.

A configuration is a pair $(\sigma, q) \in \mathcal{A}^\infty \times Q$.

We define a labeled relation $\mapsto$ on the set of configurations as follows:

$(a\sigma, q) \overset{b}{\longmapsto} (a\sigma, q')$ if $q \xrightarrow[b]{\cdot a} q'$ is an insertion step in A.

Observe that an insertion is not possible if the input is empty.

$(a\sigma, q) \overset{\epsilon}{\longmapsto} (\sigma, q')$ if $q \xrightarrow[\epsilon]{a\cdot} q'$ is a suppression step in A.

A *computation* of the edit automaton is a finite or infinite sequence

$(\sigma_0, q_0) \overset{b_1}{\longmapsto} (\sigma_1, q_1) \overset{b_2}{\longmapsto} \ldots (\sigma_n, q_n) \overset{b_{n+1}}{\longmapsto} (\sigma_{n+1}, q_{n+1}) \ldots$.

In such a computation, on the input σ_0 starting from state q_0, the edit automaton produces the output $b_1 b_2 \ldots b_n \ldots$ and the piece of the input which is read after n steps is the prefix σ' of σ_0 such that $\sigma_0 = \sigma' \sigma_n$.

The reflexive and transitive closure of $\mapsto$ is denoted $\mapsto_*$. For $\beta \in \mathcal{A}^\omega$ we write $(\alpha, q) \overset{\beta}{\longmapsto} {}_\omega$ if from configuration (α, q) there is no more suppression and the output is β. More precisely, for every n there is a computation $(\alpha, q) \overset{\beta[..n]}{\longmapsto} (\alpha, q_n)$.

We define the function $\mathcal{T}_A : \mathcal{A}^\infty \to \mathcal{A}^\infty$ as the function assigning to each execution σ its output τ from the initial state s.

More precisely $\mathcal{T}_A(\sigma) = \tau$ iff

D1 if $(\sigma, s) \overset{\tau'}{\longmapsto}_* (\sigma', q)$ then $\tau' \leq \tau$

D2 for every $\tau' \in \mathcal{A}^*$ prefix of τ there exists a state q and a sequence σ' such that $(\sigma, s) \overset{\tau'}{\longmapsto}_* (\sigma', q)$.

In [4], several types of enforcements are defined. As it is done in [5] we limit our study to effective enforcement. An effective enforcement preserves soundness and transparency. An enforcement mechanism of a policy P is *sound* when it ensures that outputs always obey P. Soundness is the main goal of an enforcement. It is *transparent* if it preserves the executions that already obey P. Notice that without transparency, any policy can be enforced in a trivial way just by outputing the empty sequence for example without reading the input. The formal definition is given below.

An edit automaton A *effectively enforces a policy* $P \subset \mathcal{A}^\infty$ if

E1 $\forall \sigma \in \mathcal{A}^\infty,\ \mathcal{T}_A(\sigma) \in P$ (soundness)

E2 $\forall \sigma \in P\ \ \mathcal{T}_A(\sigma) = \sigma$. (transparency)

In [5] a characterization of properties enforceable by edit automata is given:

Theorem 2.1 *There exists an edit automaton A that effectively enforces a security property P iff*

- $\forall \sigma \in \mathcal{A}^\omega$
  - $\sigma \in P_{\texttt{inf}}$ *iff* $\sigma \in \overrightarrow{P_{\texttt{fin}}}$ *or*
  - $\exists \sigma' < \sigma \ \forall \tau > \sigma' \ \tau \in P \implies \tau = \sigma$ *and the existence and actions of* σ *are computable from* σ'
- *the membership problem is decidable for* $P_{\texttt{fin}}$.

3 Properties of edit automata

By transitivity of $\mapsto_*$ we have the following proposition on the monotonicity of $\mathcal{T}$.

Proposition 3.1 *Let A be an edit automaton. If $\sigma \leq \tau$, then $\mathcal{T}_A(\sigma) \leq \mathcal{T}_A(\tau)$.*

The next Lemma describes the possible outputs of a finite input, and the possible inputs for a finite output.

Lemma 3.2 *Let A be an edit automaton that effectively enforces a policy P.*

(i) *If $\mathcal{T}_A(\sigma) = \tau$ and $\sigma \in \mathcal{A}^*$ then*
- *either $\tau \in \mathcal{A}^*$ and $(\sigma, s) \overset{\tau}{\longmapsto} (\epsilon, q)$ for some state q*
- *or τ is infinite and the computation from (σ, s) is infinite and from some step it contains only insertions.*

(ii) *If $\mathcal{T}_A(\sigma) = \tau$ and $\tau \in \mathcal{A}^*$ then*
- *either $\sigma \in \mathcal{A}^*$ and $(\sigma, s) \overset{\tau}{\longmapsto} (\epsilon, q)$ for some state q*
- *or σ is infinite and there exists a prefix σ_1 of $\sigma = \sigma_1\sigma_2$ such that $(\sigma_1, s) \overset{\tau}{\longmapsto} (\epsilon, q)$ and from (σ_2, q) the computation contains only suppressions.*

(iii) *if $\sigma \in P_{fin}$ then $(\sigma, s) \overset{\sigma}{\longmapsto} (\epsilon, q)$ for some state q and the last step of this computation is a suppression step.*

Proof. 1. Let $\sigma \in \mathcal{A}^*$ and $\mathcal{T}_A(\sigma) = \tau$. The number of suppression steps in the computation from (σ, s) is therefore finite and bounded by $|\sigma|$.

Suppose that $\tau \in \mathcal{A}^*$. Then the number of insertion steps in the computation from (σ, i) is also finite. Let us consider the last insertion step in the computation with input σ:

$(\sigma, s) \overset{\tau'}{\longmapsto} {}_*(a\sigma', q_1) \overset{b}{\longmapsto} (a\sigma', q_2)$ where $(a\sigma', q_1) \overset{b}{\longmapsto} (a\sigma', q_2)$ is the last insertion step.

From configuration $(a\sigma', q_2)$ only suppression steps occur. Moreover A is complete therefore there exists a state q such that $(a\sigma', q_2) \longmapsto {}_*(\epsilon, q)$. Concatening the two computations we get $(\sigma, s) \overset{\tau}{\longmapsto} (\epsilon, q)$.

Suppose now that $\tau \in \mathcal{A}^\omega$. Here the number of insertion steps in the computation from (σ, s) is infinite. Since there is no possible computation from any configuration (ϵ, q), the last suppression step is of the form $(a\sigma', q_1) \longmapsto (\sigma', q_2)$ for some non empty suffix σ' of σ and some states q_1, q_2. It follows that from configuration (σ', q_2) there are only insertion steps and infinitely many.

2. the second point is proved in a symmetric way.

3. if $\sigma \in P_{fin}$ then $\mathcal{T}_A(\sigma) = \sigma$ and from point 2. of the lemma we get

$(\sigma, s) \overset{\sigma}{\longmapsto} (\epsilon, q)$ for some state q. The last step of this computation cannot be an insertion step because the input is empty at the end of this step. □

Lemma 3.3 *If A is an edit automaton effectively enforcing P, then for any $\sigma \in \mathcal{A}^\infty$:*

$(\sigma' < \sigma \text{ and } \sigma' \in P) \implies \sigma' \leq \mathcal{T}_A(\sigma)$.

Proof. From Proposition 3.1, since $\sigma' \leq \sigma$, then $\mathcal{T}(\sigma') \leq \mathcal{T}_A(\sigma)$. Moreover since $\sigma' \in P$ we have $\mathcal{T}_A(\sigma') = \sigma'$. □

Given $\sigma \in \mathcal{A}^*$ and a policy P, with σ_P we denote the longest prefix of σ in P. This prefix exists since $\epsilon \in P$.

Given $\sigma \in P_{\texttt{fin}}$, with $\texttt{PreIm}(\sigma)$ we denote the set

$$\{\tau \in \mathcal{A}^* \mid \tau_P = \sigma,\ \tau \notin P \text{ and } \tau a \in P \text{ for some } a \in \mathcal{A}\}.$$

Intuitively the set $\texttt{PreIm}(\sigma)$ is the set of executions τ not in P having their longest prefix in P equal to σ and such that τa belongs to P for some a. This notion will be useful in the next section.

Lemma 3.4 *Let A be an edit automaton effectively enforcing P let $\sigma \in P$ and $\tau_1, \tau_2 \in \texttt{PreIm}(\sigma)$ such that $\tau_1 \neq \tau_2$. If τ is a prefix of τ_1 and τ_2 such that $\sigma < \tau$, then $\mathcal{T}_A(\tau) = \sigma$.*

Proof.

First of all we note that by definition of $\texttt{PreIm}(\sigma)$, there exist $a_1, a_2 \in \mathcal{A}$ such that $\tau_1 a_1$ and $\tau_2 a_2$ are in P. We note also that, since $\tau_1 \geq \tau > \sigma$ and $\tau_1 \in \texttt{PreIm}(\sigma)$, we have that, for any γ such that $\tau_1 \geq \gamma > \sigma$, it holds that $\gamma \notin P$. Hence $\tau_P = \sigma$.

Therefore by Lemma 3.3, $\sigma \leq \mathcal{T}_A(\tau)$. Hence either $\mathcal{T}_A(\tau) = \sigma$ or $\sigma < \mathcal{T}_A(\tau)$.

We prove by contradiction that $\mathcal{T}_A(\tau) = \sigma$.

Let us suppose that $\sigma < \mathcal{T}_A(\tau)$. Since $\tau_1 a_1 \in P$ we have that $\mathcal{T}_A(\tau_1 a_1) = \tau_1 a_1$. Since $\tau < \tau_1 a_1$, by monotonicity, $\sigma < \mathcal{T}_A(\tau) \leq \tau_1 a_1$.

By definition of $\mathcal{T}_A$, we have that $\mathcal{T}_A(\tau) \in P$, but we have noticed that, for any γ such that $\sigma < \gamma \leq \tau_1$, it holds that $\gamma \notin P$. Hence $\mathcal{T}_A(\tau) = \tau_1 a_1$.

Similarly we can prove that $\mathcal{T}_A(\tau) = \tau_2 a_2$. Hence $\tau_1 a_1 = \tau_2 a_2$ implying that $\tau_1 = \tau_2$ that is a contradiction by hypothesis.

□

4 Finite edit automata

A *finite* edit automaton is an edit automaton with a finite set of states. Our goal is to characterize properties enforceable by a finite edit automaton. We briefly recall some definitions about regular sets of finite or infinite sequences. For more details see [6].

4.1 Regular sets of finite or infinite sequences

A *deterministic finite automaton* on an alphabet $\mathcal{A}$ is a tuple $A = (Q, \mathcal{A}, s, F, \delta)$, where Q is a finite set of states, s is the initial state, F the set of terminal states, and $\delta : Q \times \mathcal{A} \to Q$ is a partial transition function. We write $q \stackrel{a}{\longmapsto} q'$ if $\delta(q, a) = q'$. A finite sequence $u \in \mathcal{A}^*$ is recognized (or accepted) by A if $s \stackrel{a}{\longmapsto} {}_*q'$ and $q' \in F$. The set of sequences recognized by A is denoted $L(A)$. The automaton A is *pruned* if every state q is reachable from s and q can reach at least one state of F.

A set $L \subset \mathcal{A}^*$ is *regular* if there exists a deterministic finite automaton A such that $L = L(A)$.

A *deterministic Muller automaton* on an alphabet $\mathcal{A}$ is a tuple $A = (Q, \mathcal{A}, s, \mathcal{F}, \delta)$, where Q is a finite set of states, s is the initial state, $\mathcal{F} \subset 2^Q$ the family of sets of infinitely repeated states, and $\delta : Q \times \mathcal{A} \to Q$ is a partial transition function. An infinite sequence $u \in \mathcal{A}^\omega$ is recognized by A if there is an infinite run of A with input u whose set of infinitely repeated set of states belongs to $\mathcal{F}$.

A set $L \subset \mathcal{A}^\omega$ is *ω-regular* if there exists a deterministic Muller automaton such that $L = L(A)$.

A set $F \subset Q$ is *alive* if there is at least one run from s whose set of infinitely repeated set of states is equal to F.

The automaton A is *pruned* if

- every set of $\mathcal{F}$ is alive

- every state q is reachable from s and q can reach at least one state of one set of $\mathcal{F}$.

A set $P \subset \mathcal{A}^\infty$ is *∞-regular* if $P_{\mathtt{fin}}$ is regular and $P_{\mathtt{inf}}$ is *ω-regular*.

Clearly a set $P \subset \mathcal{A}^\infty$ is *∞-regular* iff there exists a generalized Muller automaton $A = (Q, \mathcal{A}, s, F, \mathcal{F}, \delta)$ such that $(Q, \mathcal{A}, s, F, \delta)$ recognizes $P_{\mathtt{fin}}$ and $(Q, \mathcal{A}, s, \mathcal{F}, \delta)$ recognizes $P_{\mathtt{inf}}$.

The generalized Muller automaton A is pruned if

- every state q is reachable from s and q can reach at least one state of F or one set of $\mathcal{F}$

- every set of $\mathcal{F}$ is alive.

In the next two subsections we study the properties of $P_{\mathtt{fin}}$ and $P_{\mathtt{inf}}$ for a property P enforced by a finite edit automaton.

4.2 Properties of $P_{\mathtt{fin}}$

Lemma 4.1 *If there exists a finite edit automaton A that effectively enforces a security policy P then P_{fin} is regular.*

Proof. We give only a sketch of the proof. Let q be a state in A. We define L_q as the set of finite sequences $v \in \mathcal{A}^*$ such that there exists a finite sequence u and a computation $(u, s) \stackrel{v}{\longmapsto} {}_*(\epsilon, q)$ (*i.e.* $\mathcal{T}_A(u) = v$).

If A enforces the policy P, from Lemma 3.2(iii) we have $P_{\mathtt{fin}} = \cup_{q \in Q} L_q$.

Proving that L_q is regular will imply that $P_{\mathtt{fin}}$ is regular.

It is easy to construct a finite automaton A_q that accepts L_q.

□

Proposition 4.2 *If there exists a finite edit automaton A that effectively enforces a security policy P then for any $\sigma \in P_{\texttt{fin}}$, it holds that $\texttt{PreIm}(\sigma)$ is a finite set.*

Proof. By contradiction suppose that $\texttt{PreIm}(\sigma)$ is an infinite set for some $\sigma \in P$. Recall that all sequences of $\texttt{PreIm}(\sigma)$ have σ as prefix. Since the alphabet $\mathcal{A}$ is finite, there is an action a and c such that $\sigma\alpha \in \texttt{PreIm}(\sigma)$ and $\sigma\alpha a \in P$.

For each of these sequences α, using Lemma 3.2(iii), we have:

$(\sigma, s) \stackrel{\sigma}{\longmapsto} (\epsilon, q)$ and $(\sigma\alpha a, s) \stackrel{\sigma\alpha a}{\longmapsto} (\epsilon, q_\sigma)$ for some states q, q_σ, and the last steps of these two computations are a suppression step.

Thus $(\sigma\alpha a, s) \stackrel{\sigma}{\longmapsto} (\alpha a, q) \stackrel{\alpha a}{\longmapsto} (\epsilon, q_\sigma)$.

Let us analyse the second part of this computation. There is a step where the input contains only the last action a, it means that $\sigma\alpha$ has been suppressed from the input. From Lemma 3.4, since $\sigma\alpha \notin P$ the output is σ.

So we have:

$(\sigma\alpha a, s) \stackrel{\sigma}{\longmapsto} (\alpha a, q) \stackrel{\epsilon}{\longmapsto} (a, q'_\sigma) \stackrel{\alpha a}{\longmapsto} (a, q''_\sigma) \stackrel{\epsilon}{\longmapsto} (\epsilon, q_\sigma)$.

Because the set of states of A is finite there is three states q_1, q_2, q_3 and infinitely many sequences α such that

$(\alpha a, q) \stackrel{\epsilon}{\longmapsto} (a, q_1) \stackrel{\alpha a}{\longmapsto} (a, q_2) \stackrel{\epsilon}{\longmapsto} (\epsilon, q_3)$. $(*)$

Therefore, for fixed q, q_1, q_2, we have for infinitely many σ:

$(\alpha a, q) \stackrel{\epsilon}{\longmapsto} (a, q_1) \stackrel{\alpha a}{\longmapsto} (a, q_2)$. Since the alphabet $\mathcal{A}$ is finite, there is a sequence σ_0 of length greater than n, where n is the number of states of A satisfying $(*)$.

Thus there is a repeated state in the part $(a, q_1) \stackrel{\alpha_0 a}{\longmapsto} (a, q_2)$. But in this computation there are only insertions, so if a state is repeated in this part, since the input does not change and is equal to a, it implies that the controller which his deterministic will make insertions for ever and will never realize the suppression of a in the last part. Contradiction.

□

A finite automaton A is *simple* if every cycle in A contains at least one state of F.

Lemma 4.3 *If $P_{\texttt{fin}}$ is a regular set recognized by a deterministic pruned finite automaton A then*

the set $\texttt{PreIm}(\sigma)$ is finite for any $\sigma \in P_{\texttt{fin}}$ iff *A is simple.*

Proof. The "if" part is proved by contradiction. Suppose there is a cycle in A without final states. Then there is a path $s \stackrel{w_1}{\longrightarrow}_* q_1 \stackrel{w}{\longrightarrow}_* q \stackrel{v}{\longrightarrow}_* q \stackrel{w_2 a}{\longrightarrow}_* q_2$ where

- q_1, q_2 are final states,
- v is the label of the non-final cycle,
- there is no final state except q_1, q_2 on the path $q_1 \stackrel{wvw_2 a}{\longrightarrow}_* q_2$.

Then $w_1 w v^* w_2 \subset \texttt{PreIm}(w_1)$ that contradicts Proposition 4.2.

Conversely, let $u \in P_{\texttt{fin}}$ and $s \xrightarrow{u}_* q$ be the path labeled by u starting from s. There are finitely many reachable final states from final state q. A sequence τ in $\texttt{PreIm}(u)$ labels a path where q is the last final state of the path, because u is its longest prefix that belongs to $P_{\texttt{fin}}$. Moreover, this path can be extended up to a final state q' such as $s \xrightarrow{u}_* q \xrightarrow{\tau a}_* q'$. As there are no cycles between q and q', the number of such sequences is finite. □

4.3 Properties of $P_{\texttt{inf}}$

From Theorem 2.1, we know that if a security policy P is effectively enforceable by a finite edit automaton A then $\overrightarrow{P_{\texttt{fin}}} \subset P_{\texttt{inf}}$.

Let us study now the infinite sequences σ of $P_{\texttt{inf}}$ which are not in $\overrightarrow{P_{\texttt{fin}}}$. The next Lemma proves that the computation of such a sequence can be decomposed in three parts, the first part reads and outputs σ_P, in the second part the controller consumes a piece of the input and outputs nothing, in the last part, the input no longer changes and the controller makes infinitely many insertions.

Lemma 4.4 *If a finite edit automaton effectively enforces a security policy P then every $\sigma \in P_{inf} \setminus \overrightarrow{P_{\texttt{fin}}}$ can be written in a unique way $\sigma_1\alpha\beta$ such that:*

- *σ_1 is the longest finite prefix of σ in $P_{\texttt{fin}}$*
- *$(\sigma_1\alpha\beta, s) \overset{\sigma_1}{\longmapsto}_* (\alpha\beta, q) \overset{\epsilon}{\longmapsto}_* (\beta, q')$ for some q, q'*
- *$(\beta, q') \overset{\alpha\beta}{\longmapsto}_{\omega}$ and β is ultimately periodic*
- *$\sigma_1\alpha \notin Pref(P_{\texttt{fin}})$.*

Proof. Let $P' = P_{inf} \setminus \overrightarrow{P_{\texttt{fin}}}$. Consider $\sigma \in P'$. Let σ_1 be the longest finite prefix of σ such that $\sigma_1 \in P_{\texttt{fin}}$. The sequence σ_1 exists since $\sigma \notin \overrightarrow{P_{\texttt{fin}}}$ and $\epsilon \in P$. Then using Lemma 3.2 there is a computation $(\sigma_1, s) \overset{\sigma_1}{\longmapsto}_* (\epsilon, q)$ where the last computation step is a suppression one.

Let $\sigma = \sigma_1\sigma'$. So one has $(\sigma_1\sigma', s) \overset{\sigma_1}{\longmapsto}_* (\sigma', q)$ **(1)**.

Since $\sigma \in P$, for every $\beta \in \mathcal{A}^+$ such that $\sigma_1\beta < \sigma$ we have a computation $(\sigma, s) \overset{\sigma_1\beta}{\longmapsto}_* (\sigma'', q'')$ **(2)** for some σ'', q''.

Now, using the determinism of the controller, computation **(1)** is a prefix of computation **(2)** Then from (σ', q), there must be an insertion step in order to output β.

Let then $\sigma' = \alpha\sigma''$ where α is the longest finite prefix of σ' on which A produces only suppressions (α can be the empty sequence). We have the computation $(\sigma', q) \overset{\epsilon}{\longmapsto}_* (\sigma'', q') \overset{b}{\longmapsto} (\sigma'', q_1)$ where $(\sigma'', q') \overset{b}{\longmapsto} (\sigma'', q_1)$ is the first insertion step after (σ', q).

Suppose now that there is a suppression step after this insertion step and again consider the first one : we have $\sigma' = \alpha\sigma''$ and states q_1, q_2 such that there is a computation $(\sigma, s) \overset{\sigma_1}{\longmapsto}_* (\sigma', q) \overset{\epsilon}{\longmapsto}_* (\sigma'', q') \overset{\alpha'}{\longmapsto}_* (\sigma'', q_1)$ and the next step is a suppression one. It follows that $\mathcal{T}_A(\sigma_1\alpha) = \sigma_1\alpha'$. Hence we have $\sigma_1\alpha' \in P_{fin}$ and $\sigma_1 < \sigma_1\alpha' \leq \sigma$ whereas σ_1 is the longest finite prefix of σ that belongs to P :

contradiction.

Therefore in the computation from (σ', q), once there is an insertion step, there are always insertion steps. Let q be the state from which only insertions steps occur in computation from (σ', q). Let then $\sigma = \sigma_1\alpha\beta$ with computation $(\sigma, s) \stackrel{\sigma_1}{\longmapsto}_* (\alpha\beta, q) \stackrel{\epsilon}{\longmapsto}_* (\beta, q')$. Moreover since there is no more suppression step, the computation of length n from (β, q') produces as output the prefix of length n of $\alpha\beta$: $(\beta, q') \stackrel{(\alpha\beta)[..n]}{\longmapsto} (\beta, q_n)$ for some state q_n. But A has a finite set of states, and the input does not change from (β, q') thus there exists integers $n_1 < n_2$ such that $q_{n_1} = q_{n_2}$ and from q_{n_1} the output is periodic, so from (β, q') the output is ultimately periodic.

We have proved that any infinite sequence in P' is of the form $\sigma_1\alpha\beta$ where $\sigma_1 \in P_{\texttt{fin}}$, α as input corresponds to a sequence of suppression steps and β is an ultimately periodic sequence.

Remark that for such a sequence σ in P', the set $\texttt{Pref}(\sigma) \cap \texttt{Pref}(P_{\texttt{fin}})$ is finite. Indeed if it was not the case there would exist a prefix γ of σ and $\gamma' \in \mathcal{A}^*$ such that $\gamma = \sigma_1\alpha a\beta'$ with $\beta' < \beta$ and $\gamma\gamma' \in P_{\texttt{fin}}$. Then we should have $\mathcal{T}(\gamma\gamma') = \gamma\gamma'$. But on the other hand $\mathcal{T}(\gamma\gamma') = \mathcal{T}(\sigma_1\alpha a\beta'\gamma') = \sigma$. A contradiction. We deduce that there is a longest prefix of σ in $\texttt{Pref}(P_{\texttt{fin}})$ and this prefix is of the form $\sigma_1\alpha'$ where $\alpha' \leq \alpha$. □

We are now in position to give an ω-regular expression of $P_{\texttt{inf}} \setminus \overrightarrow{P_{\texttt{fin}}}$

Proposition 4.5 *If a finite edit automaton effectively enforces a security policy P then $P_{inf} \setminus \overrightarrow{P_{\texttt{fin}}}$ is of the form $\cup_{j\in J} R_j\beta_j$ where*

- *J is finite*
- *for every $j \in J, R_j$ is regular*
- *for every $j \in J, \beta_j$ is an ultimately periodic sequence in $\mathcal{A}^\omega$*
- *for every $j \in J, R_j \cap \texttt{Pref}(P_{\texttt{fin}}) = \emptyset$*
- *$R_j \cap \texttt{Pref}(R_i) = \emptyset$ for $i \neq j$*
- *for every $u < v$ with $u \notin \texttt{Pref}(P_{\texttt{fin}})$ and $v \in R_j$ we have $|v| - |u| \leq K$ where K is the number of states of A.*

As a consequence $P_{\texttt{inf}}$ is ω-regular.

Proof. Let $P' = P_{\texttt{inf}} \setminus \overrightarrow{P_{\texttt{fin}}}$, then σ has the form $\sigma_1\alpha_\sigma\beta_\sigma$ satisfying properties of Lemma 4.4. Remark that the set $\texttt{Pref}(\sigma) \cap \texttt{Pref}(P_{\texttt{fin}})$ is finite. Indeed if it is not finite then there exists a prefix γ of σ and $\gamma' \in \mathcal{A}^*$ such that $\gamma = \sigma_1\alpha_\sigma\beta'$ with $\beta' < \beta_\sigma$ and $\gamma\gamma' \in P_{\texttt{fin}}$. Then we should have the following : $\mathcal{T}_A(\gamma\gamma') = \gamma\gamma'$. But we have using Lemma 4.4, $\mathcal{T}_A(\gamma\gamma') = \mathcal{T}_A(\sigma_1\alpha_\sigma\beta'\gamma') = \sigma$. Contradiction.

We deduce that there is a longest prefix of σ in $\texttt{Pref}(P_{\texttt{fin}})$ and this prefix is of the form $\sigma_1\alpha'$ where $\alpha' \leq \alpha_\sigma$.

Now we focus on the set E of finite sequences α_σ for all σ in P' and we prove that E is finite.

Recall that each $\sigma \in P'$ is written in a unique way $\sigma_1\alpha_\sigma\beta_\sigma$ such that the com-

putation starting in $(\sigma_1\alpha_\sigma\beta_\sigma, s)$ is as follows:

$(\sigma_1\alpha_\sigma\beta_\sigma, s) \stackrel{\sigma_1}{\longmapsto}_* (\alpha_\sigma\beta_\sigma, q_\sigma) \stackrel{\epsilon}{\longmapsto}_* (\beta_\sigma, q'_\sigma) \stackrel{\alpha_\sigma\beta_\sigma}{\longmapsto} {}_\omega$.

If E is infinite, there exists an infinite sequence α such that every prefix $\alpha[..p]$ of α is also a prefix of some α_{σ_p} in E for some $\sigma_p \in P'$. Since the set of states of A and the alphabet $\mathcal{A}$ are finite, there exist an action a and states q and q' such that for all σ_p we have $q_{\sigma_p} = q, q'_{\sigma_p} = q'$ and a is the first letter of β_{σ_p}.

Thus for all σ_p we have:

$(\alpha_\sigma\beta_\sigma, q) \stackrel{\epsilon}{\longmapsto}_* (\beta_\sigma, q') \stackrel{\alpha_\sigma\beta_\sigma}{\longmapsto} {}_\omega$.

But the computation $(\beta_{\sigma_p}, q') \stackrel{\alpha_{\sigma_p}\beta_{\sigma_p}}{\longmapsto} {}_\omega$ depends only on q' and the first letter of β_{σ_p}. Thus all the outputs $\alpha_{\sigma_p}\beta_{\sigma_p}$ are equal and so necessarily equal to α. The sequence α is ultimately periodic $\alpha = \alpha_1\alpha_2^\omega$. There exists p large enough such that $\alpha_p = \alpha_1\alpha_2^K\alpha'_p$, where K is the number of states of A. Then the computation $(\alpha_{\sigma_p}\beta_{\sigma_p}, q) \stackrel{\epsilon}{\longmapsto}_* (\beta_{\sigma_p}, q')$ has some repeated state in the following way:

$(\alpha_1\alpha_2^K\alpha'_p, q) \stackrel{\epsilon}{\longmapsto}_* (\alpha_2^{K_1}\alpha'_p\beta_{\sigma_p}, q_1) \stackrel{\epsilon}{\longmapsto}_* (\alpha_2^{K_2}\alpha'_p\beta_{\sigma_p}, q_1) \stackrel{\epsilon}{\longmapsto}_* (\beta_{\sigma_p}, q')$.

On the other hand $\alpha'_p\beta_{\sigma_p} = \alpha_2^\omega$. So from configuration $(\alpha'_p\beta_{\sigma_p}, q_1)$ the computation makes only suppressions for ever. It contradicts the fact that $(\beta_\sigma, q') \stackrel{\alpha_\sigma\beta_\sigma}{\longmapsto} {}_\omega$.

Since $\sigma \in P'$, we deduce that $\sigma_1\alpha_1\alpha'^k\alpha_2 a\beta \in P'$ for any positive integer k.

Then we must have for any positive integer k:

- $\mathcal{T}_A(\sigma_1\alpha a\beta) = \sigma$
- $\mathcal{T}_A(\sigma_1\alpha_1\alpha'^k\alpha_2 a\beta) = \sigma_1\alpha_1\alpha'^k\alpha_2 a\beta$

Besides that we have $\mathcal{T}_A(\sigma_1\alpha_1\alpha'^k\alpha_2 a\beta) = \sigma$.

Hence for any positive integer k we have $\sigma_1\alpha_1\alpha'\alpha_2 a\beta = \sigma_1\alpha_1\alpha'^k\alpha_2 a\beta$.

We deduce that $\sigma = \sigma_1\alpha_1\alpha'^\omega$. In that case $\mathcal{T}_A(\sigma) = \sigma_1$. Contradiction.

We have proved that E is finite and the number of sequences in E is bounded by $K^2|\mathcal{A}|$.

Now we make precise the set P'. From Lemma 4 $P_{\texttt{fin}}$ is regular. Let $L'_q = \{u \mid (u, s) \stackrel{v}{\longmapsto} (\epsilon, q) \text{ for some } v \in \mathcal{A}^*\}$. From A it is easy to construct a finite automaton which recognizes L'_q. Consider $\bar{L}'_q = L'_q \cap P_{\texttt{fin}}$ Clearly, for each $u \in \bar{L}'_q$ we have $(u, s) \stackrel{u}{\longmapsto} (\epsilon, q)$. And $\bar{L}'_q$ is a regular set.

Let $a \in \mathcal{A}$. We define the set of states $Q_a = \{q \in Q | \exists q' \in Q\ \delta(q, a) = (q', b)\}$.

For $q \in Q_a$ we can notice that there is exactly one infinite computation from $(a\gamma, q)$ for any finite or infinite sequence γ; this computation performs only insertion steps and the output for this computation is ultimately periodic as proved in Lemma 7. Let us denote $\beta_{q,a}$ this ultimately periodic sequence.

Let q be a state of A and q' be a state of Q_a for a non empty Q_a. Let $F_{q,q',a}$ be the set of sequences α in $\mathcal{A}^*$ whose length is less than K such that there is a computation $(\alpha, q) \stackrel{\epsilon}{\longmapsto}_* (\epsilon, q')$ and such that $\alpha a < \beta_{q',a}$.

The set $F_{q,q',a}$ has at most one sequence. Indeed, if α and α' are two different sequences in $F_{q,q',a}$, since α and α' are prefixes of $\beta_{q',a}$ one has $\alpha < \alpha'$ (or the converse). It follows that $\alpha' = \alpha a u$ for some u.

Besides we have also the computations

$(\alpha, q) \stackrel{\epsilon}{\longmapsto}_* (\epsilon, q')$ and $(\alpha', q) \stackrel{\epsilon}{\longmapsto}_* (\epsilon, q')$. From the first one we deduce the computation $(\alpha a u, q) \stackrel{\epsilon}{\longmapsto}_* (au, q')$. From the second one we deduce the computation $(\alpha a u, q) \stackrel{\epsilon}{\longmapsto}_* (\epsilon, q')$. But from (au, q') there are only insertions steps that is in contradiction with this last computation. Let us denote $\alpha_{q,q',a}$ the unique sequence in $F_{q,q',a}$ when $F_{q,q',a}$ is not empty. And we can set :

$\beta_{q',a} = \alpha_{q,q',a} a \beta'_{q',a}$ for q, q', a such that $Q_a \neq \emptyset$ and $F_{q,q',a} \neq \emptyset$. Moreover one can prove that $|\alpha_{q,q',a}| < K^2$, otherwise in the last part of the computation when there are only insertion steps, the repetitive part would begin inside the production of $\alpha_{q,q',a}$ and $\alpha_{q,q',a}$ would contain at least K times the period. But in that case the computation starting from $(\alpha_{q,q',a} a \beta'_{q',a}, q)$ would be made of suppressions for ever. A contradiction.

We have then

- $|\alpha_{q,q',a}| < K^2$
- $\beta_{q',a}$ is ultimately periodic.

We define the sets $R_{q,q',a} = \bar{L}_q \alpha_{q,q',a} a$.

We have proved that $P' \subset \cup_{q,a,q'} R_{q,q',a} \beta_{q',a}$. By construction we have clearly also $\cup_{q,q',a} R_{q,q',a} \beta_{q',a} \subset P_{\mathtt{inf}}$. Moreover an infinite sequence in some $R_{q,q',a}\beta_{q',a}$ cannot belong to $\overrightarrow{P_{\mathtt{fin}}}$ since $R_{q,q',a} \cap \mathtt{Pref}(P_{\mathtt{fin}}) = \emptyset$.

Hence $P' = \cup_{q,q',a} R_{q,q',a} \beta_{q',a}$.

We have to prove that sets $R_{q,q',a}$ satisfy

- $R_{q,q',a} \cap \mathtt{Pref}(P_{\mathtt{fin}}) = \emptyset$,
- they are mutually disjoints,
- they are regular
- $R_{q,q',a} \cap \mathtt{Pref}(R_{q_1,q'_1,b}) = \emptyset$ for any $(q, q', a) \neq (q_1, q'_1, b)$.

• The first property follows from Lemma 7.

• Let $u \in R_{q,q',a} \cap R_{q_1,q'_1,b}$. Then we have $u = v\alpha_a a = w\alpha_b b$ with $v, w \in P_{\mathtt{fin}}$ such that there are computations

$(v, s) \stackrel{v}{\longmapsto}_* (\epsilon, q)$ with a suppression last step

$(w, s) \stackrel{w}{\longmapsto}_* (\epsilon, q_1)$ with a suppression last step

$(\alpha_a a, q) \stackrel{\epsilon}{\longmapsto}_* (a, q')$

$(\alpha_b b, q_1) \stackrel{\epsilon}{\longmapsto}_* (b, q'_1)$.

Firstly $a = b$ clearly holds.

Suppose now $v < w$. Then we have $w = v\alpha'$ where $\alpha' \leq \alpha_a$ and there is a computation $(w, q_0) \stackrel{v}{\longmapsto}_* (\alpha', q) \stackrel{\epsilon}{\longmapsto}_* (\epsilon, q_2)$ for some q_2. Hence $v = w$ and moreover $\alpha_a = \alpha_b$.

• $R_{q,q',a}$ has been proved to be regular.

• Let $u \in R_{q,q',a} \cap \mathtt{Pref}(R_{q_1,q'_1,b})$. Then on one hand we have $u = v\alpha_a a =$ with $v \in P_{\mathtt{fin}}$ with a computation $(u, q_0) \stackrel{v}{\longmapsto}_* (a, q')$ with a suppression last step. On the other hand we have a sequence u' such that $uu' = v\alpha_a a u' \in R_{q_1,q'_1,b}$. Thus

there must be the computation $(uu', q_0) \stackrel{w}{\longmapsto}_* (b, q_1')$. Hence we have $(uu', q_0) \rightarrow_* (au', q') \longmapsto_* (b, q_1')$. But from (au', q') there are only insertion steps. It follows that $u' = \epsilon$ and $u \in R_{q_1, q_1', b}$. As aforementioned we deduce $(q, q', a) = (q_1, q_1', b)$. □

Definition 4.6 A policy P is memory bounded if P is of the form $P = P_{\mathtt{fin}} \cup \overrightarrow{P_{\mathtt{fin}}} \cup_{j \in J} R_j \beta_j$ where

- $\epsilon \in P$
- J is finite
- $P_{\mathtt{fin}} \subset \mathcal{A}^*$ is recognized by a simple finite automaton
- for every $j \in J, R_j$ is regular
- for every $j \in J, \beta_j$ is an ultimately periodic sequence in $\mathcal{A}^\omega$
- for every $j \in J, R_j \cap Pref(P_{\mathtt{fin}}) = \emptyset$
- $R_j \cap \mathtt{Pref}(R_i) = \emptyset$ for $i \neq j$
- there exists a constant K such that for every $u < v$ with $u \notin \mathtt{Pref}(P_{\mathtt{fin}})$ and $v \in R_j$ $|v| - |u| \leq K$.

From Proposition 4.5 we obtain:

Theorem 4.7 *If a security policy P is enforced by a finite edit automaton, then P is memory bounded.*

We intend now to characterize the generalized Muller automata which recognize properties that are memory bounded.

Definition 4.8 A pruned generalized Muller automaton A is simple if:

S0. $s \in F$

S1. each cycle encounters F or a set of $\mathcal{F}$

S2. the restriction of the automaton A to each set F_i in $\mathcal{F}$ which has no state in F is an elementary cycle C_i

S3. there is no edge from a state in C_i to a state not in C_i for every i

S4. each alive set G such that $G \cap F \neq \emptyset$ belongs to $\mathcal{F}$.

Proposition 4.9 *Given a pruned generalized Muller automaton A, $L(A)$ is memory bounded iff A is simple.*

Proof. • Let us suppose that A is simple. Let P be the property recognized by A. Because of $S0$, $\epsilon \in P$. Because of condition $S4$ we have $\overrightarrow{P_{\mathtt{fin}}} \subset P_{\mathtt{inf}}$. Let $P' = P \setminus (P_{\mathtt{fin}} \cup \overrightarrow{P_{\mathtt{fin}}})$. Let $\mathcal{F}'$ be the family of sets F_j in $\mathcal{F}$ disjoint from F. Let us remark that from properties $S2$ and $S3$ one can deduce that for distinct F_i and F_j in $\mathcal{F}'$, cycles C_i and C_j are disjoint. Let us call this property, property $S5$. For each G in $\mathcal{F}'$, and each $q \in G$ let R_q be the set of finite sequences recognized by A in a computation which starts in the initial state and stops in state q without running through G before the last state q. Let β_q be the periodic infinite sequence

recognized by A in a computation which starts in q. Because of properties $S2$ and $S3$, β_q is unique.

Clearly $P' = \bigcup_{G \in \mathcal{F}', q \in G} R_q \beta_q$ where $\mathcal{F}'$ is the set of F_i in $\mathcal{F}$ disjoint of F.

By construction, $R_q \cap P_{\mathtt{fin}} = \emptyset$, and from property $S3$, $R_q \cap \mathtt{Pref}(P_{\mathtt{fin}}) = \emptyset$. Because of properties $S3$ and $S5$, $R_q \cap \mathtt{Pref}(R_{q'}) = \emptyset$. At last, for two different q and q', there is no common sequence to $R_q\beta_q$ and $R_{q'}\beta_{q'}$ because either q and q' are not in the same G of $\mathcal{F}'$ and the computation of σ cannot have two distinct infinitely repeated sets, or q and q' are in the same G of $\mathcal{F}'$ and the first state of G reached in the computation of σ is unique.

So if A satisfies properties $S0 - S4$, then P' has the required form.

• Conversely, if property $S0$ is not satisfied then $\epsilon \notin P$. If $S4$ is not satisfied, then $\overrightarrow{P_{\mathtt{fin}}} \not\subset P_{inf}$. If property $S2$ is not satisfied then P' contains infinite sequences which are not ultimately periodic. Thus P' cannot have the required form.

Let us suppose that A satisfies properties $S0, S2, S4$ but not property $S3$. There exists a cycle C_i in $\mathcal{F}'$ which has an outgoing edge. In that case, since the automaton is pruned, this edge can be extended either in a path which reaches F, or in a path which reaches another cycle C_j. Because of Proposition 2 and Lemma 5 C_i cannot reach F. So it reaches another cycle C_j.

For any constant $K > 0$ one can build an infinite path with a piece p larger than K and larger than the number of states of A inside C_i and reaching after that C_j for ever. Let σ the infinite sequence labeling this infinite path. Invoking the determinism of the automaton, the piece p cannot contain a period of the ultimately periodic part of σ. So if P' has the required form, σ belongs to some $R_i\beta_i$, but the the piece p corresponds to prefixes of σ which are not in $\mathtt{Pref}(P_{\mathtt{fin}})$ and the length of p is larger than K. A contradiction. We have proved that if A satisfies property $S2$ but not property $S3$ then P' cannot have the required form.

Suppose now that A satisfies properties $S0$, $S2$ $S3$ and $S4$ but not property $S1$. It means that there exists a cycle C from which one can reach either F, or some C_i. Then along the same lines as in the previous case P' cannot have the required form. □

Proposition 4.10 *Given a pruned generalized Muller automaton A with n states, one can decide in time $O(n^2)$ whether $L(A)$ is memory bounded.*

Proof. Here is the algorithm :

- Check that $s \in F$
- Compute the set $\mathcal{C}$ of terminal strongly connected components of A
- Compute the set $\mathcal{C}'$ of terminal strongly connected components which do not intersect F
- Check whether there is only one infinite path from one state of each component in $\mathcal{C}'$
- Compute the set of states D not in F that can reach F.
- Check that the set of paths that reach F from each state in D is finite.

□

4.4 Characterization of policies effectively enforced by a finite edit automaton

In this subsection we give the reverse part of Theorem 4.7.

Theorem 4.11 *Given a pruned generalized Muller automaton A recognizing a security policy P which is memory bounded, one can build a finite edit automaton which effectively enforces P.*

Proof. From Proposition 4.9 the automaton A is simple. We now build the edit automaton which enforces P. Let us first describe informally the behavior of this controller. The states of the automaton A can be divided into three parts: F, I the set of *internal* states not in F but that can reach F, and the set O for the other ones. Let an input $u_1u_2...u_k$ in $P_{\mathtt{fin}}$ which has $k+1$ prefixes in $P_{\mathtt{fin}}$, namely $\epsilon, u_1, u_1u_2, ...u_1u_2...u_k$. The controller reads u_1 except its last letter and memorizes it, then observes the last letter of u_1 and writes u_1, and finally reads the last letter of u_1, the controller processes in the same way for $u_2, ...u_k$. The reason why the controller does not read immediately the last letter of u_1 is that after this reading, the output must be u_1, so the controller must write entirely u_1 before the end of the reading of u_1. Sequences in $P_{\mathtt{fin}} \cup \overrightarrow{P_{\mathtt{fin}}}$ are processed in this way. For infinite sequences in $P \setminus (P_{\mathtt{fin}} \cup \overrightarrow{P_{\mathtt{fin}}})$, as long as the prefix is in $\mathtt{Pref}(P_{\mathtt{fin}})$ the treatment is as before, but when the input is no longer in $\mathtt{Pref}(P_{\mathtt{fin}})$, which happens when the automaton A enters a state in O then the controller reads the input and memorizes it until a final cycle is reached. When a final cycle is reached the controller stops the reading and writes the memorized factor followed by the periodic final part. For any sequence σ not in P, the behavior of the controller at the beginning is the same, but as soon as the automaton A cannot read a letter the controller stops the writing and reads the input up to the end. So the output is the longest prefix of σ that is in $P_{\mathtt{fin}}$. We give now the detailed transitions of the controller. Let $A = (Q, \mathcal{A}, s, F, \delta)$. The set of states Q can be divided into three parts: F the set of final states, I the set of states not in F but that can reach F, and O the set of other states. We divide O into O_i the set of states in O which do not belong to a set in $\mathcal{F}$ and O_f the complement. Due to the fact that A is simple, each state q of O_f defines a unique sequence $a_q u_q$ which is the label of the elementary cycle starting in q.

The set of states of the edit automaton is: $\mathcal{Q} = Q \cup \hat{Q} \times A \cup (Q \cup \bar{Q}) \times A^{\leq n} \cup \{t\}$.

Transitions are :

- $q \xrightarrow{a\cdot|\epsilon} q'_a$ if $q \in F$ and $\delta(q,a) = q' \notin F$

$q \xrightarrow{\cdot a|a} \hat{q}'_a$ if $q \in F$ and $\delta(q,a) = q' \in F$

$\hat{q}_a \xrightarrow{a\cdot|\epsilon} q$ if $q \in F$

$q_u \xrightarrow{a\cdot|\epsilon} q'_{ua}$ if $\delta(q,a) = q'$ and $q' \notin F \cup O_f$

The controller memorizes the factor it will write later if the input is admissible.

- $q_{bu} \xrightarrow{\cdot a|b} \bar{q}'_{ua}$ if $\delta(q,a) = q'$ and $q' \in F \cup O_f$

$\bar{q}_{bua} \xrightarrow{\cdot a|b} \bar{q}_{ua}$ if $q \in F \cup O_f$

$\bar{q}_a \xrightarrow{\cdot a|a} \hat{q}_a$ if $q \in F \cup O_f$

$\hat{q}_a \xrightarrow{\cdot a|b} \hat{q}'_a$ if $q \in O_f$ and $\delta(q,b) = q' \in O_f$

The controller writes the factor it has memorized.

$\hat{q}_a \xrightarrow{a\cdot|\epsilon} q$ if $q \in F$ The controller ends the reading of a sequence in $P_{\texttt{fin}}$.

$\hat{q}_a \xrightarrow{\cdot a|b} \hat{q}'_a$ if $q \in O_f$ and $\delta(q,b) = q' \in O_f$

The controller writes an infinite periodic sequence and no longer reads any letter.

• $q_u \xrightarrow{a\cdot|\epsilon} t$ and $q \xrightarrow{a\cdot|\epsilon} t$ if there is no transition $\delta(q,a)$

$t \xrightarrow{a\cdot|\epsilon} t$ for every $a \in \mathcal{A}$.

The input is not in P, its longest prefix in P has been written, the end of the input is read without writing anything.

□

From Theorems 4.7 and 4.11 we get:

Theorem 4.12 *A security policy is effectively enforceable by a finite edit automaton iff it is memory bounded.*

and from Proposition 4.10:

Theorem 4.13 *Given a pruned generalized Muller automaton A with n states, one can decide in $O(n^2)$ whether $L(A)$ is effectively enforceable by a finite edit automaton.*

5 An example

In the following example the set of actions is $\mathcal{A} = \{0, 1, a, \sharp\}$ where

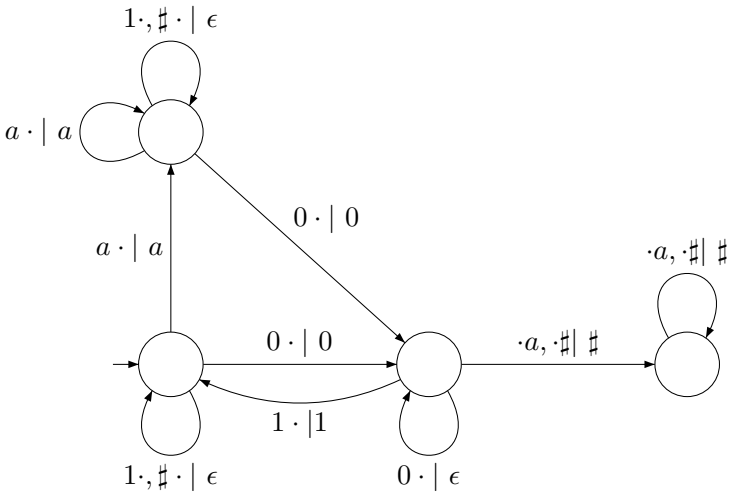
- 0 is an action for opening a session
- 1 is an action for closing a session
- a is an action that is allowed to be done only outside a session
- $\sharp$ is an interruption that can be used to end processing while a user intends to have a forbidden behavior.

The policy we consider here is :
$P = \texttt{Pref}(X) \cup \overrightarrow{X} \cup X0\sharp^\omega$ where $X = ((01)^* \cup a)^*$.

The "normal" behavior is represented by $\texttt{Pref}(X) \cup \overrightarrow{X}$. When an attempt to execute action a inside a session, namely just after an opening action 0 the process is interrupted by an infinite sequence of $\sharp$ actions that corresponds to $X0\sharp^\omega$.

An edit automaton that enforces P is shown below. It interrupts any attempt of running action a when a session is opened but not closed. Any irrelevant opening or closing action is suppressed as well as irrelevant interruptions. An insertion step $q \xrightarrow{\cdot a|a} q'$ followed by a suppression step $q' \xrightarrow{a\cdot|\epsilon} q$" is compressed in one single

transition $q \xrightarrow{a\cdot|a} q$".



Discussion

In this paper we have characterized the policies that can be enforced by finite edit automata. These policies are a subclass of ∞-regular policies. Moreover, we provide an algorithm which constructs the program monitor from an automaton that recognizes the policy.

Finite transducers [1] are a classical notion very close to edit automata. A finite transducer on an alphabet A is defined by its finite set of states Q, an initial state s and a set of transitions $\delta \subset Q \times A \times \{\epsilon\} \times Q \cup Q \times \{\epsilon\} \times A \times Q$. Transitions (q, a, ϵ, q') (*read transitions*) represent a suppression of a on the input and nothing is written on the output, transitions (q, ϵ, b, q') (*write transitions*) correspond to an input unchanged and a b is written on the output. Determinism implies there are no two different read transitions from the same state and the same read action, no two different write transitions from the same state and if there is a read action form a state there is no write action from the same state. A finite transducer is complete if from every state where a write transition is impossible there is a read transition for every letter in input. Thus the difference between deterministic complete finite transducers (dcft) and edit automata is very minimal, the write transitions in dcft do not depend on a future input (the input can empty) but only on the current state contrary to edit automata. One can prove and it is not surprising that finite edit automata and dcft enforce the same class of policies.

Our future work will be done in several directions. A natural question is whether one can decide if a given edit automaton enforces the set of sequences of its output. We will solve this question positively at least in the case when the edit automaton is finite. Secondly, we plane to explore the power of pushdown edit automata. At last, we intend to distinguish actions of different types. Actually in practice there are some limitations about the power of the controller. Some actions are unsuppressible by the controller or uninsertable. So it may be of interest to consider a specification

that takes into account this feature.

References

[1] J. Berstel. *Transductions and Context-free Languages*. Teubner Verlag, 1979.

[2] L. Bauer, J. Ligatti, and D. Walker. More enforceable security policies, 2002.

[3] Kevin W. Hamlen, Greg Morrisett, and Fred B. Schneider. Computability classes for enforcement mechanisms. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 28(1):175 – 205, 2006.

[4] J. Ligatti, L. Bauer, and D. Walker. Edit automata: Enforcement mechanisms for run-time security policies. *International Journal of Information Security*, 4:2–16, 2005.

[5] J. Ligatti, L. Bauer, and D. Walker. Enforcing non-safety security policies with program monitors. In *Computer Security - ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Sciences*, pages 353–373, 2005.

[6] D. Perrin and J.E. Pin. *Infinite Words Automata, Semigroups, Logic and Games*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.

[7] Fred B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.

[8] C. Talhia, N. Tawbia, and M. Debbabib. Execution monitoring enforcement under memory-limitation constraints. *Information and Computation*, 2008.