

Efficiently Testing Emptiness of Automata on Linear Orderings

Thomas Braipson^[0009–0003–0543–5265] and Tom Clara^[0009–0009–2938–1410]

Montefiore Institute, B28, University of Liège, Belgium
`{Thomas.Braipson,Tom.Clara}@uliege.be`

Abstract. Automata on linear orderings are a type of finite-state automata introduced by Bruyère and Carton, that unify and broadly extend the usual finite-state automata on finite words, (bi-)infinite words, and ordinal words. These automata read words indexed by linear orderings, defined as mappings from the elements of any linear ordering to a finite alphabet. Several authors have studied the emptiness problem for these automata, on several domains, mainly to obtain decision procedures for some logics. The resulting algorithms are polynomial. In this paper, we review them and provide new algorithms, whose complexity is improved by one power in time and space. In particular, their space complexity is linear.

Keywords: Finite Automata · Linear Orderings · Emptiness Test

1 Introduction

Finite-state automata are expressive objects that are easy to manipulate algorithmically. For these reasons, their various forms have been used for deciding several logical fragments. For instance, Büchi introduced automata on infinite words to establish the decidability of the second-order theory of one successor (S1S) [4]. The strategy is to transform a formula into an automaton whose accepted language corresponds to encodings of the models of the formula, and then test if this language is empty. Automata on linear orderings are a broad extension of automata on infinite words, that accept words indexed by linear orderings [3,1]. These automata have been used in decision procedures for linear-time temporal logic on any linear domain [6,8] and for the monadic first-order theory of order, on the domains $\mathbb{R}$ and $\mathbb{Q}$ [2]. In the literature, emptiness tests for automata on linear orderings on several domains are polynomial in time and have a quadratic space complexity. The contribution of this paper is to provide new algorithms, whose complexity is improved by one power in time and in space. This is an essential step towards the implementation of efficient decision procedures that use automata on linear orderings as data structures.

The paper is structured as follows. We start by recalling basic notions about words and automata on linear orderings in Section 2. Then in Section 3 we present an emptiness test for words indexed by scattered linear orderings due

to [5] and a general one due to [7]. Finally, we introduce new tests of improved complexity in Section 4.

2 Basic Notions

2.1 Linear Orderings and Cuts

We recall elementary notions on linear orderings and refer to [9] for further details. A *linear ordering* is a set J equipped with a binary relation $<$ that is a *total order*. A *subordering* of J is a subset $I \subseteq J$ equipped with the same total order, restricted to I .

Two elements $j < k$ are said to be *consecutive* if there does not exist any element between them. In that case, j is the *predecessor* of k and k is the *successor* of j . If a linear ordering does not have any consecutive elements, it is *dense*. Conversely, a linear ordering without any infinite dense subordering is *scattered*.

A *cut* of a linear ordering J is a partition (K, L) of J such that for every $k \in K$ and $\ell \in L$, we have $k < \ell$. The set of cuts of the ordering J is written $\hat{J}$. This set can be linearly ordered by the relation $(K_1, L_1) < (K_2, L_2)$ iff $K_1 \subsetneq K_2$ (or equivalently, $L_2 \subsetneq L_1$). For any linear ordering J , $\hat{J}$ contains a least element $\hat{J}_{min} = (\emptyset, J)$ called the *first cut* and a greatest element $\hat{J}_{max} = (J, \emptyset)$ called the *last cut*. If (K, L) is a cut different from $\hat{J}_{min}$ and $\hat{J}_{max}$ such that K does not have a greatest element and L does not have a least element, this cut is a *gap*. An ordering for which there does not exist any gap is *complete*.

2.2 Words and Automata on Linear Orderings

The definitions of this section are borrowed from [3,1]. A *word indexed* by a linear ordering J is a mapping $w : J \rightarrow \Sigma$, where Σ is a finite set called the *alphabet*. We also say that J is the *length* of w and write it $|w| = J$.

Definition 1. An automaton on linear orderings is a tuple $\mathcal{A} = (Q, \Sigma, I, F, \Delta)$ where:

- Q is a finite set of states,
- Σ is a finite alphabet,
- $I \subseteq Q$ is a set of initial states,
- $F \subseteq Q$ is a set of final states, and
- $\Delta \subseteq (Q \times \Sigma \times Q) \cup (2^Q \times Q) \cup (Q \times 2^Q)$ is a transition relation.

The transition relation Δ is composed of three types of transitions. Those of the form $(p, a, q) \in Q \times \Sigma \times Q$ are called *successor* transitions and coincide with the usual notion of transition in automata. In a transition (p, a, q) , p is called the *origin*, a the *label*, and q the *destination*. Transitions of the form $(P, q) \in 2^Q \times Q$ (alternatively noted $P \rightarrow q$) are called *left-limit* transitions. In that case, P is called the *limit set* and q is called the *destination*. Symmetrically, transitions

of the form $(q, P) \in Q \times 2^Q$ (alternatively noted $q \rightarrow P$) are called *right-limit* transitions, have as origin q and as limit set P . Several limit transitions sharing the same limit set can be written in a compact form, by grouping them around the limit set, e.g. $P \rightarrow q_0$, $q_1 \rightarrow P$, and $q_2 \rightarrow P$ are written $q_1, q_2 \rightarrow P \rightarrow q_0$. The *cardinality* of a limit transition is the cardinality of its limit set.

Definition 2. A run of an automaton on linear orderings $\mathcal{A} = (Q, \Sigma, I, F, \Delta)$ reading a word w indexed by J is a mapping $\rho : \hat{J} \rightarrow Q$ that satisfies the following conditions:

- For every pair of cuts $c_1 = (K_1, L_1)$ and $c_2 = (K_2, L_2)$ such that c_2 is the successor of c_1 in $\hat{J}$, $(\rho(c_1), w(j), \rho(c_2)) \in \Delta$, where $j \in J$ is such that $K_2 = K_1 \cup \{j\}$.
- For every cut $c \neq \hat{J}_{\min}$ which does not have any predecessor, $(\lim_{c^-} \rho, \rho(c)) \in \Delta$, where $\lim_{c^-} \rho$ is the left-limit set of ρ in c , defined as

$$\lim_{c^-} \rho = \{q \in Q \mid \forall c_1 < c . \exists c_2 . c_1 < c_2 < c \wedge \rho(c_2) = q\}.$$

- For every cut $c \neq \hat{J}_{\max}$ which does not have any successor, $(\rho(c), \lim_{c^+} \rho) \in \Delta$, where $\lim_{c^+} \rho$ is the right-limit set of ρ in c , defined as

$$\lim_{c^+} \rho = \{q \in Q \mid \forall c_1 > c . \exists c_2 . c < c_2 < c_1 \wedge \rho(c_2) = q\}.$$

This run is *accepting* if $\rho(\hat{J}_{\min}) \in I$ and $\rho(\hat{J}_{\max}) \in F$.

A word w is *accepted* by an automaton $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ that reads w . The class of the words accepted by $\mathcal{A}$ is its *accepted language*.

Intuitively, a run reads a word from a state to another and respects transitions locally. In the case of two successive cuts, the automaton must be able to read the letter that labels the element “between” these two cuts. If a cut is not the first one and does not have any predecessor, then it is the limit of an increasing sequence of cuts. In that case, this cut can be reached if a Muller-like condition is respected: a specific set of states must appear exactly infinitely often, arbitrarily close to that cut without predecessor. This is a left-limit transition. Right-limit transitions are symmetrical. If the run starts in an initial state and ends in a final state, the automaton accepts the word.

A left-limit transition $P \rightarrow q$ is *enabled* if there exists a run ρ reading a word w of length J and a cut $c \in \hat{J}$ such that $\rho(c) = q$ and $\lim_{c^-} \rho = P$. A right-limit transition is enabled under symmetric conditions. We illustrate these definitions by means of some examples.

Example 1. The automaton $\mathcal{A}$ in Fig. 1.a accepts the words that can be described as ω copies of bi-infinite words on the alphabet $\{a, b\}$ that do not have two successive occurrences of the same letter¹. Indeed, from q_0 to itself the automaton

¹ Actually, all these words are the same up to an isomorphism. We refer to [3] for a formal definition of isomorphism between words, a notion not necessary in this work.

can read bi-infinite words, without occurrences of the factors aa or bb . If this is done infinitely often, the limit transition $\{q_0, q_1, q_2\} \rightarrow q_3$ can be followed and the resulting word is accepted.

Example 2. The automaton $\mathcal{B}$ in Fig. 1.b accepts all the non-empty words w on the alphabet $\{a\}$ such that

- the linear ordering $|w|$ does not have a least (resp. greatest) element, since the only transition leaving (resp. reaching) the initial (resp. final) state is a right-limit (resp. left-limit) transition,
- $|w|$ is dense, since it is impossible to read two consecutive letters,
- and for any two elements $i < j \in |w|$, there exists a gap (K, L) such that $i \in K$ and $j \in L$. Informally speaking, the elements of $|w|$ and its gaps need to be densely interleaved.

For instance, the word $w : \mathbb{Q} \rightarrow \{a\}$ is accepted by $\mathcal{B}$. Let us describe an accepting run ρ reading this word. It maps the first cut to q_0 and the last cut to q_4 . The remaining cuts of $|w|$ form three groups: those immediately before a letter a , those immediately after a letter a , and the gaps. These three groups are mapped by ρ to q_1 , q_2 , and q_3 , respectively. Notice that each irrational number x corresponds to a gap of $\mathbb{Q}$, since the sets $\{r \in \mathbb{Q} \mid r < x\}$ and $\{r \in \mathbb{Q} \mid r > x\}$ partition $\mathbb{Q}$ (here, $<$ denotes the ordering relation of $\mathbb{R}$). Therefore, it is clear that there exists a gap of $\mathbb{Q}$ between any two rational numbers. Hence in every right neighbourhood of a cut c without successor (different from the last one), there are cuts mapped to q_1 , q_2 and q_3 by ρ . A right-limit transition involving $\{q_1, q_2, q_3\}$ can thus be followed to leave such a cut c . A similar analysis holds for a cut without predecessor. This shows that ρ is indeed a run reading w .

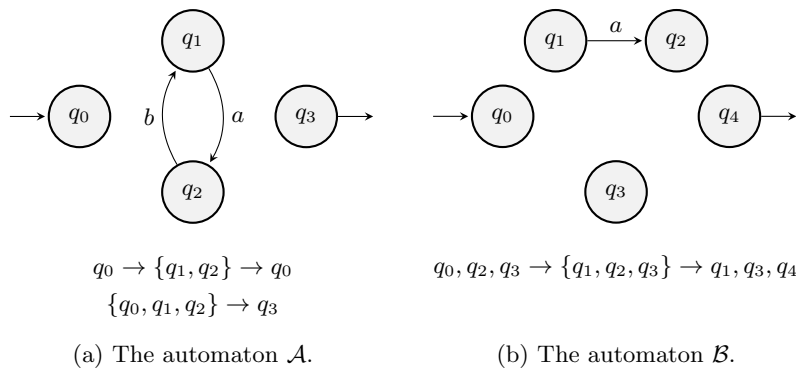


Fig. 1: Examples of automata on linear orderings.

3 State of the Art

In this section, we review several algorithms to check the emptiness of automata on linear orderings, for several classes of word lengths. We first recall the cases of finite and ω -words. Then, we present tests dedicated to any scattered linear ordering and any linear ordering. These tests are the basis of the efficient algorithms introduced in Section 4.

3.1 Finite and ω -words

Automata on linear orderings without limit transitions read finite words, similarly to non-deterministic finite-state automata on finite words. Therefore, to test whether an automaton on linear orderings accepts a finite word, one can ignore its limit transitions and look for a final state reachable from an initial one, through a finite sequence of successor transitions.

Non-deterministic Muller automata can be seen as automata on linear orderings, where accepting sets of states play the role of left-limit transitions leading to a final state. The existence of a limit set that is reachable from an initial state and that forms a strongly connected component implies that such an automaton accepts an ω -word.

3.2 Words Indexed by Scattered Linear Orderings

An emptiness test for words indexed by scattered linear orderings is proposed in [5]. Given an automaton on linear orderings $\mathcal{A}$, it checks the existence of a final state reachable from an initial one, by following any kind of transitions, provided that they are compatible with words of scattered length. To do so, the algorithm progressively builds a *reachability automaton*. It is a finite-word automaton whose states are those of $\mathcal{A}$ and whose transitions encode the possibility of moving from one state to another by visiting some set of states in $\mathcal{A}$. Therefore, a transition in a reachability automaton is labelled by a subset of states of $\mathcal{A}$.

Concretely, the algorithm checks whether each limit transition of $\mathcal{A}$ is enabled. Whenever it is the case, fresh transitions are added to the reachability automaton to encode this information. Due to the Muller-like semantics of limit transitions in automata on linear orderings (i.e., states in the limit set must appear exactly infinitely often to enable the transition), a limit transition of limit set P can only be enabled by following successor transitions or limit transitions that involve limit sets included in P . Therefore, the test processes limit transitions by increasing cardinality (which strengthens the order given by inclusion). Once each limit transition has been processed, the emptiness of the resulting reachability automaton (which is a finite-word automaton) is tested.

We now formalise this algorithm. First, we define the notion of path in a reachability automaton.

Definition 3. A path $p \xrightarrow{P} q$ in a reachability automaton is a finite sequence of states $p_1, \dots, p_n$, with $n > 1$ such that there exists a transition (p_i, P_i, p_{i+1}) for all $0 < i < n$, $p = p_1$, $q = p_n$, and $P = \bigcup_{0 < i < n} P_i$.

The intuition of this definition is that a path collects the labels of the transitions that it follows by taking their union.

Formally, for an automaton on linear orderings $\mathcal{A} = (Q, \Sigma, I, F, \Delta)$, the algorithm computes the reachability-automaton sequence $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_M$, where M is the cardinality of the largest limit set in $\mathcal{A}$. These automata are recursively defined as $\mathcal{A}_i = (Q, 2^Q, I, F, \Delta_i)$, with

$$\begin{aligned} \Delta_0 &= \{(p, \{p, q\}, q) \mid (p, a, q) \in \Delta\} \\ \Delta_i &= \Delta_{i-1} \cup \Delta_i^L \cup \Delta_i^R \quad \text{for } 0 < i \leq M \\ \Delta_i^L &= \{(p, P \cup \{q\}, q) \mid P \rightarrow q \in \Delta \text{ and } |P| = i \text{ and } C\} \\ \Delta_i^R &= \{(q, P \cup \{q\}, p) \mid q \rightarrow P \in \Delta \text{ and } |P| = i \text{ and } C\} \end{aligned}$$

where the condition C imposes the existence of a path of the form $p \xrightarrow{P} p$ in $\mathcal{A}_{i-1}$.

The key property of the reachability automaton $\mathcal{A}_i$ is that the label of a path in between two states p and q is equal to a set of states that can be visited in $\mathcal{A}$, to move from p to q , while following limit transitions of cardinality at most i .

Example 3. Consider the automaton $\mathcal{A}$ in Fig. 1.a. First, $\mathcal{A}_0$ is built by replacing the labels of the successor transitions of $\mathcal{A}$ with their respective set of origin and destination states. Since there is not any limit transition of cardinality 1, $\mathcal{A}_1$ is equal to $\mathcal{A}_0$. Then, since the path $q_1 \xrightarrow{\{q_1, q_2\}} q_1$ exists in $\mathcal{A}_1$, the limit transitions $q_0 \rightarrow \{q_1, q_2\} \rightarrow q_0$ are transformed into the transitions $(q_0, \{q_0, q_1, q_2\}, q_1)$ and $(q_1, \{q_0, q_1, q_2\}, q_0)$. The same arguments yield the transitions $(q_0, \{q_0, q_1, q_2\}, q_2)$ and $(q_2, \{q_0, q_1, q_2\}, q_0)$. The result is $\mathcal{A}_2$, in Fig. 2.a, where we denote transitions of label $\{q_1, q_2\}$ by dashed arrows and those of label $\{q_0, q_1, q_2\}$ by dotted arrows.

These transitions endow $\mathcal{A}_2$ with the path $q_0 \xrightarrow{\{q_0, q_1, q_2\}} q_0$. Together with the limit transition $\{q_0, q_1, q_2\} \rightarrow q_3$, this yields the transition $(q_0, \{q_0, q_1, q_2, q_3\}, q_3)$. The paths $q_1 \xrightarrow{\{q_0, q_1, q_2\}} q_1$ and $q_2 \xrightarrow{\{q_0, q_1, q_2\}} q_2$, together with $\{q_0, q_1, q_2\} \rightarrow q_3$, yield similar transitions. The result, $\mathcal{A}_3$, is depicted in Fig 2.b, where these transitions are written $\dashrightarrow$. In this reachability automaton, there exists a path from q_0 to q_3 . This shows that $\mathcal{A}$ accepts at least one word of scattered length.

Lemma 1. *The emptiness for words indexed by scattered linear orderings can be tested in time $\mathcal{O}(|\Delta|^2 \cdot |Q|^2)$ and space $\mathcal{O}(|\Delta| \cdot |Q|)$.*

Proof. Each reachability automaton $\mathcal{A}_i$ has $\mathcal{O}(|\Delta| \cdot |Q|)$ transitions, since each enabled limit transition induces as many transitions as states in its limit set, and there are $\mathcal{O}(|\Delta|)$ limit transitions and $\mathcal{O}(|Q|)$ states in a limit set. Given

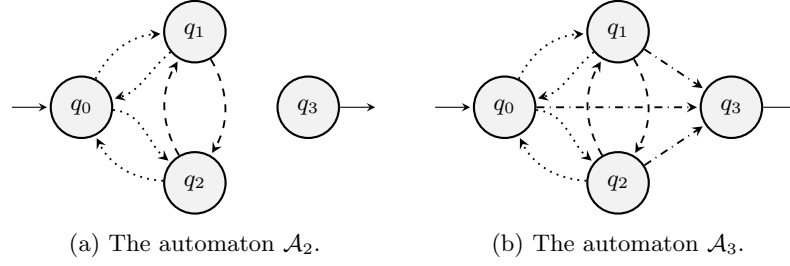


Fig. 2: Illustration of the test for scattered orderings.

that only the most recent reachability automaton needs to be stored in memory, the space complexity of the algorithm is $\mathcal{O}(|\Delta| \cdot |Q|)$.

For each limit transition $P \rightarrow q$ or $q \rightarrow P$, the algorithm needs to check the presence of a path $p \xrightarrow{P} p$ in $\mathcal{A}_{|P|}$, where p is an arbitrary state in P . This can be done by means of a depth-first search in $\mathcal{A}_{|P|}$ starting from p and following only transitions labelled by a subset of P . If this search reaches p after having followed transitions whose union of labels is exactly P , then the limit transition can be followed by some run. In the worst case, it is necessary to explore all the transitions of $\mathcal{A}_{|P|}$. Assuming that all the necessary manipulations of subsets of states (checking the inclusion of a subset in another, computing the union of two subsets or their difference) run in $\mathcal{O}(|Q|)$, the time complexity of the emptiness test is thus $\mathcal{O}(|\Delta| \cdot |\Delta_{|P|}| \cdot |Q|)$, that is, $\mathcal{O}(|\Delta|^2 \cdot |Q|^2)$. $\square$

Note that the exploration of the final reachability automaton has a cost linear in its size, i.e., $\mathcal{O}(|\Delta| \cdot |Q|)$. Exploring this automaton thus has a negligible cost with respect to building it.

3.3 Words Indexed by any Linear Ordering

An extension of the previous algorithm is proposed in [7] to deal with words indexed by any linear ordering. It is similar to the one presented in the scattered case, except for an additional rule that deals with density. It checks whether each limit set of cardinality i is strongly connected in $\mathcal{A}_{i-1}$, as previously, but it also checks whether it forms a *shuffle structure*, such as the automaton $\mathcal{B}$ in Fig. 1.b. This is the mechanism that enables automata on linear orderings to read words of infinite dense length.

The transition relation of each automaton $\mathcal{A}_i$ is now defined as

$$\begin{aligned} \Delta_0 &= \{(p, \{p, q\}, q) \mid (p, a, q) \in \Delta\} \\ \Delta_i &= \Delta_{i-1} \cup \Delta_i^L \cup \Delta_i^R \cup \Delta_i^S \quad \text{for } 0 < i \leq M \end{aligned}$$

where Δ_i^L and Δ_i^R are defined as previously and Δ_i^S is the following:

$$\Delta_i^S = \{(p, P \cup \{p, q\}, q) \mid p \rightarrow P, P \rightarrow q \in \Delta \text{ and } |P| = i \text{ and } C^S\}.$$

The condition C^S checks the presence of a shuffle structure, by imposing the existence of $n > 0$ paths $p_1 \xrightarrow{P_1} q_1, \dots, p_n \xrightarrow{P_n} q_n$ in $\mathcal{A}_{i-1}$ and $m \geq 0$ states $r_1, \dots, r_m$ such that:

$$\begin{aligned} q_i &\rightarrow P \rightarrow p_i \in \Delta \text{ for each } 0 < i \leq n \\ r_i &\rightarrow P \rightarrow r_i \in \Delta \text{ for each } 0 < i \leq m \\ \bigcup_{0 < i \leq n} P_i \cup \{r_1, \dots, r_m\} &= P \end{aligned}$$

Remark 1. This rule generates transitions witnessing shuffles of simpler languages (see [1] for a formal definition) that can be read in $\mathcal{A}$, from p_i to q_i , respectively. If these languages are the empty word, they form gaps in the shuffle and these gaps are mapped to states r_i . The existence of at least one path in $\mathcal{A}_{i-1}$ ensures that the shuffle is not composed of empty words only. If it were the case, there would only be one cut in the resulting ordering and the limit transition would not be enabled. For instance, adding the limit transitions $q_0, q_3 \rightarrow \{q_3\} \rightarrow q_3, q_4$ in the automaton $\mathcal{B}$ from Fig. 1.b would not change its accepted language. Indeed, assume that there exists a run ρ reading a word w of length J in this automaton, with $\rho(\hat{J}_{min}) = q_0$, $\rho(\hat{J}_{max}) = q_4$, and $\rho(c) = q_3$ for all the cuts c different from the first and the last one. Then it is clear that ρ cannot follow any successor transition. Hence w needs to be equal to the empty word. This is a contradiction, given that $q_0 \neq q_4$.

Example 4. We illustrate the algorithm on the automaton $\mathcal{B}$ in Fig. 1.b. First, the reachability automaton $\mathcal{B}_0$ is built by replacing the unique successor transition (q_1, a, q_2) of $\mathcal{B}$ by the transition $(q_1, \{q_1, q_2\}, q_2)$. Automata $\mathcal{B}_1$ and $\mathcal{B}_2$ are equal to $\mathcal{B}_0$ since the only limit-set cardinality is 3. In the automaton $\mathcal{B}_2$, the path $q_1 \xrightarrow{\{q_1, q_2\}} q_2$ exists and the set $\{q_1, q_2\} \cup \{q_3\}$ is exactly equal to the limit set $P = \{q_1, q_2, q_3\}$. Moreover, the limit transitions $q_2 \rightarrow P \rightarrow q_1$ and $q_3 \rightarrow P \rightarrow q_3$ belong to the transition relation of $\mathcal{B}$. Therefore, for every pair of limit transitions $p \rightarrow P \rightarrow q$, the transition $(p, P \cup \{p, q\}, q)$ is added to $\mathcal{B}_3$. The result is shown in Fig. 3, where dashed transitions are those added to $\mathcal{B}_3$, regardless of their label. There exists a path from q_0 to q_4 in the reachability automaton $\mathcal{B}_3$, hence $\mathcal{B}$ accepts at least one word.

Lemma 2. *The emptiness for words indexed by any linear ordering can be tested in time $\mathcal{O}(|\Delta|^2 \cdot |Q|^2)$ and space $\mathcal{O}(|\Delta| \cdot |Q|)$.*

Proof. The arguments are similar to those of Lemma 1. The only difference is the presence of additional transitions in the reachability automata, to take into account the potential shuffle structures. For each limit transition $q \rightarrow P$ or $P \rightarrow q$, at most $\mathcal{O}(|Q|)$ such transitions are added to $\Delta_{|P|}^S$, hence each reachability automaton still has $\mathcal{O}(|\Delta| \cdot |Q|)$ transitions.

Notice that checking that the limit transitions involving a limit set P form a shuffle structure (i.e., checking the condition C^S) can be done in a time

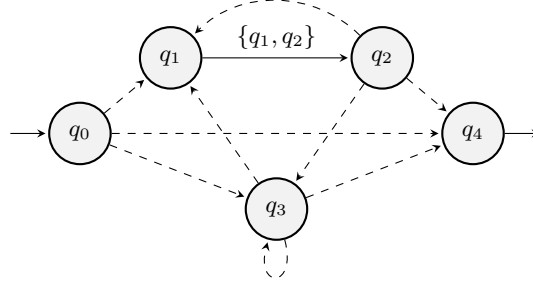


Fig. 3: Illustration of the test for any ordering.

$\mathcal{O}(|\Delta_{|P|}| \cdot |Q|)$, as for checking the presence of a path $p \xrightarrow{P} p$ in Lemma 1. Indeed, as already pointed out in [7] (in Lemma 26), checking the condition C^S amounts to

1. keeping only the transitions of $\mathcal{A}_{|P|}$ whose labels are included in P ,
2. keeping only the states of $\mathcal{A}_{|P|}$ that are reachable from some state p such that $P \rightarrow p \in \Delta$, and co-reachable from some state q such that $q \rightarrow P \in \Delta$,
3. and checking that the union of the labels of the transitions that remain is exactly P .

Assuming, as in Lemma 1, that all the necessary manipulations of subsets of states run in $\mathcal{O}(|Q|)$, this yields an algorithm running in $\mathcal{O}(|\Delta_{|P|}| \cdot |Q|)$.

The overall time complexity of the emptiness test is thus $\mathcal{O}(|\Delta| \cdot |\Delta_{|P|}| \cdot |Q|)$, that is, $\mathcal{O}(|\Delta|^2 \cdot |Q|^2)$. $\square$

4 Efficient Test of Emptiness

4.1 Preliminaries

The algorithms testing the emptiness for automata on linear orderings presented in the previous section convert each enabled limit transition into $\mathcal{O}(|Q|)$ transitions.

We introduce a new algorithm, inspired by the previous ones, that converts each enabled limit transition into a single transition. This is achieved by selecting a specific state in the limit set associated with the enabled transition, and using it as a representative. Intuitively, the limit set must form some loop to enable its limit transition. Therefore, choosing one member of this loop is enough to connect its content to the origin or destination state.

Importantly, there exist refined emptiness tests in the literature, such as those checking the existence of at least one accepted word indexed by $\mathbb{R}$ or $\mathbb{Q}$ [2]. These tests follow the same principles as those presented in Section 3, and they check additional conditions (closely related to specific properties of $\mathbb{R}$ and $\mathbb{Q}$). The algorithm introduced below can be adapted to these tests with the same gain in complexity.

4.2 Algorithm

In this section, we present an efficient algorithm testing the emptiness for words indexed by any linear ordering. We first introduce an arbitrary total order $<_Q$ on the set of states Q and define a function $\min : 2^Q \setminus \{\emptyset\} \rightarrow Q$ that returns the least element of any non-empty subset of Q , according to this total order.

The transition relation Δ_i of each reachability automaton $\mathcal{A}_i$ is recursively defined as follows:

$$\begin{aligned}\Delta_0 &= \{(p, \{p, q\}, q) \mid (p, a, q) \in \Delta\} \\ \Delta_i &= \Delta_{i-1} \cup \Delta_i^L \cup \Delta_i^R \\ \Delta_i^L &= \{(p, P \cup \{q\}, q) \mid P \rightarrow q \in \Delta \text{ and } |P| = i \text{ and } p = \min(P) \text{ and } C\} \\ \Delta_i^R &= \{(q, P \cup \{q\}, p) \mid q \rightarrow P \in \Delta \text{ and } |P| = i \text{ and } p = \min(P) \text{ and } C\}\end{aligned}$$

The condition C is defined as:

- there exists a path $p \xrightarrow{P} p$ in $\mathcal{A}_{i-1}$, or
- there exist $n > 0$ paths $p_1 \xrightarrow{P_1} q_1, \dots, p_n \xrightarrow{P_n} q_n$ in $\mathcal{A}_{i-1}$ and $m \geq 0$ states $r_1, \dots, r_m$ such that:

$$\begin{aligned}q_i &\rightarrow P \rightarrow p_i \in \Delta \text{ for each } 0 < i \leq n \\ r_i &\rightarrow P \rightarrow r_i \in \Delta \text{ for each } 0 < i \leq m \\ \bigcup_{0 < i \leq n} P_i \cup \{r_1, \dots, r_m\} &= P\end{aligned}$$

Intuitively, for each enabled limit transition, the algorithm now chooses one specific state of the limit set to generate only one transition in the reachability automaton. When the condition to enable a limit transition is the existence of a cycle from that state to itself, there is no need to add more than one transition to obtain the desired reachability properties. When the condition is the existence of a shuffle structure, picking a single state remains sufficient, provided that each new transition is connected to it. This improves time and space complexities by one power, with respect to previously existing algorithms. Notice that the case of a shuffle structure is covered by the transition relations Δ_i^L and Δ_i^R . It is not necessary to handle this case separately. Therefore there is no need for a relation Δ_i^S , even though this algorithm checks emptiness for words indexed by any linear ordering.

Example 5. We apply the algorithm to the automaton $\mathcal{A}$ from Fig. 1.a. We decide to order its states according to their index. $\mathcal{A}_0$ is built by relabelling the successor transitions of $\mathcal{A}$ and is equal to $\mathcal{A}_1$ since there is no limit transition of cardinality 1. Then, the path $q_1 \xrightarrow{\{q_1, q_2\}} q_1$ exists in $\mathcal{A}_1$, hence the limit transitions $q_0 \rightarrow \{q_1, q_2\} \rightarrow q_0$ are converted into the transitions $(q_0, \{q_0, q_1, q_2\}, q_1)$ and $(q_1, \{q_0, q_1, q_2\}, q_0)$, since we have decided that $q_1 <_Q q_2$. This is depicted in Fig. 4.a. The transitions that were added at the previous step create the path

$q_0 \xrightarrow{\{q_0, q_1, q_2\}} q_0$ in $\mathcal{A}_2$. The existence of this path shows that the limit transition $\{q_0, q_1, q_2\} \rightarrow q_3$ is enabled. It is thus converted into $(q_0, \{q_0, q_1, q_2, q_3\}, q_3)$. The result is shown in Fig. 4.b. There exists a path from q_0 to q_3 , which shows that $\mathcal{A}$ accepts a non-empty language.

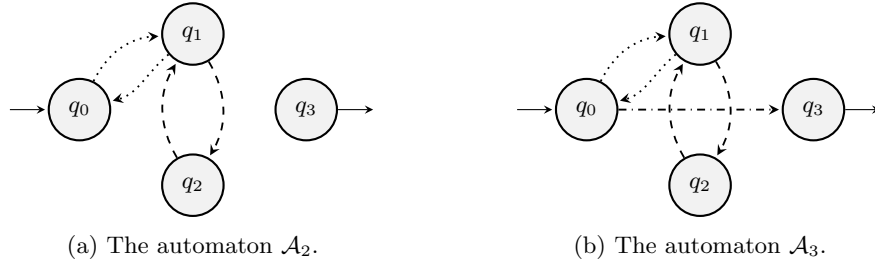


Fig. 4: Illustration of the efficient test on the automaton $\mathcal{A}$.

Example 6. We apply the efficient emptiness test to the automaton $\mathcal{B}$ from Fig. 1.b. The automata $\mathcal{B}_0, \dots, \mathcal{B}_2$ and the criteria to determine that every limit transition is enabled are the same as in Example 4. The difference is about the transitions added to $\mathcal{B}_3$ (Fig. 5). For this, we first choose q_1 as representative in the limit set $P = \{q_1, q_2, q_3\}$. Left-limit transitions $P \rightarrow q$ are converted into transitions that start in q_1 and end in q . Symmetrically, right-limit transitions $q \rightarrow P$ are converted into transitions that start in q and end in q_1 . The whole limit set forms a strongly connected component in $\mathcal{B}_3$, because there exist transitions from q_1 to the origin of the path $q_1 \xrightarrow{\{q_1, q_2\}} q_2$, from q_1 to q_3 , from the destination of the path $q_1 \xrightarrow{\{q_1, q_2\}} q_2$ to q_1 and from q_3 to q_1 . The state q_0 is connected to q_1 , which is itself connected to q_4 . Hence there exists a path between q_0 and q_4 , which shows that $\mathcal{B}$ accepts a non-empty language.

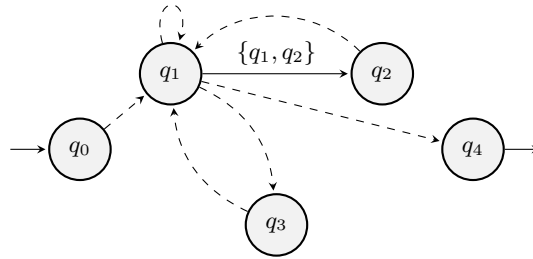


Fig. 5: Illustration of the efficient test on the automaton $\mathcal{B}$.

Lemma 3. *The emptiness for words indexed by (scattered) linear orderings can be tested in time $\mathcal{O}(|Q| \cdot |\Delta|^2)$ and space $\mathcal{O}(|\Delta|)$.*

Proof. Each enabled limit transition induces one transition in the reachability automaton, therefore $|\Delta_i|$ scales as $|\Delta|$. This is the only difference with respect to Lemmas 1 and 2, where $|\Delta_i|$ scales as $|\Delta| \cdot |Q|$. The remainder of the analysis is unchanged. Removing the second item in Condition C , restricts the test to scattered orderings without affecting its complexity. The correctness of the construction is given by Theorem 1. $\square$

We start by showing that this algorithm adds all the necessary transitions in each reachability automaton.

Lemma 4. *If a limit transition $P \rightarrow q$ of cardinality i (resp. $q \rightarrow P$) is enabled in $\mathcal{A}$, the set of paths $\{p \xrightarrow{P \cup \{q\}} q \mid p \in P\}$ exists in $\mathcal{A}_i$ (resp. $\{q \xrightarrow{P \cup \{q\}} p \mid p \in P\}$).*

Proof. The proof is by induction on i . If $i = 0$, a limit transition with an empty limit set can never be enabled and the lemma is correct. The interesting case is the induction step. If a limit transition $P \rightarrow q$ of cardinality $i > 0$ can be followed in $\mathcal{A}$ ($q \rightarrow P$ is symmetric and not discussed here), there are two cases:

1. There exists a cycle in $\mathcal{A}$ that visits exactly P and involves limit transitions of cardinality at most equal to $i-1$ and successor transitions that start in a state of P and end in a state of P . By induction hypothesis, any left-limit transition of cardinality at most $i-1$ is encoded in $\mathcal{A}_{i-1}$ by the presence of paths that join each state of its limit set to its destination. Symmetrically, for any right-limit transition of cardinality at most $i-1$, there exist paths in $\mathcal{A}_{i-1}$ from its origin to each state of its limit set. Moreover, every successor transition of $\mathcal{A}$ corresponds to a transition in $\mathcal{A}_i$, for every i . Thus in $\mathcal{A}_{i-1}$, there exists the path $p \xrightarrow{P} p$ for any $p \in P$. In particular, this path can be composed of the two paths $p_{\min} \xrightarrow{P_1} p$ and $p \xrightarrow{P_2} p_{\min}$, with $p_{\min} = \min(P)$ and $P_1 \cup P_2 = P$. Since the transition $(p_{\min}, P \cup \{q\}, q)$ belongs to Δ_i , the path $p \xrightarrow{P \cup \{q\}} q$ can be built in $\mathcal{A}_i$ by the concatenation of the path $p \xrightarrow{P_2} p_{\min}$ with the path $p_{\min} \xrightarrow{P \cup \{q\}} q$, formed by the freshly added transition.
2. There exists a shuffle structure in P . Namely, there exist limit transitions $q_1, \dots, q_n \rightarrow P \rightarrow p_1, \dots, p_n$ such that the state q_j is reachable from the state p_j , by visiting a subset P_j of P via successor transitions and limit transitions whose limit set is strictly included in P .

Now, we build the path $p \xrightarrow{P \cup \{q\}} q$, for any $p \in P$. It is composed of the concatenation of two or three paths:

- $p \xrightarrow{P} p_{\min}$ and $p_{\min} \xrightarrow{P \cup \{q\}} q$, if $p \rightarrow P \in \Delta$,
- $p \xrightarrow{P'_k} q_k$, $q_k \xrightarrow{P} p_{\min}$, and $p_{\min} \xrightarrow{P \cup \{q\}} q$ otherwise, for some $0 < k \leq n$ and where P'_k is a subset of P_k containing p . This is illustrated in Fig. 6.

The existence of the path $p \xrightarrow{P'_k} q_k$ is guaranteed by induction hypothesis

and Δ_0 , since it is a portion of the path $p_k \xrightarrow{P_k} q_k$ that is necessary to enable the limit transition through a shuffle.

The paths $p \xrightarrow{P} p_{min}$ and $p_{min} \xrightarrow{P \cup \{q\}} q$ exist, since they are both composed of a transition added to Δ_i at step i .

Cases 1 and 2 are exhaustive, as already shown in Lemma 24 of [7]. $\square$

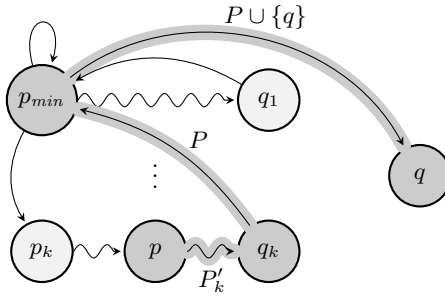


Fig. 6: Construction of the path $p \xrightarrow{P \cup \{q\}} q$.

Theorem 1. *In the reachability automaton $\mathcal{A}_i$, there exists a path $p \xrightarrow{P} q$ if and only if there exists a word w that can be read in $\mathcal{A}$, from p to q , by visiting exactly the set of states P , and by following limit transitions of cardinality at most equal to i .*

Proof. We first show that if there exists a path in $\mathcal{A}_i$, then w exists, by induction on i . When $i = 0$, only finite words can be read by $\mathcal{A}$, and $\mathcal{A}_0$ inherits the successor transition relation of $\mathcal{A}$. For the induction step, we only need to show that a word w can be associated with the path $p \xrightarrow{P \cup \{p, q\}} q$, composed of a single transition added at step i , resulting from the left-limit transition $P \rightarrow q$ (the case $q \rightarrow P$ is symmetric). There are two cases to consider:

1. This transition results from the existence of a path $p \xrightarrow{P} p$ in $\mathcal{A}_{i-1}$. By induction hypothesis, this path can be associated with a word u , that can be read in $\mathcal{A}$ from p to itself, by visiting exactly P . The word $w = u^\omega$ can be read in $\mathcal{A}$ from p to q , by visiting exactly $P \cup \{p, q\}$. Indeed, since u can be read from p to itself, by visiting exactly P and $P \rightarrow q$ is a left-limit transition, u can be read infinitely often to eventually reach q .

2. This transition results from the existence of $n > 0$ paths $p_1 \xrightarrow{P_1} q_1, \dots, p_n \xrightarrow{P_n} q_n$ in $\mathcal{A}_{i-1}$ and $m \geq 0$ states $r_1, \dots, r_m$ such that:

$$\begin{aligned} q_j &\rightarrow P \rightarrow p_j \in \Delta && \text{for } 0 < j \leq n \\ r_j &\rightarrow P \rightarrow r_j \in \Delta && \text{for } 0 < j \leq m \\ \bigcup_{0 < j \leq n} P_j \cup \{r_1, \dots, r_m\} &= P \\ p &= \min(P) \end{aligned}$$

By induction hypothesis, there exist words $u_1, \dots, u_n$ such that each u_j can be read in $\mathcal{A}$ from p_j to q_j , by visiting exactly P_j . Moreover, the empty word can be read from each r_j to itself, by visiting $\{r_j\}$. Let $0 < k \leq n$ be such that $p \in P_k$. There are two cases: either there exists a path in $\mathcal{A}_{i-1}$ going from p to q_k , or $p = q_k$. In the first case, there exists a non-empty word u'_k that can be read in $\mathcal{A}$ from p to q_k , by induction hypothesis. In the second one, the empty word $u'_k = \varepsilon$ can be read in $\mathcal{A}$ from p to $q_k = p$.

We are now ready to build the word w . Consider the partition of $\mathbb{R}$ into $\ell = n + m$ subsets $R_1, \dots, R_\ell$, such that between any two elements of $\mathbb{R}$, there exists an element of R_j , for any $0 < j \leq \ell$ (i.e., these subsets are dense in $\mathbb{R}$). We first build a word v by replacing each element of the subset R_j by a copy of the word u_j , for $0 < j \leq n$, and we erase each element of R_j for $n < j \leq \ell$. The word w is obtained by concatenating u'_k and v , i.e., we define $w = u'_k \cdot v$.

Let us show that w can be read in $\mathcal{A}$ from p to q , while visiting exactly $P \cup \{p, q\}$. First, u'_k can be read from p to q_k , by visiting some set of states $P'_k \subseteq P_k$ that contains p . Then, v can be read from q_k to q , by visiting $P \cup \{q\}$. Indeed, consider a cut $c \in \widehat{|w|}$ that coincides with the first cut of some u_j . This cut does not have any predecessor and is not the first one in $\widehat{|w|}$, since $\mathbb{R}$ is dense and does not have a least element. Moreover, to the left of this cut, arbitrarily close to it, there are infinitely many occurrences of each state in P . Indeed, the subsets R_j defined above are dense in $\mathbb{R}$, and any state r_j can read the empty word (therefore, gaps can be mapped to the states r_j). Therefore, the transition $P \rightarrow p_j$ is enabled. Moreover u_j can be read from p_j to q_j by visiting exactly P_j , by induction hypothesis. The same arguments show that q_j can be left by the limit transition $q_j \rightarrow P$. This analysis holds for all the limit transitions $q_j \rightarrow P \rightarrow p_j$ and $r_j \rightarrow P \rightarrow r_j$. It can also be applied to show that the state q can be eventually reached by following the transition $P \rightarrow q$. The word v can thus be read in $\mathcal{A}$ from q_k to q .

Hence, one concludes that w can be read in $\mathcal{A}$, from p to q , by visiting exactly $P \cup \{p, q\}$, and following limit transitions of cardinality at most equal to i .

For the other direction, we consider a word w and a run ρ of $\mathcal{A}$ reading it from a state p to a state q , visiting exactly the set of states P , and following limit transitions of cardinality at most equal to i . We show that there exists a path

$p \xrightarrow{P \cup \{p, q\}} q$, in $\mathcal{A}_i$. This is again done by induction on i . When $i = 0$, since $\mathcal{A}_0$ inherits the successor transition relation of $\mathcal{A}$, the property holds. Indeed, if the paths $p \xrightarrow{P_1} q'$ and $q' \xrightarrow{P_2} q$ exist, then the path $p \xrightarrow{P_1 \cup P_2} q$ exists as well. For the induction step, we use Lemma 4. By induction hypothesis, to runs reading words in $\mathcal{A}$ that follow limit transitions of cardinality less than i , correspond paths in $\mathcal{A}_{i-1}$ that share the same origin, destination, and states visited in between. Let $c \in \widehat{|w|}$ be a cut that is not the first one, that does not have a predecessor, and that ρ reaches by following a limit transition $P \rightarrow q$, with $|P| = i$. Since ρ uses this limit transition, it is enabled (either by an infinite repetition or by a shuffle structure). Hence, in $\mathcal{A}_i$, there exist paths from every state $p \in P$ to q , labelled by $P \cup \{q\}$. The same holds for right-limit transitions, and thus there exists the path $\rho(\widehat{|w|}_{min}) \xrightarrow{\{\rho(c) | c \in \widehat{|w|}\}} \rho(\widehat{|w|}_{max})$ in $\mathcal{A}_i$. $\square$

5 Conclusions and Perspectives

The contribution of this paper is to provide a family of efficient emptiness tests for automata on linear orderings. The gain in complexity with respect to previously existing algorithms comes from the choice of a specific representative state per limit transition.

Automata on linear orderings can serve as data structures to represent the models of formulas belonging to some logical fragments, for instance the monadic first-order theory of order on $\mathbb{R}$ or $\mathbb{Q}$ [2]. These decidable fragments might be of practical interest in verification or theorem proving, hence the need for decision procedures that are efficient in practice. In this context, checking that an automaton accepts at least one word is a crucial problem, since it amounts to checking the satisfiability of a logical formula. Therefore, efficient emptiness tests for automata on linear orderings are an important step towards the implementation of new practically usable decision procedures. The next task is to implement and benchmark these tests.

Acknowledgments. Thomas Braipson has been granted with a FRIA grant and Tom Clara is a Research Fellow of the F.R.S.-FNRS.

References

1. Bès, A., Carton, O.: A Kleene theorem for languages of words indexed by linear orderings. *International Journal of Foundations of Computer Science* **17**(03), 519–541 (2006)
2. Boigelot, B., Fontaine, P., Vergain, B.: Non-emptiness test for automata over words indexed by the reals and rationals. In: *Proc. 28th International Conference on Implementation and Application of Automata (CIAA)*. *Lecture Notes in Computer Science*, vol. 15015, pp. 94–108. Springer (2024)
3. Bruyère, V., Carton, O.: Automata on linear orderings. *Journal of Computer and System Sciences* **73**(1), 1–24 (2007)

4. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Proc. International Congress on Logic, Methodology and Philosophy of Science. pp. 1–12. Stanford University Press (1962)
5. Carton, O.: Accessibility in automata on scattered linear orderings. In: Mathematical Foundations of Computer Science 2002. pp. 155–164. Springer Berlin Heidelberg (2002)
6. Cristau, J.: Automata and temporal logic over arbitrary linear time. In: Proc. IARCS 29th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). LIPIcs, vol. 4, pp. 133–144 (2009)
7. Cristau, J.: Jeux et automates sur les ordres. Ph.D. thesis, Université Paris-Diderot - Paris VII (Dec 2010)
8. Rabinovich, A.: Temporal logics over linear time domains are in PSPACE. *Information and Computation* **210**, 40–67 (2012)
9. Rosenstein, J.G.: Linear orderings. Academic press (1982)