Path-equivalent developments in acyclic weighted automata

Mathieu Giraud, Philippe Veber, Dominique Lavenier

International Journal of Foundations of Computer Science

PATH-EQUIVALENT DEVELOPMENTS IN ACYCLIC WEIGHTED AUTOMATA

MATHIEU GIRAUD*

Bioinfo/Sequoia, LIFL, CNRS, INRIA Futurs, Université Lille 1
59650 Villeneuve d'Ascq, France

and

PHILLIPE VEBER

Symbiose, IRISA, INRIA, Campus de Beaulieu
35042 Rennes cedex, France

and

DOMINIQUE LAVENIER

Symbiose, IRISA, CNRS, Campus de Beaulieu
35042 Rennes cedex, France

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

Weighted finite automata (WFA) are used with FPGA accelerating hardware to scan large genomic banks. Hardwiring such automata raises surface area and clock frequency constraints, requiring efficient ε-transitions-removal techniques. In this paper, we present bounds on the number of new transitions for the *development* of acyclic WFA, which is a special case of the ε-transitions-removal problem. We introduce a new problem, a partial removal of ε-transitions while accepting short chains of ε-transitions.

Keywords: Removal of ε-transitions, partial removal, weighted finite automaton, hardware acceleration, chains of ε-transitions

1. Introduction

Weighted Finite Automata (WFA) are used to find occurrences of biological patterns in genomic databases containing tens of gigabytes of data. Biological patterns can be seen as regular or weighted expressions over the 20-letter amino acid alphabet. They may represent the signature of a protein family, the features

*giraud@lifl.fr

of a domain or the specific location of an active site. The usual length ranges of the patterns are from a few amino acids to a few tens.

Today, with the exponential growth of genomic data, a pure software implementation of WFA is inefficient when dealing with large databanks. WFA can be efficiently hardwired onto reconfigurable architectures (FPGA components) to speed up the search of biological patterns, reducing computational time from hours to minutes [3]. Hardware speed comes from the ability to compute all WFA states simultaneously. Actually, genomic data (input string) are processed on-the-fly, and the performance of a hardwired WFA is mainly determined by the input data rate. Thus, the processing time becomes independent of the WFA size, and is only dictated by the time for accessing all the items of the database.

1.1. WFA on Field Programmable Gate Arrays

FPGA (Field Programmable Gate Array) technology aims to bridge the gap between ASIC (Application Specific Integrated Circuit) components and conventional microprocessors [5]. Any function or algorithm can be hardwired in a few milliseconds into such a programmable support. Compared to a Von Neumann machine, which requires several programming steps, the result can be computed in a single cycle. Compared to an ASIC of which implementation is definitive, its *programmability* allows users to use a large number of different hardware configurations.

Basically, an FPGA is a matrix of Look-Up Tables (LUTs) with a flexible interconnection (Fig. 1). Each LUT is a 2^n-bit memory, which can be configured to compute any $\langle n \mapsto 1 \rangle$ function (n binary inputs, 1 binary output). This LUT is often tightly coupled with a 1-bit register. Recent FPGAs (2006) can hold up to 200,000 LUTs with 6 inputs.

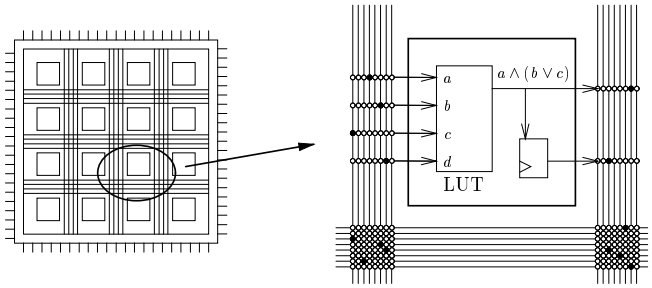


Fig. 1. Structure of an FPGA. On the left, the global structure. On the right, detail of a 16-bit Look-Up Table (LUT) generating a $\langle 4 \mapsto 1 \rangle$ function.

The hardware implementation of WFA is a linear encoding scheme that maps each state and each transition to a given logic cell. The limitations of such an implementation are in the surface area of the circuit (related to the number of transitions of the WFA and their bit width) and the clock frequency (related to the critical path, see below).

Moreover, this implementation is valid as long as the WFA *fits* into FPGA components. Unfortunately, biological patterns may require consequent reconfigurable

resources, particularly when insertion/deletion errors are considered. In that case, insertions are modeled by cyclic transitions and deletions by ε-transitions. Resulting WFA are thus much larger in terms of the number of transitions. From a hardware point of view, the resources are directly related to the number of transitions to hardwire. Hence, finding equivalent automata with less transitions is highly beneficial.

Besides the automaton size, a direct hardware implementation of ε-transitions is not realistic. Fig. 2 exemplifies the hardware mapping of a WFA with ε-transitions. Paths with ε-transitions are represented by dotted lines: they systematically bypass state registers. The main consequence is that a long *critical path* (dashed line) is created from the input to the output. The critical path is defined as the longest path between two registers, and determines the maximum clock frequency of the circuit. The longer the path, the lower the frequency. Hence, to keep a reasonable working frequency, the critical path needs to be broken into smaller parts by removing some ε-transitions.

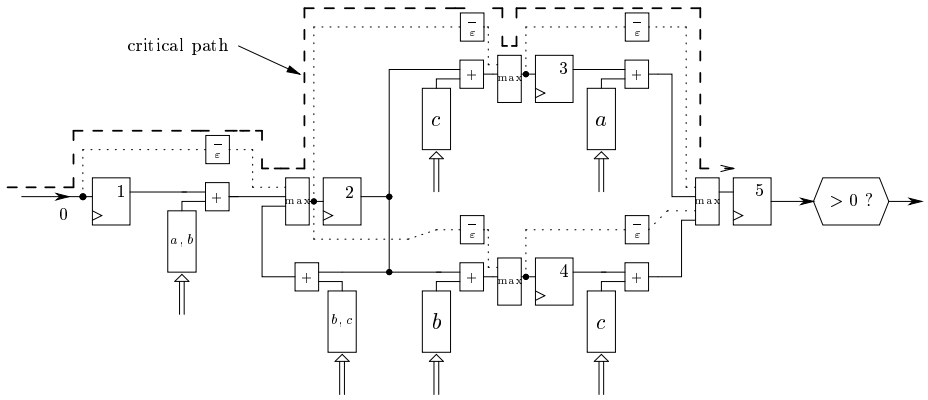


Fig. 2. Hardwiring a WFA [4] with 5 regular transitions doubled with ε-transitions. A critical path runs through the whole automaton.

1.2. Removals of ε-transitions

The classical method for removing ε-transitions in automata uses the ε-closure of every state [1, 14]. Mohri proposed for WFA a generic algorithm with a smallest distance method [10]. A certain condition must be checked to ensure that the weights are well-defined in cycles.

These algorithms can raise the number of transitions from n to $\mathcal{O}(n^2)$. The resulting automaton can be minimized [11], but for large automata, such a limit makes the hardware implementation impossible. As an example, in [12], we experienced an 80-state automaton for discovering olfactory receptor genes in the dog genome. On this automaton, the classical ε-transitions-removal algorithms produce more than 3100 new transitions. This number reaches the limit of today's FPGA technology and prevents larger automata from being hardwired.

Hromkovic proposed a study for ε-transitions in finite automata [7]. There are rational expressions of size $\mathcal{O}(n)$ such that every ε-free recognizing automaton has a size $\Omega(n \log n)$. Lifshits raised this bound to $\Omega(n \log^2 n / \log\log n)$ [9], then Schnigter raised it to $\Omega(n \log^2 |\Sigma|)$ [13]. The same paper provided a $\mathcal{O}(n \log n \log |\Sigma|)$ upper bound. Other works optimized the creation time of those automata [6]. In [8], it is shown that there are some languages recognized by NFA with $\mathcal{O}(n \log n)$ transitions, but such that every ε-free recognizing NFA has $\Omega(n^2)$ transitions.

In this paper we study the *development* of WFA: we double every transition of a WFA with an ε-transition, and we study the number of new transitions created by removing the ε-transitions. Here a transition is a function assigning a value to each letter of Σ. We previously proposed a first study for linear-shaped automata: in this case, we designed an optimal method that produces automata with $\Theta(n \log n)$ new transitions, without a minimization phase [4].

The rest of the paper is organized as follows. Section 2 provides WFA background. Then, in Section 3, we study the development of WFA for acyclic automata. Section 4 presents a new problem driven by the hardware constraints: the removal while accepting short ε-chains.

2. Background

2.1. WFA and Pattern Matching

We consider automata with weights on the transitions. The weights are in the set $\{-\infty, \ldots, -1, 0, 1, \ldots\}$.

Definition 1 *A Weighted Finite Automaton (WFA) is a 5-tuple* $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$*, where* Q *is a finite set of states,* Σ *a finite alphabet,* $\Delta \subset Q \times Q \times (\delta : \Sigma \cup \{\varepsilon\} \mapsto \mathbb{Z} \cup \{-\infty\})$ *a finite transition table,* $I \subseteq Q$ *and* $F \subseteq Q$ *the sets of initial and final states.*

The number of transitions of the WFA is $|\Delta|$. For each transition $\tau = (q, q', \delta) \in \Delta$, we denote by $\mathsf{i}[\tau] = q$ its initial state, $\mathsf{f}[\tau] = q'$ its final state, and $\delta[\tau] = \delta$ its weight function. A WFA without ε-transitions is a WFA such that $\delta(\varepsilon) = -\infty$ for every transition (q, q', δ). Now we define paths as consecutive labeled transitions:

Definition 2 *A path* $\pi = (\tau_1, \alpha_1) \ldots (\tau_k, \alpha_k) \in (\Delta \times (\Sigma \cup \{\varepsilon\}))^*$ *in a WFA* $\mathcal{A}$ *is a succession of pairs of transitions and characters where the transitions* $\tau_1, \ldots, \tau_k$ *are consecutive transitions, that is* $\mathsf{f}[\tau_i] = \mathsf{i}[\tau_{i+1}]$ *for* $i = 1 \ldots k-1$*, and where the characters* α_i *are in* $\Sigma \cup \{\varepsilon\}$*. The label of* π *is the word* $\alpha_1 \ldots \alpha_k$*.*

The weight function δ can be extended to paths: for a path $\pi = (\tau_1, \alpha_1) \ldots (\tau_k, \alpha_k)$, we define $\delta(\pi) = \delta[\tau_1](\alpha_1) + \ldots + \delta[\tau_k](\alpha_k)$. Weights on words are computed as weights on paths between some initial and final states. For pattern matching applications, those weights are compared to a fixed threshold.

2.2. Path-equivalence

Now we give a definition of our ε-transition-removal problem. We define it as finding a new automaton with a special kind of equivalence, the path-equivalence, which requires that some paths (the closed paths, see below) have a superior path in the corresponding automaton.

Definition 3 *One path π is superior to another one π' if both paths have the same label, the same initial state and the same final state, and if $\delta(\pi) \geq \delta(\pi')$.*

Definition 4 *A path $\pi = (\tau_1, \alpha_1) \dots (\tau_k, \alpha_k)$ is left-closed if it begins with an initial state ($\mathfrak{i}[\tau_1] \in I$) or if its first character α_1 is different than ε. Similarly, a path is right-closed if $\mathfrak{f}[\tau_k] \in F$ or $\alpha_k \neq \varepsilon$. A path is closed if it is closed at both sides.*

Definition 5 *Two WFA $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ and $\mathcal{A}' = (Q, \Sigma, \Delta', I, F)$ are path-equivalent if every closed path in $\mathcal{A}$ labeled by a word $w \neq \varepsilon$ has a superior path in $\mathcal{A}'$ and reciprocally.*

Basically, the path-equivalence states that the two automata simulate each other through their paths. Usual algorithms that remove the ε-transitions such as [14] or [10] produce path-equivalent automata.

2.3. Development of an Automaton

Given a WFA without ε-transitions $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ and a deletion cost c_ε, we define $\mathcal{A}_\varepsilon$ as the WFA in which all transitions of $\mathcal{A}$ are doubled by ε-transitions. More precisely, every transition $(q, q', \delta) \in \Delta$ is updated with $\delta(\varepsilon) = c_\varepsilon$.

Definition 6 *Given a WFA $\mathcal{A}$, any WFA $\mathcal{A}'$ is a development of $\mathcal{A}$ if $\mathcal{A}'$ is path-equivalent to $\mathcal{A}_\varepsilon$ and has no ε-transitions. We say that $\mathcal{A}'$ is developed from $\mathcal{A}$ if $\mathcal{A}'$ is a development of $\mathcal{A}$.*

To be efficiently hardwired, a WFA needs to be developed with *as few new transitions as we can.* In the general case, the ε-transitions-removal from an automaton with n transitions gives an automaton with $\mathcal{O}(n^2)$ new transitions. Depending on the topology of the automaton, the optimal number of new transitions is in the range from $\mathcal{O}(1)$ to $\Omega(n^2)$ (Fig. 3).

In [4], we studied the case of automata with a simple topology. A *linear-shaped* WFA is a WFA $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ whose states can be ordered in a way that $Q = \{q_o, q_1, \dots q_n\}$, $I = \{q_0\}$, $F = \{q_n\}$, and its transitions are exactly the n transitions (q_{i-1}, q_i, δ_i) for $i = 1, 2, \dots n$. We designed an optimal method that produces automata with $\Theta(n \log n)$ new transitions, without a minimization phase.

3. Removal in Acyclic Automata

Here we use the results on linear-shaped WFA to analyze the number of new transitions in the developments of some more generic automata. To ensure that the weights are well defined, automata with cycles require special constraints [10]. The section 3.1 considers *acyclic automata* with n states : we give an upper bound to develop such automata. The section 3.2 extends the result to automata with cycles, but with no cycles on ε-transitions. Such automata are common in biological

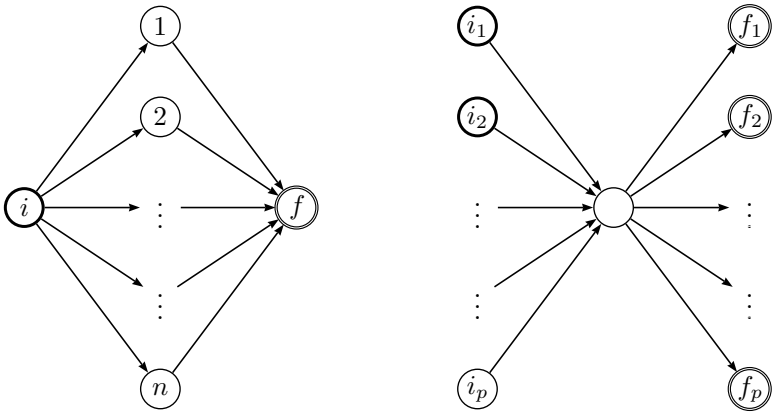


Fig. 3. When $n = 2p + 1$, both automata have $n + 2$ states. The left one can be developed with one new transition (q_i, q_f, δ), whereas the right one must have not less than the $(n-1)^2/4$ new transitions $(i_a, f_b, \delta_{a,b})$ to be developed.

applications (Fig. 4).

3.1. Acyclic Automata

Definition 7 *A WFA $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ is acyclic if its graph has no cycle. The states of an acyclic WFA can be numbered $q_1, q_2, \ldots q_n$ such than there is no backward transition $(q_i, q_j, \delta_{i,j})$ with $i \geq j$.*

We call such a WFA a *numbered* automaton. The algorithm 1 develops a numbered automaton with n states from the development of two sub-automatons obtained by cutting the automaton at a state q_z.

In the algorithm for linear-shaped WFA (Algorithm 1 in [4]), initial and final states of both sub-automata guarantee that the paths are closed. Here some transitions are *cut* over q_z (Fig. 5). All the paths are closed if one adds to each sub-automaton a set of states Z that *touches* the cut transitions, that is a set Z such that any cut transition starts or ends in Z. Each state in Z is a final state for the left sub-automaton and an initial state for the right one : the sub-automata are overlapping. We have the following property:

Property 1 *The algorithm 1 builds an automaton which is path-equivalent to the initial automaton.*

Proof. We just give the sketch of the proof, which is similar to the case of linear-shaped WFA (Lemma 3 in [4]).

$\boxed{\mathcal{A} \mapsto \mathcal{A}'}$ Each closed path of $\mathcal{A}$ not labeled by ε and not completely included in $\mathcal{A}_1$ or in $\mathcal{A}_2$ can be written as $\pi_1\pi_2$, where π_1 and π_2 are closed paths in $\mathcal{A}_1$ and $\mathcal{A}_2$. Any such decomposition leads to a superior closed path in $\mathcal{A}'$.

$\boxed{\mathcal{A}' \mapsto \mathcal{A}}$ Reciprocally, any closed path of $\mathcal{A}'$ either goes through a state $q \in \{q_z\} \cup Z$, or jumps over such a state. In both cases, a superior closed path of $\mathcal{A}$ can

Algorithm 1 Development of a numbered WFA

Input: a numbered WFA with n states $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, an integer $z \in [2, n-1]$, a cost c_ε

Let $\mathcal{C}$ be the set of all cut transitions $(q_i, q_j, \delta_{i,j})$ with $i < z < j$
Let Z be a set of states touching $\mathcal{C}$, with $|Z| \leq |\mathcal{C}|$

Let $\mathcal{A}_1 = (Q_1 = \{q_1 \dots q_z\} \cup Z, \Sigma, \Delta_1, I, \{q_z\} \cup Z)$
and $\mathcal{A}_2 = (Q_2 = \{q_z \dots q_n\} \cup Z, \Sigma, \Delta_2, \{q_z\} \cup Z, F)$
where the transition tables Δ_1 and Δ_2 are the restrictions of Δ on Q_1 and Q_2

Let $\mathcal{A}'_1$ and $\mathcal{A}'_2$ recursively be two developments of $\mathcal{A}_1$ and $\mathcal{A}_2$
Let $\mathcal{A}'$ be the concatenation of $\mathcal{A}'_1$ and $\mathcal{A}'_2$: $\mathcal{A}' = (Q, \Sigma, \Delta', I, F)$, $\Delta' = \Delta'_1 \cup \Delta'_2$

For all q_i in Q_1
Add to Δ' the transition (q_i, q, δ'_i) for all final states $q \in F$
with $\delta'_i(\alpha) = \max_{i+1 \leq k \leq n} [(n-i-1)c_\varepsilon + \delta_k(\alpha)]$

For all q_i in Q_2
Add to Δ' the transition (q, q_i, δ''_i) for all initial states $q \in I$
with $\delta''_i(\alpha) = \max_{1 \leq k \leq i} [(i-1)c_\varepsilon + \delta_k(\alpha)]$

Output: the WFA $\mathcal{A}' = (Q, \Sigma, \Delta', I, F)$

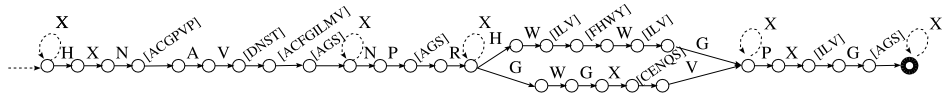


Fig. 4. Detail of a genomic automaton recognizing MIP membrane proteins [2]. The complete automaton has more than 300 transitions. Except for some insertion transitions (X), this automaton is acyclic.

be reconstructed. □

Each step of the algorithm adds no more than $|Q_1| \cdot |F| + |Q_2| \cdot |I|$ transitions. To bound this value, we need a bound on $|Z|$.

Definition 8 *Let there be a numbered WFA with states* $\{q_1, q_2 \dots q_n\}$*, and* q_z *a state. The* width κ_z *is the number of transitions* (q_a, q_b, δ) *with* $a < z < b$*.*

The *maximal width* is $\mathcal{K} = \max_i \kappa_i$. It can be seen as the maximal number of branches in the WFA, except the main branch. We always have $\mathcal{K} \leq n$. On the automaton depicted on Fig. 4, we have $\mathcal{K} = 1$ for all numberings. In the general case, the widths depend on the chosen numbering.

At each step, the set Z has no more than $\mathcal{K}$ elements. When applying recursively algorithm 1, the sets I and F will always have no more than $\mathcal{K}+1$ elements. Then one step of the algorithm adds no more than $(|Q_1|+|Q_2|)\cdot(\mathcal{K}+1) \leq (n+\mathcal{K}+1)\cdot(\mathcal{K}+1)$ transitions. We thus have the following consequence of Property 1:

Property 2 *Any numbered WFA with a maximal width* $\mathcal{K}$ *can be developed with*

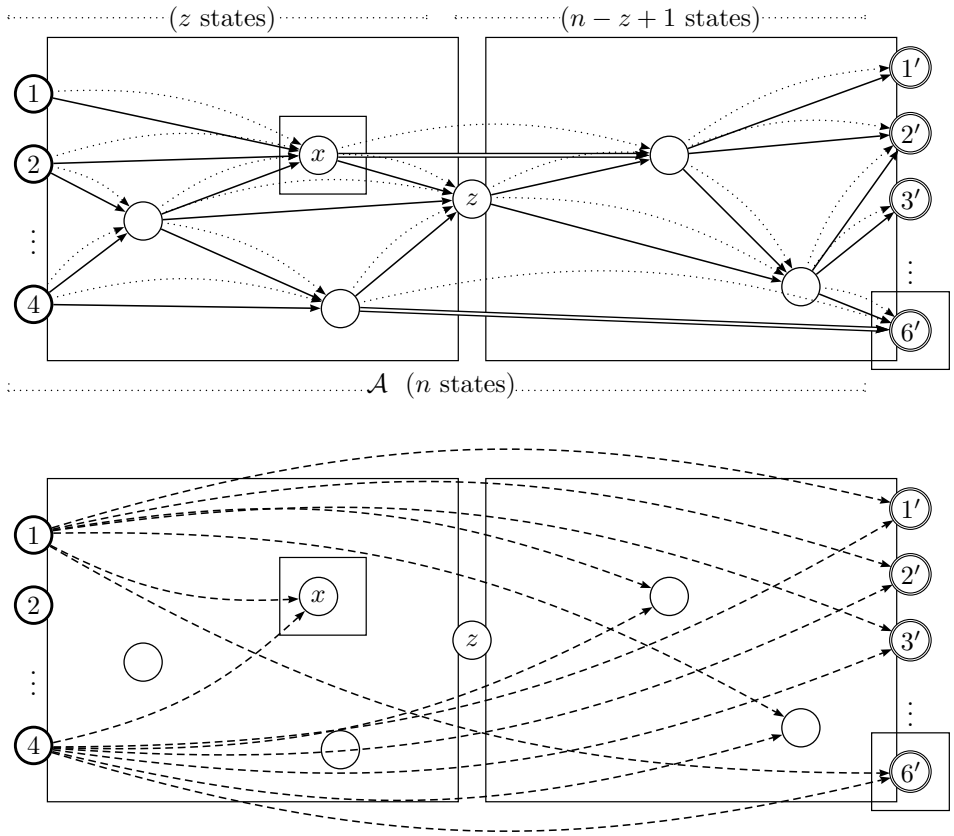


Fig. 5. Algorithm developing a numbered WFA with n states. A state q_z is chosen to split the automaton into two parts with z and $n - z + 1$ states. The two cut transitions are shown in double lines. The set $Z = \{x, 6'\}$ touches every cut transition. This set Z is added to the two parts to give the sub-automata $\mathcal{A}_1$ and $\mathcal{A}_2$. Final states of $\mathcal{A}_1$ (and initial states of $\mathcal{A}_2$) are $\{z\} \cup Z$. At the bottom, we add to the developments of the two sub-automata transitions from initial states of $\mathcal{A}_1$ to all states of $\mathcal{A}_2$. With the symmetrical operation, no more than $|Q_1| \cdot |F| + |Q_2| \cdot |I|$ transitions are created.

$\mathcal{O}((\mathcal{K} + 1) \cdot n \log n)$ *transitions.*

This coarse bound guarantees that automata with a small maximum width are developed with very few new transitions (Tab. 1). This is sufficient for real-life genomic automata representing biological features. Such automata, hand-crafted or computed by state-merging techniques [2], are compounds of a few linear-shaped parts (Fig. 4).

For the *lower bound*, the generic argument on linear-shaped WFA can be applied to the longest path in the WFA. If this longest path has a size $\ell \leq n$, we have a bound of $\Omega(\ell \log \ell)$. In fact, the maximal cut $\mathcal{K}$ is not a good metric for the lower bound. On the Fig. 3, the left automaton has, for any numbering, a maximal cut of $n - 1$, but is developed with only one new transition.

Given a WFA without ε-transitions $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, but *with cycles*, how can we extend the previous bounds ? We can consider a slightly modified automaton in the development. We choose an acyclic sub-automaton $\mathcal{A}_1 = (Q, \Sigma, \Delta_1, I, F)$ with $\Delta_1 \subset \Delta$, then we develop $\mathcal{A}_1$ by the algorithm 1 and we add the remaining transitions.

Concretly, by doubling *some* transitions of $\mathcal{A}$ by ε-transitions, we obtain an *ε-acyclic automaton*, that is an automaton without cycles of ε-transitions (Fig. 6). As an ε-acyclic automaton has a numbering with no backward ε-transition, the algorithm 1 can still be used while ignoring the backward regular transitions. The same bound of $\mathcal{O}((\mathcal{K}+1)\cdot n \log n)$ is obtained (each width κ_i is now the number of ε-transitions cut by the state q_i).

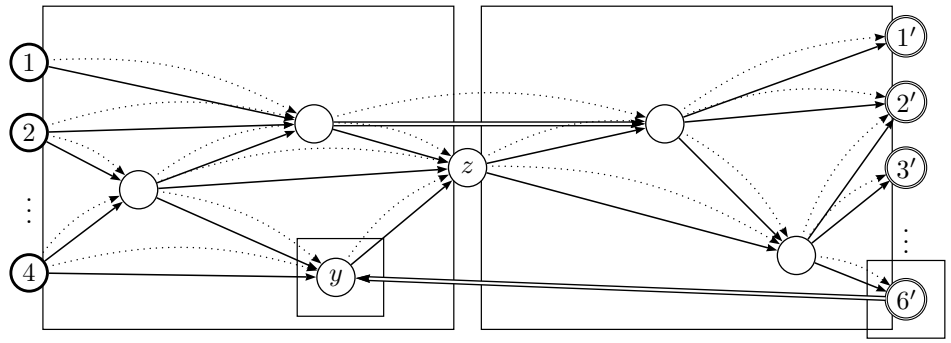


Fig. 6. Unlike the automaton on Fig. 5, this numbered automaton has a backward transition $(6', y, \delta)$. However, that transition is not doubled with an ε-transition.

In real applications, if we have an automaton $\mathcal{A}$ without ε-transitions, we add *some* ε-transitions while keeping the automaton ε-acyclic. This construction is justified when the automaton represents biological structures made of similar units. Those units are separated by sequences that cannot be deleted, as for instance in the case of exon recognition.

4. Removal with Short ε-Chains

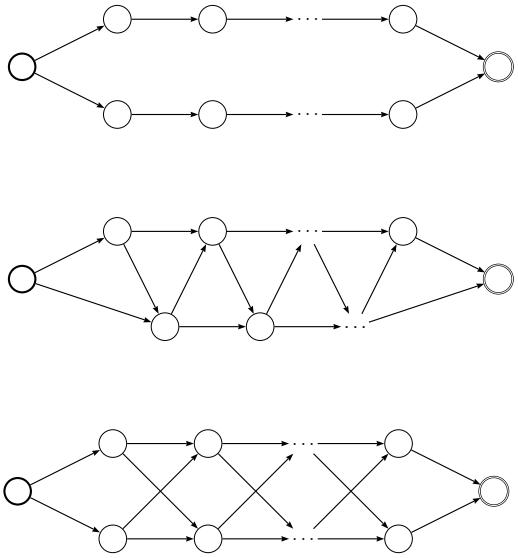
To further lower the number of new transitions, we proposed in [4] to only approximate the chains of ε-transitions by restricting the number of consecutive ε-transitions to be followed. Here we propose to apply this idea *only on the hardware* with a partial removal of ε-transitions. This time, the resulting automaton will be path-equivalent to the initial one, without approximation.

The key idea of this section is the remark that short *ε-chains* (that is, chains of successive ε-transitions) can be actually hardwired with a reasonable critical path (Fig. 8). Thus, we propose to extend the ε-transitions-removal problem to allow short chains of successive ε-transitions. The following is a generalization of Def. 6.

Definition 9 *Given a WFA $\mathcal{A}$ and an integer $n_\varepsilon \in \mathbb{N}$, any WFA $\mathcal{A}''$ is a develop-*

Table 1. Number of new transitions produced while removing ε-transitions on various automata. Data presented here depend only of the topology on the WFA, and not of the actual values of the transitions. For the quadratical algorithms (without further minimization) and the linear-shaped WFA, the results depend only of the number of states n. For acyclic WFA, we give some bounds (see Fig. 7 for examples of topologies). In the case of "separate branches", only the first cut has the maximal width $\mathcal{K}$; the following cuts consider linear-shaped parts. Even in the worst-case situation, when the WFA is branching at every state, genomic WFA with 80 states and $\mathcal{K} = 2$ can be efficiently hardwired with less than 1300 new transitions.

Number of states of the initial automaton (n)	20	80	200
Quadratical ε-transitions-removal algorithms	190	3160	19900
Linear-shaped WFA [4]	69	433	1345
Linear-shaped WFA, development with short ε-chains (section 4)			
ε-chains of length $\leq n_\varepsilon = 3$	42	310	1022
ε-chains of length $\leq n_\varepsilon = 5$	30	250	890
(ε-)acyclic WFA (section 3)			
$\mathcal{K} = 1$, two separate branches	96	522	1555
$\mathcal{K} = 1$, worst-case	141	925	2835
$\mathcal{K} = 2$, three separate branches	124	612	1766
$\mathcal{K} = 2$, worst-case	176	1292	4096



(a) $\mathcal{K} = 1$, two separate branches

(b) $\mathcal{K} = 1$, branching at every state

(c) $\mathcal{K} = 2$, branching at every state

Fig. 7. Example of different WFA topologies reflecting best-case and wort-case of the Tab. 1. Complexity range from best-case ((a): only one additional branch through the whole WFA) to the worst-case ((b) and (c): each cut has a maximal width: the automaton is constantly branching).

ment with short ε-chains of $\mathcal{A}$ if $\mathcal{A}''$ is path-equivalent to $\mathcal{A}_\varepsilon$ and if all ε-chains of

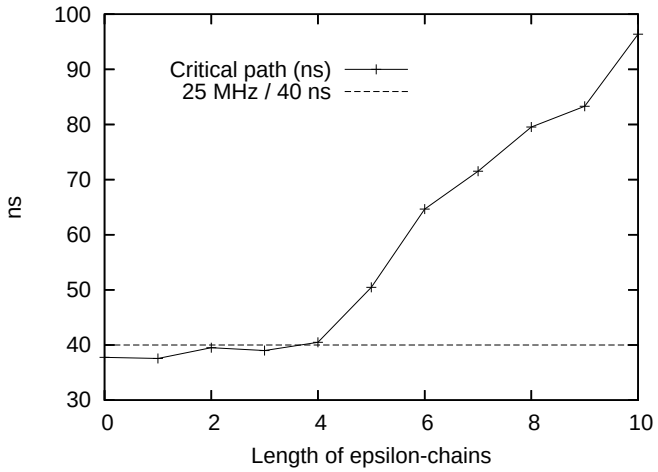


Fig. 8. Critical path for ε-chains with 8-bit weights and a 40 ns (25 MHz) constraint on a WFA with 20 regular transitions and different lengths of ε-chains. Chains of 3 ε-transitions can be hardwired. With smaller bit weights or newer FPGA, this number can be increased to 10 successive ε-transitions. The FPGA being half-filled (between 48% and 51%), the hardware compiler has a moderate pressure on the different optimisation phases. The critical path should be linear to the length of ε-chains, but the hardware compiler does not further minimize it as soon as it meets the constraint.

$\mathcal{A}''$ have a length $\leq n_\varepsilon$.

Given a linear-shaped WFA with n states, we can split it into n_ε parts of size $\mathcal{O}(n/n_\varepsilon)$, develop each sub-automaton with $\mathcal{O}(n/n_\varepsilon \cdot \log(n/n_\varepsilon))$ transitions, and finally add an ε-transition that covers each sub-automaton.

Thus we have the following property:

Property 3 *A linear-shaped WFA with n states can be developed with short ε-chains with $\mathcal{O}(n \log(n/n_\varepsilon))$ new transitions.*

Furthermore, if we restrict that all remaining ε-transitions are *original*, that is, they were in the automaton before the removal, we have a lower bound:

Property 4 *Given n_ε, any development with short original ε-chains of a linear-shaped WFA with n states has $\Omega(n/n_\varepsilon \cdot \log(n/n_\varepsilon))$ new transitions.*

The proof uses a similar technique to the proof of Lemma 6 in [4], but additional work is done to handle the short ε-chains. The proof, given in the next paragraph, enumerates some sets in which at least one transition must appear in the automaton. The *span* of a transition (q_i, q_j, δ) is $|j - i|$.

Proof. Let $\mathcal{A}$ be a linear-shaped WFA with n states, and $\mathcal{A}''$ a development with short original ε-chains of $\mathcal{A}$. Let $\pi = (\tau_{a+1}, \alpha_A)(\tau_{a+2}, \varepsilon) \ldots (\tau_{b-1}, \varepsilon)(\tau_b, \alpha_B)$ be a closed path in $\mathcal{A}$, where α_A and α_B are two characters different from ε. This path has a superior path π' in $\mathcal{A}''$ that can be written as $\pi' = (\pi'_1, \varepsilon)\ (\tau_A, \alpha_A)\ (\pi'_2, \varepsilon)\ (\tau_B, \alpha_B)\ (\pi'_3, \varepsilon)$.

As the three paths π'_1, π'_2 and π'_3 are original ε-chains, any of them has a span not greater than n_ε transitions, that is $3n_\varepsilon$ globally. Therefore, at least one of the two transitions τ_A and τ_B has a span included in $\{\lceil \frac{k-3n_\varepsilon}{2} \rceil, \ldots, k-1\}$ with $k = b-a$ (Fig. 9).

If we consider all $n-k+1$ pairs (a,b) with the same $k = b-a$, then the WFA $\mathcal{A}''$ has no less than $(n-k+1)/2(n_\varepsilon+1)$ transitions of span included in the set $S_k = \{\lceil \frac{k-3n_\varepsilon}{2} \rceil, \ldots, k-1\}$.

Let (k_i) be a sequence defined by $k_{i+1} = 2k_i + 3n_\varepsilon$ and $k_0 = 1$. We have $k_i = 2^i(1+3n_\varepsilon) - 3n_\varepsilon$. We consider several ks taking the values of (k_i) from $i = 1$ to the last i such that $k_i \leq n$, that is $i_f = \left\lfloor \log \frac{n+3n_\varepsilon}{1+3n_\varepsilon} \right\rfloor = \Theta(\log(n/n_\varepsilon))$.

As the sets of spans S_{k_i} and S_{k_j} are disjoint as soon as $i \neq j$, the WFA $\mathcal{A}''$ has not less than $\Sigma_{i=1}^{i_f}(n-k_i+1)/2(n_\varepsilon+1) = \Theta(n/n_\varepsilon \cdot \log(n/n_\varepsilon))$ transitions. □

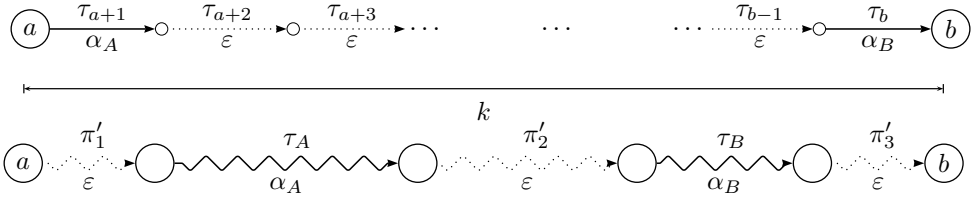


Fig. 9. Proof of the property 4. At least one of the transitions τ_A and τ_B has a span included in $\{\lceil \frac{k-3n_\varepsilon}{2} \rceil, \ldots, k-1\}$, where $k = b-a$ is the span of the path $(\tau_A, \alpha_A)(\tau_B, \alpha_B)$.

Although accepting short chains of ε-transitions is a local change, this technique lowers the actual number of new transitions (Table 1). The ε-chains can be used in (ε)-acyclic WFA to obtain $\mathcal{O}((\mathcal{K}+1) \cdot n \log(n/n_\varepsilon))$ new transitions.

5. Conclusions and Perspectives

The removal techniques presented in sections 3 and 4 allow larger automata to be hardwired on a given FPGA. For acyclic automata, the best results for a strict application of algorithm 1 would require finding the numbering of the states that minimizes the maximal width $\mathcal{K}$. In fact, for real automata with a small number of branches as the one in Fig. 4, good solutions are found when cutting at the branching states.

Other studies could find more precise bounds. For acyclic automata, the initial number of transitions could be taken into account. Finally, we plan to study *approximated developments* of automata, in which the resulting automaton would not strictly be path-equivalent to the initial one. In real applications, the cost assigned to deletions prevents sequences with too many ε-transitions from being accepted.

References

1. A.V. Aho, R. Sethi and J.D. Ullman, *Compilers, Principles, Techniques and Tools,* (Addison Wesley, 1986).
2. F. Coste and G. Kerbellec, "A similar fragments merging approach to learn automata on proteins," in *16th European Conference on Machine Learning Machine Learning (ECML 2005), LNCS 3720* (2005), pp. 522–529.
3. M. Giraud and D. Lavenier, "Linear encoding scheme for weighted finite automata," in *9th International Conference on Implementation and Application of Automata (CIAA 2004), LNCS 3317* (2004), pp. 146–155.
4. M. Giraud and D. Lavenier, "Dealing with hardware space limits when removing epsilon-transitions in a genomic weighted finite automaton," *Journal of Automata, Languages and Combinatorics* **10** (2005), pp. 265–285.
5. M. B. Gokhale and P. S. Graham, *Reconfigurable Computing – Accelerating Computation with Field-Programmable Gate Arrays*, (Springer, 2005).
6. C. Hagenah and A. Muscholl, "Computing ε-free NFA from regular expressions in $\mathcal{O}\big(n \log^2(n)\big)$ time," *Mathematical Foundations of Computer Science (MFCS 1998), LNCS 1450* (1998), pp. 277–285.
7. J. Hromkovic, S. Seibert and T. Wilke, "Translating regular expressions into small epsilon-free nondeterministic finite automata," in *14th Symposium on Theoretical Aspects of Computer Science (STACS 1997), LNCS 1200* (1997), pp. 55–66.
8. J. Hromkovic and G. Schnitger, "NFAs With and Without ε-Transitions," in *32nd International Colloquium on Automata, Languages and Programming (ICALP 2005), LNCS 3580* (2005), pp. 385–396.
9. Y. Lifshits, "A lower bound on the size of ε-free NFA corresponding to a regular expression," *Information Processing Letters* **85** (2003), pp. 293–299.
10. M. Mohri, "Generic epsilon-removal and input epsilon-normalization algorithms for weighted transducers," *International Journal of Foundations of Computer Science* **13** (2002), pp. 129–143.
11. M. Mohri, "Finite-State Transducers in Language and Speech Processing". *Computational Linguistics* **23** (1997), pp. 269–311.
12. P. Quignon, M. Giraud, M. Rimbault, P. Lavigne, S. Tacher, E. Morin, E. Retout, A.S. Valin, K. Lindblad-Toh, J. Nicolas and F. Galibert, "The dog and rat olfactory receptor repertoires," *Genome Biology* **6** (2005), R83.
13. G. Schnitger, "Regular Expressions and NFAs Without ε-Transitions," in *23th Symposium on Theoretical Aspects of Computer Science (STACS 2006), LNCS 3884* (2006), pp. 432–443.
14. K. Thompson, "Regular expression search algorithm," *Communications of the ACM,* **11** (1968), pp. 419–422.