Büchi automata reduction by means of left and right trace inclusion preorders

J.-M. Champarnaud and F. Coulon

May 12, 2004

Abstract

The size reduction of Büchi automata has been widely investigated, particularly with the aim of reducing the complexity of state model checking. This paper is interested in the automata theoretic part of the reduction problem. We manipulate left and right traces of states in finite automata deduced from the Büchi automaton. The left (resp. right) trace is the set of calculations leading to (resp. starting from) a state. Simulations enable a fast detection of these trace inclusions. We aim to extend known results about NFAs to NBAs, and we test the practical relevance of this extension by proceeding to random tests. The reduction is analyzed on Büchi automata that have already been reduced by use of the delayed simulation and other classical reduction techniques used by SPIN. Our final reduction algorithm has an $\mathcal{O}(mna)$ time complexity, where m is the number of transitions, n the number of states and a is the number of accepting states in the automaton we consider.

1 Introduction and definitions

In this study, we are interested by two particular state machines: Nondeterministic Finite Automata (NFAs) and Nondeterministic Büchi Automata (NBAs) [**?**]. Our aim is to extend known results about NFAs to NBAs, and to test the practical relevance of this extension by proceeding to random tests.

The reduction of the size of NBAs that are manipulated is of critical importance since the space complexity of operations on state machines constitute the main obstacle to model checking [**?**]. There are several stages on which reduction can be performed. This paper is only interested in the automata theoretic stage.

Both NFAs and NBAs are given as quintuples $< Q, \Sigma, \delta, I, F >$, where Q is a set of states, Σ is an alphabet, δ is a nondeterministic transition function that maps $Q \times \Sigma$ on 2^Q, I is either the set of initial states (for NFAs) or the initial state (for NBAs), F is either the set of final states (for NFAs) or the set of accepting states (for NBAs).

The notion of a state simulating an other state has been widely studied during the last decades [**?**, 2]. A simulation relation is a preorder on the set of states — a reflexive and transitive relation. It is usually defined in the context of game theory. There are several notions of simulation, but in general a simulation is a maximal preorder that respects at least the following assumption: if a state q simulates another state p, then for all $a \in \Sigma$ and all states $p' \in \delta(p, a)$, there exists a state $q' \in \delta(q, a)$ that simulates p'. The assertion "q simulates p" relates, in the context of games, to the following notion: "player at q cannot loose against a player at p", or more practically: "whatever move player at p chooses, the player at q can move (reading the same letter as p) so that q still simulates p at next turn. The practical interest of simulations for automata, is to detect inclusion of languages with a small polynomial complexity, and equalities of languages by double inclusion.

Let us mention four important simulations in the history of automata and Büchi automata reduction. The *ordinary simulation* does not deal with final states. It corresponds to the preceding schema, without more assumptions. The *direct simulation* relates to the inclusion of languages in

NFAs. We add the following assumption: if a state q simulates a state p, and p is final, then q must be final.

The *fair simulation* relates to the inclusion of languages in NBAs. A state q is said to fair simulate another state p iff all ω-words accepted from p are accepted from q. And finally, a state q *delayed simulates* a state p iff for all infinite path π_p outgoing from p, there is an infinite path π_q outgoing from q, labeled by the same ω-word, and such that, if an accepting state is encountered in π_p at step i, then an accepting step is encountered at some step $j \geq i$ of π_q.

Let $\mathscr{A} =< Q, \Sigma, \delta, I, F >$ be an NFA.

Definition 1 *The language of $\mathscr{A}$ is denoted $\mathcal{L}(\mathscr{A})$.*

The left language *of a state q, denoted $\overleftarrow{\mathcal{L}}^{\mathscr{A}}(q)$ is the set $\{w \in \Sigma^* \mid q \in \delta(I, w)\}$.*

The right language *of a state q, denoted $\overrightarrow{\mathcal{L}}^{\mathscr{A}}(q)$ is the set $\{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\}$.*

In the literature, the right language of a state is often referred to as the *trace* of the state, or as its *future*. The left language of a state can also be referred to as its *past*. The direct simulation enables to detect the right language inclusion of two states: if q direct simulates p, then $\overrightarrow{\mathcal{L}}(p) \subseteq \overrightarrow{\mathcal{L}}(q)$. Notice that the converse is false in general.

If the direct simulation relation is computed on the reverse automaton, then we can detect inclusions of left languages in the same manner.

Definition 2 *The* reverse automaton *of $\mathscr{A}$, denoted by $\overline{\mathscr{A}}$, is the quintuple $< Q, \Sigma, \overline{\delta}, F, I >$ where $q' \in \overline{\delta}(q, a)$ iff $q \in \delta(q', a)$ for all $q, q' \in Q$ and $a \in \Sigma$.*

If q direct simulates p in the reverse automaton, then $\overleftarrow{\mathcal{L}}(p) \subseteq \overleftarrow{\mathcal{L}}(q)$.

In our paper [1] we investigate how to compute efficiently the direct simulation and how the detected left and right language inclusions can be combined to reduce an NFA. The present paper aims to extend these results to NBAs. The technique we propose performs transformations that do not change certain finitary languages of the automaton. Indeed, the ω-language $\mathcal{L}^\omega$ recognized by a Büchi automaton can be given by

$$\mathcal{L}^\omega = \bigcup_{f \in F} \mathcal{L}(i, f).\mathcal{L}(f, f)^\omega$$

where $L(q, p)$ stands for the language of the finite automaton $< Q, \Sigma, \delta, q, \{p\} >$. So, if the automaton is transformed without changing the finitary languages $\mathcal{L}(i, f)$ and $\mathcal{L}(f, f)$ for any f, then the language of the Büchi automaton is unchanged.

The first Section recalls a part of the results about NFAs. Section 3 describes the extension we propose for Büchis automata, and Section 4 gives the performances we have observed on random tests.

2 Language preorders and reduction of NFAs

Most of the following results about NFAs are from [1], so that proofs are omitted.

Definition 3 *Let $q \in Q$. The* fan out *of q is denoted $\overrightarrow{q}$ and defined by $\overrightarrow{q} = \{(a, q') \in \Sigma \times Q \mid q' \in \delta(q, a)\}$. Symmetrically, the* fan in *of q is denoted $\overleftarrow{q}$ and defined by $\overleftarrow{q} = \{(q', a) \in Q \times \Sigma \mid q \in \delta(q', a)\}$.*

The elements of the fan in and fan out of q are called arrows.

Reduction algorithms which are considered in the following are algorithms which proceed *merging* some states of the original NFA. Let us introduce a general framework for this kind of manipulation.

Definition 4 *Let q and p be two states of $\mathscr{A}$. We let $merge(\mathscr{A}, q, p)$ stand for an automaton obtained from $\mathscr{A}$ by merging q and p, that is, p is deleted and some arrows are added to $\overleftarrow{q}$ and $\overrightarrow{q}$, so that $\overleftarrow{\mathcal{L}}^{merge(\mathscr{A},q,p)}(q) = \overleftarrow{\mathcal{L}}^{\mathscr{A}}(q) \cup \overleftarrow{\mathcal{L}}^{\mathscr{A}}(p)$ and $\overrightarrow{\mathcal{L}}^{merge(\mathscr{A},q,p)}(q) = \overrightarrow{\mathcal{L}}^{\mathscr{A}}(q) \cup \overrightarrow{\mathcal{L}}^{\mathscr{A}}(p)$.*

Whatever we know about A, we can always build $merge(A, q, p)$ by adding $\overleftarrow{p}$ to $\overleftarrow{q}$ and adding $\overrightarrow{p}$ to $\overrightarrow{q}$.

Lemma 1 *Let $q, p \in Q$, $merge(\mathscr{A}, q, p)$ recognizes $\mathcal{L}(\mathscr{A})$ if and only if $\overleftarrow{\mathcal{L}}^{\mathscr{A}}(q) \cdot \overrightarrow{\mathcal{L}}^{\mathscr{A}}(p) \subseteq \mathcal{L}(\mathscr{A})$ and $\overleftarrow{\mathcal{L}}^{\mathscr{A}}(p) \cdot \overrightarrow{\mathcal{L}}^{\mathscr{A}}(q) \subseteq \mathcal{L}(\mathscr{A})$.*

In particular, two states q and p can be merged without changing the language of the automaton if one of the three following conditions is true:

- $\overrightarrow{\mathcal{L}}(q) = \overrightarrow{\mathcal{L}}(p)$ (C1)
- $\overleftarrow{\mathcal{L}}(q) = \overleftarrow{\mathcal{L}}(p)$ (C2)
- $\overrightarrow{\mathcal{L}}(q) \subseteq \overrightarrow{\mathcal{L}}(p)$ and $\overleftarrow{\mathcal{L}}(q) \subseteq \overleftarrow{\mathcal{L}}(p)$ (C3)

Unfortunately, if one considers the relation $\sim_0$: $q \sim_0 p$ if q and p can be merged, then one does not get an equivalence relation. Hence we cannot reduce the automaton by computing the quotient w.r.t. $\sim_0$.

Indeed, it appears that $\sim_0$ is the union of two equivalence relations. Hence we have a solution that consists in "splitting" the relation $\sim_0$ into the *left mergeable* relation and the *right mergeable* relation.

Definition 5 *Two states q and p are said to be* left mergeable*, noted $q \overleftarrow{\sim}_0 p$, if condition (C2) or (C3) holds. They are said to be* right mergeable*, noted $q \overrightarrow{\sim}_0 p$, if condition (C1) or (C3) holds.*

These new relations are equivalence relations, so that the automaton can be quotiented using one of them.

Definition 6 *The automaton quotiented by $\overrightarrow{\sim}_0$ (resp. $\overleftarrow{\sim}_0$), is obtained by merging in any order pairs of states equivalent w.r.t. $\overrightarrow{\sim}_0$ (resp. $\overleftarrow{\sim}_0$).*

Theorem 1 *Relations $\overrightarrow{\sim}_0$ and $\overleftarrow{\sim}_0$ are equivalence relations on Q. Moreover, the language of $\mathscr{A}$ is preserved by quotienting w.r.t. $\overrightarrow{\sim}_0$ and by quotienting w.r.t. $\overleftarrow{\sim}_0$.*

2.1 Practical algorithms for computing left and right preorders

The exact computation of $\overrightarrow{\sim}_0$ and $\overleftarrow{\sim}_0$ is known to have an exponential complexity, see e.g. [4]. Indeed, the exact preorders can be calculated within time $O(n^2 N)$ [1], where N is the maximum between the number of reachable subsets in the automaton and its reverse.

Nevertheless, we generally need approximations running in polynomial time. The direct simulation constitutes a highly efficient approximation. The direct simulation over $\mathscr{A}$ is a preorder $\overrightarrow{\subseteq}_1$ over the set of states Q. It can be computed in time mn, where n is the number of states in the NFA, m is the number of transitions [3, 1]. If a state q direct simulates another state p, i.e. $p \overrightarrow{\subseteq}_1 q$, then $\overrightarrow{\mathcal{L}}(p) \subseteq \overrightarrow{\mathcal{L}}(q)$. The direct simulation over the reverse automaton $\overline{\mathscr{A}}$ is also calculated and noted $\overleftarrow{\subseteq}_1$. Naturally we have the dual implication $p \overleftarrow{\subseteq}_1 q \Rightarrow \overleftarrow{\mathcal{L}}(p) \subseteq \overleftarrow{\mathcal{L}}(q)$.

Let us define the three following conditions on states p, q:

- $q \overrightarrow{\subseteq}_1 p$ and $p \overrightarrow{\subseteq}_1 q$ (D1)
- $q \overleftarrow{\subseteq}_1 p$ and $p \overleftarrow{\subseteq}_1 q$ (D2)
- ($q \overleftarrow{\subseteq}_1 p$ and $q \overrightarrow{\subseteq}_1 p$) or ($p \overleftarrow{\subseteq}_1 q$ and $p \overrightarrow{\subseteq}_1 q$) (D3)

We have the following equivalence relations on states:

Definition 7 *Let $\mathscr{A}$ be an NFA. Let $\overrightarrow{\subseteq}_1$ and $\overleftarrow{\subseteq}_1$ be respectively the direct simulation over $\mathscr{A}$ and $\overline{\mathscr{A}}$. We define the two relations $\overleftarrow{\sim}_{\mathscr{A}}$ and $\overrightarrow{\sim}_{\mathscr{A}}$ over the states of $\mathscr{A}$ by letting for any two states p, q:*

- $p \overleftarrow{\sim} q$ *iff condition (D2) or (D3) is true*
- $p \overrightarrow{\sim} q$ *iff condition (D1) or (D3) is true*

Theorem 2 *Relations $\overleftarrow{\sim}_{\mathscr{A}}$ and $\overrightarrow{\sim}_{\mathscr{A}}$ are equivalence relations on Q. Moreover, the language of $\mathscr{A}$ is preserved by quotienting w.r.t $\overleftarrow{\sim}_{\mathscr{A}}$ and by quotienting w.r.t. $\overrightarrow{\sim}_{\mathscr{A}}$.*

3 Preorders in Büchi automata

Kousha Etessami *et al.* have provided an mn^2 algorithm for computing the *delayed simulation*, which is a highly powerful preorder for detecting states that can be merged. This preorder deals with the right behavior of states: that is, being in a state, we consider the set of possible future accepted calculations. We have seen that the left behavior (the set of possible calculations leading to a given state), is likely to provide further reduction. We investigate how to combine right and left behavior inclusions in the context of Büchi automata.

Let $\mathscr{B} =< Q, \Sigma, \delta, i, F >$ be an NBA with m transitions and n states.

The conditions for deciding whether two states can be merged in a Büchi automaton or an NFA are different. Indeed, let us consider $\mathscr{A} =< Q, \Sigma, \delta, \{i\}, F >$, the NFA having the same structure as $\mathscr{B}$. There are well known counter-examples where two states are mergeable in $\mathscr{A}$ because of any of the three conditions D1, D2 or D3, but are not mergeable in $\mathscr{B}$. See figure 1. States 2 and 3 have the same left language $a(b + a^2)^*$. The NBA recognizes $a(b + a^2)^* a^\omega$ before merging. It recognizes $a(b + a^2)^\omega$ after merging.

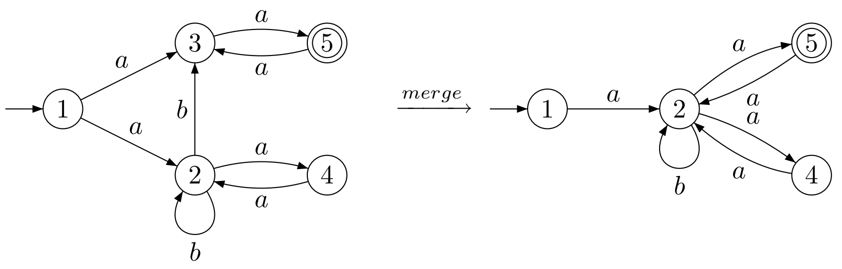


Figure 1: Example where states 2 and 3 cannot be merged in the NBA, while they can be merged in the NFA.

So, two states having the same left language are not necessarily mergeable in the Büchi automaton. The same problem happens with right languages: see Figure 2. States 1 and 2 have the same right language in the NFA, that is a^+. Although, the Büchi automaton initially recognizes a^ω. Once 1 and 2 are merged, it recognizes $(a + b)a^\omega$.

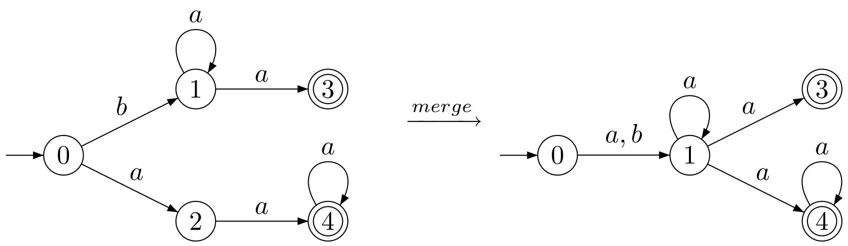


Figure 2: Example where states 1 and 2 cannot be merged in the NBA, while they can be merged in the NFA.

Given two states having the same infinitary right language, one can even not guarantee that these two states are Büchi-mergeable. An example of non mergeable states having the same infinitary right language is given by Etessami, Wilke and Schuller in [2].

We now give two equivalence relations that only rely on finitary languages, though they enable to quotient the Büchi automaton.

Definition 8 *Let $\mathscr{A} =< Q, \Sigma, \delta, I, F >$ be either an NFA or an NBA, and two states q, p, we note $\mathscr{A}(q,p)$ the NFA obtained as $< Q, \Sigma, \delta, \{q\}, \{p\} >$.*

The two equivalence relations $\overset{\leftarrow}{\sim}$ and $\overset{\rightarrow}{\sim}$ previously defined for NFAs are extended. Indeed, the language recognized by a Büchi automaton can be characterized by many ways. The following particular characterization relies only on finitary languages:

$$\mathcal{L}(\mathscr{B}) = \bigcup_{f \in F} \mathcal{L}(\mathscr{B}(i,f)).\mathcal{L}(\mathscr{B}(f,f))^{\omega}$$

This leads us to introduce the two following relations on states:

Definition 9 *Define the equivalence relations $\overset{\leftarrow}{\sim}_{\mathscr{B}}$ and $\overset{\rightarrow}{\sim}_{\mathscr{B}}$ by letting*

- $q \overset{\leftarrow}{\sim}_{\mathscr{B}} p$ *iff* $q \overset{\leftarrow}{\sim}_{\mathscr{B}(i,f)} p$ *and* $q \overset{\leftarrow}{\sim}_{\mathscr{B}(f,f)} p,\ \forall f \in F$
- $q \overset{\rightarrow}{\sim}_{\mathscr{B}} p$ *iff* $q \overset{\rightarrow}{\sim}_{\mathscr{B}(i,f)} p$ *and* $q \overset{\rightarrow}{\sim}_{\mathscr{B}(f,f)} p,\ \forall f \in F$

Indeed, two states are Büchi-mergeable if merging them does not change the language from the initial state to any accepting state, neither than the loop language of any accepting state.

Theorem 3 *The relations $\overset{\leftarrow}{\sim}_{\mathscr{B}}$ and $\overset{\rightarrow}{\sim}_{\mathscr{B}}$ are equivalence relations, and the language of $\mathscr{B}$ is preserved by quotienting by $\overset{\leftarrow}{\sim}_{\mathscr{B}}$ or by $\overset{\rightarrow}{\sim}_{\mathscr{B}}$. These relations can be computed in time $O(mna)$ and space $O(mn)$, where a is the number of accepting states in $\mathscr{B}$.*

A basic reduction technique consists in quotienting alternatively by $\overset{\leftarrow}{\sim}_{\mathscr{B}}$ and $\overset{\rightarrow}{\sim}_{\mathscr{B}}$. Notice that $\overset{\leftarrow}{\sim}_{\mathscr{B}}$ and $\overset{\rightarrow}{\sim}_{\mathscr{B}}$ are themselves modified during the quotienting operation. This suppose they are recomputed before each quotient calculation.

One can easily see that the delayed bi-simulation relation over $\mathscr{B}$ [2] is more accurate than $\overset{\rightarrow}{\sim}_{\mathscr{B}}$ for detecting states that are equivalent because of condition (D1), though it does not detect states that are useless because of left and right trace inclusion (condition D3). So, $\overset{\rightarrow}{\sim}_{\mathscr{B}}$ is not comparable to the delayed simulation and one can expect an improved reduction by combining them. Nevertheless, the time complexity for the delayed simulation has been stated at $\mathcal{O}(mn^3)$ in [2].

4 Practical tests

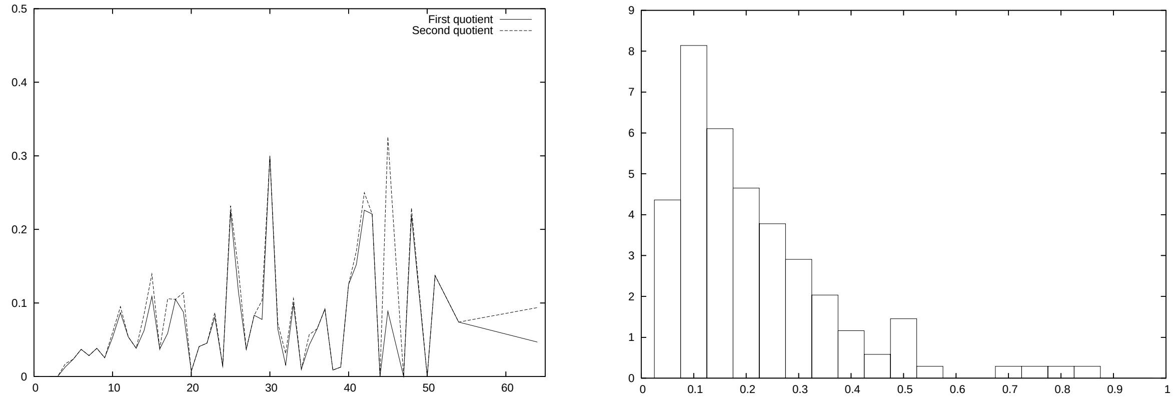
We have carried out tests on random generated LTL formulas to compare the reduction obtained by $\overset{\leftarrow}{\sim}_{\mathscr{B}}$ and $\overset{\rightarrow}{\sim}_{\mathscr{B}}$ to the reduction based on the delayed simulation.

An LTL formula is generated as a random expression tree, each node is randomly assigned an operator, respecting a given percentage for each type. Operators are divided in three types: boolean operators, safety operators (until, release, weak or not), and liveness operators (always, eventually, next). They are built on an alphabet of two or three atomic propositions.

The LTL formulas are first given to the software TMP (Temporal Massage Parlor version 2.0, available at `http://www.bell-labs.com/project/TMP/`), written by Kousha Etessami, which generates reduced Büchi automata by use of the delayed simulation [2] and various optimizations [**?**]. The resulting Büchi automata are then reduced by three successive quotientings: first by $\overset{\leftarrow}{\sim}_{\mathscr{B}}$ then by $\overset{\rightarrow}{\sim}_{\mathscr{B}}$ and then once more by $\overset{\leftarrow}{\sim}_{\mathscr{B}}$. The reduction ratio obtained at the end of each quotienting is shown with respect to the number of states in the Büchi automaton given by TMP. The graphs in Figure 3 and 4 has been traced using respectively 551 random expressions built on 2 atomic propositions and 449 random expressions built on 3 atomic propositions, involving 15 operators, 5 of each type. In each Figure, the first graphic shows the average reduction ratio obtained for a given size of the automaton output by TMP. The second graphic shows the percentage of tests leading to a given reduction ratio. Small generated automata, having less than 5 states, are not taken into account in the percentages.

Most of generated expressions lead to small Büchi automata, so that the first curves are smoother at the beginning. Nevertheless, the ratio seems rather constant, somewhat between 0.05 and 0.1. Only two curves are shown, corresponding to the first two quotientings. The third revealed to be useless, except on 3 expressions.

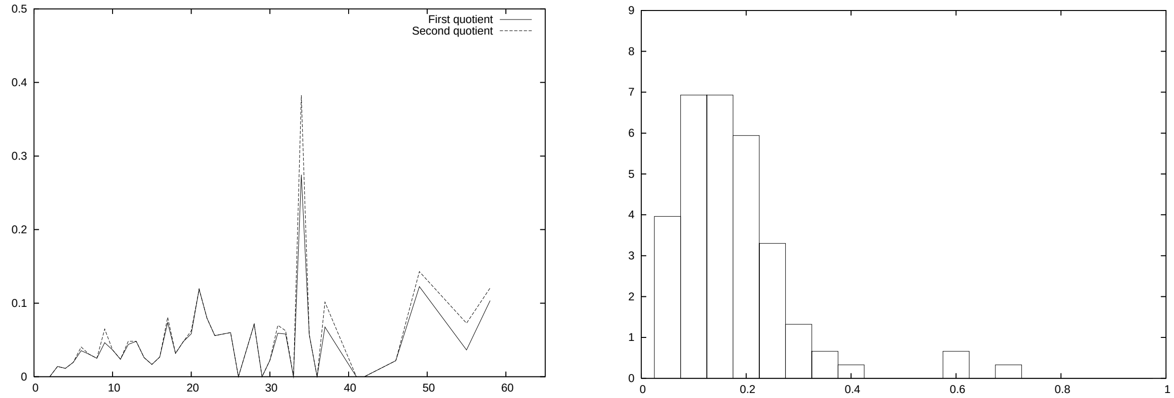
The percentages given by the second graphics reveal that in some case, we obtain a very important reduction.



Average reduction ratio / number of states in the Büchi generated by TMP.

Percentage of tests leading to a given reduction ratio. 63% of tests lead to no reduction.

Figure 3: Reduction ratio over random expressions with 15 operators, on 2 atomic propositions, *i.e.* 4 letters.



Average reduction ratio / number of states in the Büchi generated by TMP.

Percentage of tests leading to a given reduction ratio. 70% of tests lead to no reduction.

Figure 4: Reduction ratio over random expressions with 15 operators, on 3 atomic propositions, *i.e.* 9 letters.

5 Conclusion

The algorithm we have given and tested relies on the left and right relations of Definition 9. Though the constraint over states in relation seems very strong (we have to verify two relations per accepting states), the observed efficiency is surprisingly high. It seems that in practice, a state is not involved in many loop languages (languages of $\mathscr{B}(f, f)$). Also, automata generated from LTL formulas generally comes with few accepting states.

The main asset of this technique, is that two states q and p can be detected mergeable by combining different reasons: they may be mergeable because of the "useless condition" (D3) for some involved finitary language, and be mergeable for "equality condition" (D2) in some other finitary language.

References

[1] J.-M. Champarnaud and F. Coulon. NFA reduction algorithms by means of regular inequalities. In *Lecture Notes in Computer Science*, volume 2710, pages 194–205. Springer, 2003.

[2] Kousha Etessami, Rebecca Schuller, and Thomas Wilke. Fair simulation relations, parity games, and state space reduction for Bchi automata. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Collquium*, volume 2076 of *Lecture Notes in Computer Science*, pages 694–707, Crete, Greece, July 2001. Springer.

[3] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. of 36th IEEE Symp. on Foundations of Comp. Sci. (FOCS'95)*, pages 453–462, 1995.

[4] H. Hunt, D. Rosenkrantz, and T. Szymanski. On the equivalence, containment and covering problems for the regular and context-free languages. *J. Comput. System Sci.*, 12:222–268, 1976.