Epsilon Automata on Linear Orderings

Bernard Boigelot[0009−0009−4721−3824] ✉, Thomas Braipson[0009−0003−0543−5265], and Tom Clara[0009−0009−2938−1410]

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

Abstract. Automata on linear orderings are a form of finite-state automata introduced by Bruyère and Carton, that generalize the concepts of finite-word, infinite-word, and transfinite-word automata. They recognize words indexed by linear orderings, defined as mappings from the elements of such orderings to a finite alphabet. Some theoretical developments involving automata on linear orderings are hindered by the fact that their definition does not allow epsilon transitions, i.e., transitions with an empty label. In this paper, we define a variant of automata on linear orderings that allows such transitions in their transition relation, and show that this extension does not modify the expressive power of the automata. We motivate the usefulness of epsilon automata on linear orderings by using them for correcting an erroneous construction in the original proof of equivalence between automata on linear orderings and the corresponding notion of regular expressions.

Keywords: Finite automata · Linear orderings · Epsilon transitions.

1 Introduction

Finite automata are mathematical objects that are highly expressive and yet easy to handle algorithmically. For those reasons, they provide powerful theoretical tools for establishing the decidability of various logics. For instance, a simple and elegant way of deciding Presburger arithmetic, i.e., the first-order additive theory of integer numbers, consists in building a finite-word automaton recognizing the models of a given formula, and then checking whether this automaton accepts a non-empty language [3]. In 1962, Büchi introduced infinite-word automata in order to extend this approach to logics such as the second-order monadic theory of one successor (S1S) [5]. These automata have then been generalized into automata on bi-infinite words [8] and automata on transfinite words [6,12].

Automata on linear orderings are a more general class of automata defined by Bruyère and Carton, aimed at unifying and extending finite-word, (bi-)infinite-word, and transfinite-word automata [4]. Those automata recognize words indexed by linear orderings, defined as mappings from the elements of an arbitrary totally ordered set to a finite alphabet. Their expressive power has been precisely characterized, by establishing that the languages that they accept are

exactly those that can be described by an extension of the notion of regular expression [4,1]. It is however worth mentioning that the properties of automata on linear orderings do not straightforwardly generalize those of finite-word and infinite-word automata. For instance, language complementation is only feasible with such automata if some restrictions are imposed on the orderings that are considered [10].

Automata on linear orderings have mainly been introduced as theoretical tools, although there are projects to use them as actual data structures [2]. They have been useful in particular to design decision procedures for temporal logics over continuous time [7,9]. For theoretical developments, one missing feature of those automata is that their transition relation cannot contain epsilon transitions, i.e., transitions with an empty label. This needlessly complicates some constructions, in particular those required by the proof of equivalence between automata on linear orderings and regular expressions [4,1]. For instance, showing that the concatenation of two languages accepted by automata on linear orderings is itself recognizable by such automata needs to find a way of connecting every final state of the first automaton with every initial state of the second one by combinations of transitions. With epsilon transitions, this operation becomes immediate. The main motivation behind this work actually comes from the discovery of a (minor) error in [1] that, in our opinion, results from a construction made unnecessarily difficult by the lack of epsilon transitions.

The contribution of this work is to introduce a variant of automata on linear orderings that allows epsilon transitions, which we call epsilon automata on linear orderings. We show that, with the appropriate choice of semantics, this adaptation does not modify the expressive power of automata, and provide an algorithm that removes the epsilon transitions from a given input automaton without affecting its accepted language. We then demonstrate the usefulness of epsilon automata on linear orderings by correcting the erroneous construction of [1] with the help of epsilon transitions.

2 Basic Notions

2.1 Orderings and Cuts

We recall elementary notions on orderings and cuts, and refer to [11] for further details. A *linear ordering* is a set J associated with a binary relation $<_J$ that is complete (for every $j \neq k \in J$, either $j <_J k$ or $k <_J j$), irreflexive (for every $j \in J$, $j \not<_J j$), antisymmetric (for every $j, k \in J$, $j <_J k$ implies $k \not<_J j$) and transitive (for every $j, k, \ell \in J$, $j <_J k$ and $k <_J \ell$ imply $j <_J \ell$). When the ordering relation is clear from the context, we simply write $<$ in place of $<_J$. Two elements $j < k \in J$ are said to be *consecutive* if there does not exist $\ell \in J$ such that $j < \ell < k$. In such a case, k is the *successor* of j, and j the *predecessor* of k. An ordering J is *dense* if for every $j < k \in J$, there exists ℓ such that $j < \ell < k$.

A *cut* of a linear ordering J is a partition, written (K, L), of J into two sets K and L such that for every $k \in K$ and $\ell \in L$, one has $k < \ell$. The set of cuts

of J is denoted by $\widehat{J}$. This set contains a *first cut* $\widehat{J}_{min} = (\emptyset, J)$ and a *last cut* $\widehat{J}_{max} = (J, \emptyset)$. The set $\widehat{J} \setminus \{\widehat{J}_{min}, \widehat{J}_{max}\}$ is denoted by $\widehat{J}^*$. A cut $(K, L) \in \widehat{J}^*$ is called a *gap* if K does not have a greatest element and L does not have a least element. If J does not have any gap, then it is said to be *complete*.

The set $\widehat{J}$ can itself be turned into a linear ordering by equipping it with the relation $<_{\widehat{J}}$ such that $(K_1, L_1) <_{\widehat{J}} (K_2, L_2)$ iff $K_1 \subsetneq K_2$ (or equivalently $L_2 \subsetneq L_1$). Note that J and $\widehat{J}$ may have different cardinalities, for instance, the cuts of the set $\mathbb{Q}$ of rational numbers form a set that is uncountable. The set $J \cup \widehat{J}$ that combines both the elements of J and its cuts also forms a linear ordering, by considering that the subsets J and $\widehat{J}$ keep their internal ordering, i.e., for all $j_1, j_2 \in J$ and $c_1, c_2 \in \widehat{J}$, one has $j_1 < j_2$ iff $j_1 <_J j_2$ and $c_1 < c_2$ iff $c_1 <_{\widehat{J}} c_2$, and that for $j \in J$ and $(K, L) \in \widehat{J}$, one has $j < (K, L)$ iff $j \in K$, and $(K, L) < j$ iff $j \in L$.

2.2 Words and Automata on Linear Orderings

We now summarize some definitions borrowed from [4]. A *word indexed by a linear ordering* is a mapping $w : J \to \Sigma$ from a linear ordering J to a finite alphabet Σ. The ordering J is called the *length* of w; this is denoted by $|w| = J$. Note that choosing J to be equal to a finite set, the set of natural numbers $\mathbb{N}$, or the set of integer numbers $\mathbb{Z}$, leads to the familiar notions of finite, infinite, and bi-infinite words. The *empty word* ε is the only word indexed by the empty set, i.e., such that $|\varepsilon| = \emptyset$.

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

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

The transition relation Δ contains three types of transitions. Those of the form (q_1, a, q_2), with $q_1, q_2 \in Q$ and $a \in \Sigma$ are called *successor transitions*. They are alternatively denoted by $q_1 \xrightarrow{a} q_2$. Transitions (q, P) with $q \in Q$ and $P \subseteq Q$ are *right-limit transitions*; they can be written $q \to P$. Similarly, transitions (P, q) with $P \subseteq Q$ and $q \in Q$ are *left-limit transitions*, written $P \to q$.

The semantics of automata on linear orderings is defined as follows.

Definition 2. *A* run *of* $\mathcal{A}$ *reading a word* w *is a mapping* $\rho : \widehat{J} \to Q$*, where* $J = |w|$*, that satisfies the following conditions:*

- $\rho(\widehat{J}_{min}) \in I$*.*
- $\rho(\widehat{J}_{max}) \in F$*.*
- *For every consecutive* $c_1 = (K_1, L_1)$*,* $c_2 = (K_2, L_2) \in \widehat{J}$*, i.e., such that* $K_2 = K_1 \cup \{j\}$ *for some* $j \in J$*, one has* $(\rho(c_1), w(j), \rho(c_2)) \in \Delta$*.*

- *For every* $c \neq \widehat{J}_{min} \in \widehat{J}$ *that does not have a predecessor, one has* $\lim_{c^-} \rho \to \rho(c) \in \Delta$, *where* $\lim_{c^-} \rho = \{q \in Q \mid (\forall c_1 < c)(\exists c_2)(c_1 < c_2 < c \wedge \rho(c_2) = q)\}$.
- *For every* $c \neq \widehat{J}_{max} \in \widehat{J}$ *that does not have a successor, one has* $\rho(c) \to \lim_{c^+} \rho \in \Delta$, *where* $\lim_{c^+} \rho = \{q \in Q \mid (\forall c_1 > c)(\exists c_2)(c < c_2 < c_1 \wedge \rho(c_2) = q)\}$.

Intuitively, a run ρ must start in an initial state and end in a final state. Pairs of states corresponding to consecutive cuts must be linked by an instance of a successor transition reading the appropriate symbol. The *left limit* $\lim_{c^-} \rho$ of ρ at the cut c is the set of states P that are visited by ρ before c, arbitrarily close to c. Following a left-limit transition $P \to \rho(c)$ moves from such a set P to the state mapped to c. Similarly, the *right limit* $\lim_{c^+} \rho$ is the set of states P visited by ρ after c, arbitrarily close to c. A right-limit transition $\rho(c) \to P$ moves from the state mapped to c to the set P.

A word w indexed by a linear ordering is *accepted* by $\mathcal{A}$ if this automaton admits a run reading w. The class of all such words forms the *language accepted* by $\mathcal{A}$.

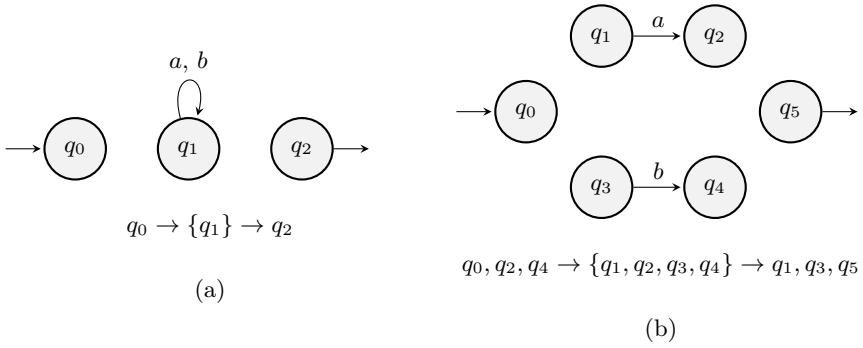

Fig. 1: Examples of automata on linear orderings.

We illustrate the concept of automata on linear orderings with the help of two examples. Initial and final states are respectively identified by incoming transitions without an origin, and outgoing transitions without a destination. For the sake of clarity, successor transitions sharing the same origin and destination are grouped together, as well as right-limit and left-limit transitions based on the same set of states. For instance, $p, q \to \{p, q\} \to p$ is shorthand for the transitions $p \to \{p, q\}$, $q \to \{p, q\}$ and $\{p, q\} \to p$. The automaton in Figure 1a accepts all bi-infinite words on the alphabet $\Sigma = \{a, b\}$, in other words all mappings from $\mathbb{Z}$ to Σ. The automaton in Figure 1b accepts all words $w : J \to \Sigma$ such that

- their length J is a non-empty complete set that does not contain any least or greatest element, and
- for every $j, k \in J$ such that $j < k$, there exist $\ell_a, \ell_b \in J$ such that $j < \ell_a < k$, $j < \ell_b < k$, $w(\ell_a) = a$, and $w(\ell_b) = b$.

An example of such a word can be constructed by setting $J = \mathbb{R}$, and for every $j \in \mathbb{R}$, $w(j) = a$ if $j \in \mathbb{Q}$ and $w(j) = b$ if $j \in \mathbb{R} \setminus \mathbb{Q}$. A run ρ reading this word is then obtained by defining

- $\rho((\emptyset, \mathbb{R})) = q_0$ and $\rho((\mathbb{R}, \emptyset)) = q_5$,
- for all $j \in \mathbb{Q}$, $\rho((\mathbb{R}_{<j}, \mathbb{R}_{\geq j})) = q_1$ and $\rho((\mathbb{R}_{\leq j}, \mathbb{R}_{>j})) = q_2$, where $\mathbb{R}_{\bowtie j}$ stands for $\{k \in \mathbb{R} \mid k \bowtie j\}$ with $\bowtie \in \{<, \leq, >, \geq\}$,
- for all $j \in \mathbb{R} \setminus \mathbb{Q}$, $\rho((\mathbb{R}_{<j}, \mathbb{R}_{\geq j})) = q_3$ and $\rho((\mathbb{R}_{\leq j}, \mathbb{R}_{>j})) = q_4$.

It is easy to see that this definition satisfies all the conditions of Definition 2.

2.3 Operators on Orderings and Words

Given two linear orderings J_1 and J_2, the ordering $J = J_1 + J_2$ is defined as the ordering obtained by placing the elements of J_1 before those of J_2, both sets keeping the internal order among their elements. Formally, J can be defined as the set $(J_1 \times \{1\}) \cup (J_2 \times \{2\})$ such that $(j, m) < (k, n)$ iff either $m <_{\mathbb{N}} n$, or $m = n$ and $j <_{J_m} k$. This leads to a natural definition for a concatenation operator on words: given two words w_1 and w_2 indexed by linear orderings, the word $w = w_1 \cdot w_2$ is such that $|w| = |w_1| + |w_2|$, and for every $j \in |w|$, one has $w(j) = w_1(j)$ if $j \in |w_1|$ and $w(j) = w_2(j)$ if $j \in |w_2|$. Note that we slightly abuse the notation by writing $|w_1|$ and $|w_2|$ for the two distinct parts of $|w|$ joined by the sum operator[1].

Concatenation can be seen as a particular case of a more general operator. Given linear orderings J and K_j for each $j \in J$, the ordering $J = \sum_{j \in J} K_j$ is obtained, schematically, by replacing every element j of J by the corresponding set K_j, keeping the internal order among the elements of this set. Formally, we have $\sum_{j \in J} K_j = \{(k, j) \mid j \in J \wedge k \in K_j\}$, with $(k_1, j_1) < (k_2, j_2)$ iff either $j_1 <_J j_2$, or $j_1 = j_2$ and $k_1 <_{K_{j_1}} k_2$. For a linear ordering J and words w_j for each $j \in J$, over the same alphabet, we define $\prod_{j \in J} w_j$ as the word w such that $|w| = \sum_{j \in J} |w_j|$, and for all $k \in |w|$, $w(k) = w_j(k)$ iff $k \in |w_j|$.

3 Epsilon Transitions

In this section, we motivate the need for epsilon transitions in automata on linear orderings, and introduce them with the help of an extended syntax and semantics.

3.1 Motivation

It has been established that the languages that can be accepted by automata on linear orderings correspond exactly to those that can be described by a generalization of the concept of regular expression. This result has first been proved

[1] A more rigorous definition involves the notion of *order type*, that we do not introduce in this paper for the sake of simplicity.

in [4] with a restriction to *countable and scattered* words, i.e., words w for which $|w|$ is a countable ordering that does not contain a dense infinite sub-ordering. It is then extended to unrestricted words in [1].

In this paper, we do not need to introduce the full formalism of regular expressions for words indexed by linear orderings, and we focus instead on a specific operator, defined as follows.

Definition 3. *Let L_1, L_2 be languages of words indexed by linear orderings, sharing the same alphabet Σ. The language $L = L_1 \diamond L_2$ contains all the words of the form $w = \prod_{j \in J \cup \widehat{J}^*} w_j$, where J is any non-empty linear ordering, and one has $w_j \in L_1$ for each $j \in J$ and $w_j \in L_2$ for each $j \in \widehat{J}^*$.*

Intuitively, $L_1 \diamond L_2$ is obtained by considering all non-empty linear orderings J, and replacing all their elements by words from L_1, and all their non-extreme cuts by words from L_2.

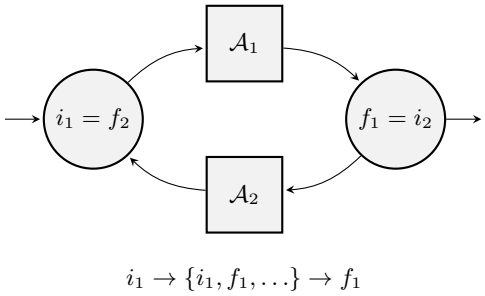

Fig. 2: Automaton accepting $L_1 \diamond L_2$.

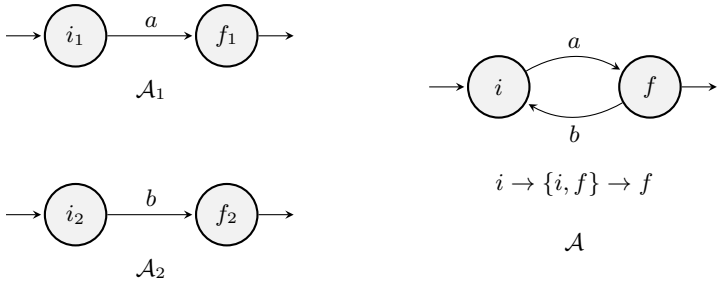

Fig. 3: Automata accepting $\{a\}$, $\{b\}$, and (erroneously) $\{a\} \diamond \{b\}$.

A construction building an automaton $\mathcal{A}$ accepting $L_1 \diamond L_2$ from automata $\mathcal{A}_1$ and $\mathcal{A}_2$ accepting respectively L_1 and L_2 is outlined in [4,1]. It consists in

first expressing each $\mathcal{A}_k$, for $k \in \{1, 2\}$, as a *normalized automaton*, which has a single initial state i_k and a single final state f_k such that $i_k \neq f_k$. In addition, i_k (resp. f_k) cannot be the destination (resp. the origin) of a successor or limit transition.

The next step is to combine $\mathcal{A}_1$ and $\mathcal{A}_2$ to form the automaton $\mathcal{A}$. The combination scheme is shown in Figure 2, in the particular case where neither $\mathcal{A}_1$ nor $\mathcal{A}_2$ accepts the empty word. In this construction, the limit transitions of the resulting automaton are those of $\mathcal{A}_1$ and $\mathcal{A}_2$, with additional transitions $i_1 \to P$ and $P \to f_1$ for all supersets P of $\{i_1, f_1\}$.

This construction is proved to be correct in [4], for the particular case of words restricted to be indexed by countable and scattered linear orderings. The claim made in [1] that it also applies to the general case is unfortunately invalid. Consider for instance the automata in Figure 3, accepting the languages $L_1 = \{a\}$ and $L_2 = \{b\}$. Let us show that the resulting automaton $\mathcal{A}$ accepts a language that differs from $L_1 \diamond L_2$. Indeed, $\mathcal{A}$ accepts in particular the word $w = \prod_{j \in \mathbb{R}} b$, i.e., the word indexed by $\mathbb{R}$ that only contains occurrences of the symbol b: a run ρ reading w can be obtained by setting $\rho((\emptyset, \mathbb{R})) = i$, $\rho((\mathbb{R}, \emptyset)) = f$, and for every $j \in \mathbb{R}$, $\rho((\mathbb{R}_{<j}, \mathbb{R}_{\geq j})) = f$ and $\rho((\mathbb{R}_{\leq j}, \mathbb{R}_{>j})) = i$. It is clear however that w does not belong to $\{a\} \diamond \{b\}$, since a occurs in every word of this language.

The error is caused by the fact that the construction in Figure 2 makes it possible to visit infinitely often all the states of $\mathcal{A}$ without ever following the successor transition labeled by a. This situation only happens when $\mathcal{A}_1$ admits a path from i_1 to f_1 that does not visit any other state.

A simple correction of this problem would thus consist in making sure that an internal state of $\mathcal{A}_1$ must necessarily be visited in order to enable the added limit transitions. However, this cannot be achieved with our current definition of automata on linear orderings, since it follows from Definitions 1 and 2 that any automaton accepting $\{a\}$ must contain a successor transition from an initial state to a final one, labeled by a. This motivates the need for a more general form of automaton on linear orderings, offering the possibility of defining internal states linked by transitions with an empty label. We introduce such automata in the next section. The correction of the construction of an automaton accepting $L_1 \diamond L_2$ is addressed in Section 4.

3.2 Epsilon Automata on Linear Orderings

We define our extended form of automata over linear orderings as follows.

Definition 4. *An* epsilon automaton on linear orderings *is a tuple* $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ *where the set of states* Q*, the alphabet* Σ*, the sets of initial and final states* I *and* F *are as in Definition 1, and the relation transition* Δ *is of the form* $\Delta \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times Q) \cup (Q \times 2^Q) \cup (2^Q \times Q)$.

Compared to Definition 1, the only difference is the possibility of having successor transitions (q_1, ε, q_2) with an empty label ε.

Designing an appropriate semantics for epsilon automata is not immediate. Following a successor transition labeled by ε should naturally make a run move

from a state to another without reading any symbol in its accepted word. The question is to decide whether following an infinite combination of epsilon transitions should enable limit transitions. In the positive case, adding the transition (q_1, ε, q_1) to the automaton in Figure 1a would make it also accept all finite words over the alphabet $\{a, b\}$. In the negative case, its accepted language would not be affected.

Our choice is to opt for the latter approach, that conservatively minimizes the impact of epsilon transitions on the semantics of automata. This leads to a semantics in which the states visited by a run reading a word w are still associated to the cuts of $|w|$, but with the property that following epsilon transitions does not modify the current cut.

Formally, given an epsilon automaton $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, we define the set $S \subseteq Q \times 2^Q \times Q$ of all the triples (q_1, U, q_2) such that $\mathcal{A}$ admits a finite path from q_1 to q_2 visiting exactly the states in U, entirely composed of successor transitions labeled by ε. In other words, such a path must be of the form $s_0 \xrightarrow{\varepsilon} s_1 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} s_n$ with $n \geq 0$, $s_0 = q_1$, $s_n = q_2$, $U = \{s_0, s_1, \ldots, s_n\}$, and for every $0 \leq k < n$, $(s_k, \varepsilon, s_{k+1}) \in \Delta$. For each $\sigma = (q_1, U, q_2) \in S$, we define $\mathit{first}(\sigma) = q_1$, $\mathit{last}(\sigma) = q_2$, and $\mathit{states}(\sigma) = U$.

Definition 5. *A* run *of $\mathcal{A}$ reading a word w is a mapping $\rho : \widehat{J} \to S$, where $J = |w|$, that satisfies the following conditions:*

- $\mathit{first}(\rho(\widehat{J}_{min})) \in I$.
- $\mathit{last}(\rho(\widehat{J}_{max})) \in F$.
- *For every $c_1 = (K_1, L_1)$, $c_2 = (K_2, L_2) \in \widehat{J}$ such that $K_2 = K_1 \cup \{j\}$ for some $j \in J$, one has $(\mathit{last}(\rho(c_1)), w(j), \mathit{first}(\rho(c_2))) \in \Delta$.*
- *For every $c \neq \widehat{J}_{min} \in \widehat{J}$ that does not have a predecessor, one has $\lim_{c^-} \rho \to \mathit{first}(\rho(c)) \in \Delta$, where $\lim_{c^-} \rho = \{q \in Q \mid (\forall c_1 < c)(\exists c_2)(c_1 < c_2 < c \wedge q \in \mathit{states}(\rho(c_2)))\}$.*
- *For every $c \neq \widehat{J}_{max} \in \widehat{J}$ that does not have a successor, one has $\mathit{last}(\rho(c)) \to \lim_{c^+} \rho \in \Delta$, where $\lim_{c^+} \rho = \{q \in Q \mid (\forall c_1 > c)(\exists c_2)(c < c_2 < c_1 \wedge q \in \mathit{states}(\rho(c_2)))\}$.*

Intuitively, instead of mapping the cuts of $|w|$ onto states of the automaton as in Definition 2, those cuts are now associated with finite paths of epsilon successor transitions, in which the only relevant information is their origin and destination states, together with the set of all the states that they visit.

An example of epsilon automaton on linear orderings is given in Figure 4. This automaton accepts the language $(ab)^\omega$, composed of a single word w mapping $\mathbb{N}$ onto $\{a, b\}$, such that $w(j) = a$ for even values of j, and $w(j) = b$ for odd j. Note that this automaton does not accept the empty word.

3.3 Expressiveness

We now establish that, with the choices made in Section 3.2, adding epsilon transitions to automata on linear orderings does not increase their expressive power.

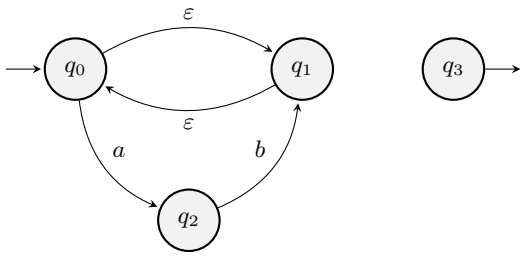

Fig. 4: Example of epsilon automaton on linear orderings.

Theorem 1. *A language of words indexed by linear orderings can be accepted by an epsilon automaton on linear orderings iff it can be accepted by an automaton on linear orderings.*

Proof. One direction is immediate: by Definitions 1 and 4, an automaton on linear orderings $\mathcal{A}$ can also be seen as an epsilon automaton $\mathcal{A}_\varepsilon$ without any epsilon transition. Every run ρ of $\mathcal{A}$ can be turned into a run of $\mathcal{A}_\varepsilon$ reading the same word w, by replacing $\rho(c)$ by $(\rho(c), \{\rho(c)\}, \rho(c))$ for each $c \in \widehat{|w|}$.

For the other direction, we prove a stronger result by providing an algorithm that turns an epsilon automaton $\mathcal{A}_\varepsilon$ into an equivalent automaton $\mathcal{A}$ on linear orderings. Let $\mathcal{A}_\varepsilon = (Q_\varepsilon, \Sigma_\varepsilon, \Delta_\varepsilon, I_\varepsilon, F_\varepsilon)$. The automaton $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ is built as follows:

- $Q \subseteq Q_\varepsilon \times 2^{Q_\varepsilon} \times Q_\varepsilon$ is the set of all triples (q_1, U, q_2) such that $\mathcal{A}_\varepsilon$ admits a finite path of epsilon successor transitions from q_1 to q_2, visiting exactly the states that belong to U. (This corresponds to the definition of the set S in Section 3.2).
- $\Sigma = \Sigma_\varepsilon$.
- Δ contains
  - all $(\sigma_1, a, \sigma_2) \in Q \times \Sigma \times Q$ such that $(\mathit{last}(\sigma_1), a, \mathit{first}(\sigma_2)) \in \Delta_\varepsilon$,
  - all $(P, \sigma) \in 2^Q \times Q$ such that $\big(\bigcup_{\sigma' \in P} \mathit{states}(\sigma'), \mathit{first}(\sigma)\big) \in \Delta_\varepsilon$,
  - all $(\sigma, P) \in Q \times 2^Q$ such that $\big(\mathit{last}(\sigma), \bigcup_{\sigma' \in P} \mathit{states}(\sigma')\big) \in \Delta_\varepsilon$.
- $I = \{\sigma \in Q \mid \mathit{first}(\sigma) \in I_\varepsilon\}$.
- $F = \{\sigma \in Q \mid \mathit{last}(\sigma) \in F_\varepsilon\}$.

It follows from Definition 5 that every run of $\mathcal{A}_\varepsilon$ describes a run of $\mathcal{A}$ reading the same word. □

The algorithm given in the proof of Theorem 1 can incur an exponential blowup in the number of states of the automaton, and a double exponential blowup in the number of limit transitions. This is not problematic, since we expect epsilon automata on linear orderings to be mainly useful for theoretical developments.

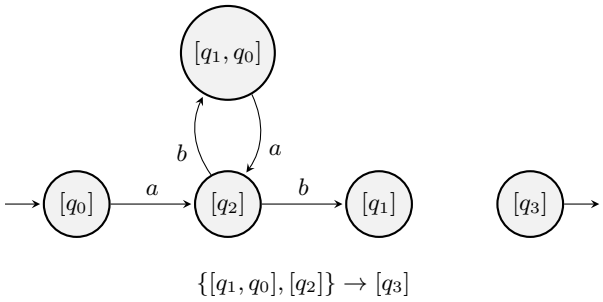

Fig. 5: Automaton after eliminating epsilon transitions.

We illustrate this result by showing in Figure 5 the construction of an automaton on linear orderings equivalent to the one in Figure 4. We use the notation $[q_i]$ as shorthand for $(q_i, \{q_i\}, q_i)$, for $i \in \{0, 1, 2, 3\}$, and write $[q_1, q_0]$ in place of $(q_1, \{q_0, q_1\}, q_0)$. To keep the picture simple, some states and transitions that cannot be visited or followed by any run are omitted.

4 Application

We are now ready to correct the construction of an automaton accepting $L_1 \diamond L_2$ with the help of epsilon automata. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be automata accepting respectively L_1 and L_2. Recall that automata on linear orderings as defined in [4,1] can be seen as particular cases of epsilon automata. We therefore assume w.l.o.g. that $\mathcal{A}_1$ and $\mathcal{A}_2$ are epsilon automata. Furthermore, we assume like in Section 3.1 that $\mathcal{A}_1$ and $\mathcal{A}_2$ are normalized, by extending to epsilon automata the same definition of normalization as for plain automata on linear orderings. Two remarks are in order. Firstly, epsilon transitions make it easy to normalize an arbitrary automaton: this operation amounts to replacing the initial states by a fresh state with outgoing epsilon transitions to them, and the final states by a fresh state with incoming epsilon transitions from them. Secondly, it is possible for a normalized epsilon automaton to accept the empty word: all that is needed is an epsilon transition linking its initial to its final states. This greatly simplifies the developments compared to [4], where languages that contain the empty word have to be handled as particular cases.

The construction of an automaton $\mathcal{A}$ accepting $L = L_1 \diamond L_2$ is shown in Figure 6. It consists in

- merging the initial state i_2 of $\mathcal{A}_2$ with the final state f_1 of $\mathcal{A}_1$ into a single state, which becomes the only final state of $\mathcal{A}$,
- making f_2 the only initial state of $\mathcal{A}$,
- adding an epsilon transition from f_2 to i_1,
- adding right-limit transitions from f_2 and left-limit transitions to i_2, for all sets of states containing i_1, i_2 and f_2.

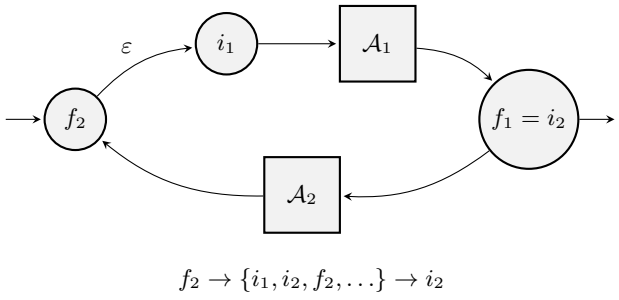

Fig. 6: Automaton accepting $L_1 \diamond L_2$.

Compared to the construction in Figure 2, the difference is that the presence of the epsilon transition (f_2, ε, i_1) now forces the state i_1, and therefore the branch associated to $\mathcal{A}_1$, to be visited in order to enable one of the added limit transitions. Let us show that this new construction is correct.

Theorem 2. *The automaton in Figure 6 accepts $L = L_1 \diamond L_2$.*

Proof. We first show that every word in $L_1 \diamond L_2$ is accepted by the constructed automaton $\mathcal{A}$. For any such word w, there exists a non-empty linear ordering J and words $w_j \in L_1$ for each $j \in J$ and $w_j \in L_2$ for each $j \in \widehat{J^*}$ such that $w = \prod_{j \in J \cup \widehat{J^*}} w_j$. For each $j \in J$ (resp. $j \in \widehat{J^*}$), let ρ_j be a run of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) that reads w_j. We build a run ρ of $\mathcal{A}$ reading w. This run maps every cut c of $|w|$ onto a triple $(q_1, U, q_2) \in Q \times 2^Q \times Q$.

As a first step, we introduce a concatenation operator on such triples. Let $\sigma_1, \sigma_2 \in Q \times 2^Q \times Q$ be two triples such that $\mathit{last}(\sigma_1) = \mathit{first}(\sigma_2)$ or $\mathit{last}(\sigma_1) \xrightarrow{\varepsilon} \mathit{first}(\sigma_2) \in \Delta$. Their concatenation $\sigma_1 \cdot \sigma_2$ is defined as $(\mathit{first}(\sigma_1), \mathit{states}(\sigma_1) \cup \mathit{states}(\sigma_2), \mathit{last}(\sigma_2))$.

We also define two operators $\mathit{first}(\cdot)$ and $\mathit{last}(\cdot)$ on runs. For a run ρ reading a word w, we have $\mathit{first}(\rho) = \rho\big(\widehat{|w|}_{min}\big)$ and $\mathit{last}(\rho) = \rho\big(\widehat{|w|}_{max}\big)$. Intuitively, these operators respectively identify the triples onto which the first and last cuts of $|w|$ are mapped.

Now, we are ready to build a run of $\mathcal{A}$ reading w from the words w_j, with $j \in J \cup \widehat{J^*}$. Each internal cut c of $|w_j|$ is mapped to $\rho_j(c)$. The first cut of $|w_j|$ is mapped to $\mathit{last}(\rho_k) \cdot \mathit{first}(\rho_j)$ if j has a predecessor k in $J \cup \widehat{J^*}$, to $(f_2, \{f_2\}, f_2) \cdot \mathit{first}(\rho_j)$ if j is the first element of $j \in J \cup \widehat{J^*}$, and to $\mathit{first}(\rho_j)$ otherwise. The last cut of $|w_j|$ is mapped to $\mathit{last}(\rho_j) \cdot \mathit{first}(\rho_k)$ if j has a successor k in $J \cup \widehat{J^*}$, and to $\mathit{first}(\rho_j)$ otherwise. Note that if $|w_j| = \emptyset$, and k and ℓ are respectively the predecessor and the successor of j in $J \cup \widehat{J^*}$, the only cut c of $|w_j|$ is mapped onto $\mathit{last}(\rho_k) \cdot \rho_j(c) \cdot \mathit{first}(\rho_\ell)$.

The second part of the proof is to show that every word accepted by $\mathcal{A}$ belongs to the language $L_1 \diamond L_2$. Let ρ be a run of $\mathcal{A}$ reading w. The idea is to show that ρ is a succession of passages through $\mathcal{A}_1$ and $\mathcal{A}_2$. Formally, let us

define the following subsets of $\widehat{|w|}$:

$$\Gamma_1 = \big\{c \in \widehat{|w|} \text{ such that } i_1 \in \mathit{states}\big(\rho(c)\big)\big\}$$

and

$$\Gamma_2 = \big\{c \in \widehat{|w|} \text{ such that } i_2 \in \mathit{states}\big(\rho(c)\big)\big\}.$$

Intuitively, a cut in Γ_1 represents a passage through $\mathcal{A}_1$, since after visiting the state i_1, a word accepted by $\mathcal{A}_1$ must necessarily be read. The set Γ_2 can be interpreted in a similar way in $\mathcal{A}_2$. The set $\Gamma = \Gamma_1 \cup \Gamma_2$ can be turned into a linear ordering by keeping the order relation from $\widehat{|w|}$. The intuition is that Γ represents the succession of passages through $\mathcal{A}_1$ and $\mathcal{A}_2$. However, Γ_1 and Γ_2 might have a non empty intersection, since a cut c can be mapped onto a triple $\rho(c)$ such that $\mathit{states}\big(\rho(c)\big)$ contains both i_1 and i_2 (this is only possible if $\mathcal{A}_1$ or $\mathcal{A}_2$ accepts the empty word). In other words, there might exist cuts in Γ that represent a passage through $\mathcal{A}_1$ followed by a passage through $\mathcal{A}_2$, or conversely. We need to distinguish these two types of passages, hence we define $\bar{\Gamma}$ as the linear ordering Γ in which any cut c that belongs to $\Gamma_1 \cap \Gamma_2$ has been replaced by two distinct elements c_1, c_2 such that:

- c_2 is the successor of c_1 in $\bar{\Gamma}$ if c represents a passage through $\mathcal{A}_1$ followed by a passage through $\mathcal{A}_2$. This happens when $\mathit{first}(\rho(c)) \in \{i_1\} \cup Q_2 \setminus \{i_2\}$.
- c_1 is the successor of c_2 in $\bar{\Gamma}$ if c represents a passage through $\mathcal{A}_2$ followed by a passage through $\mathcal{A}_1$. This happens when $\mathit{first}(\rho(c)) \in \{i_2\} \cup Q_1 \setminus \{i_1\}$.

The linear ordering $\bar{\Gamma}$ can then be partitioned into $\bar{\Gamma}_1$ and $\bar{\Gamma}_2$, whose elements represent passages through $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

We now enumerate three properties that come directly from the structure of $\mathcal{A}$. Let us denote by $\Delta^\diamond$ the set of limit transitions $f_2 \to \{i_1, i_2, f_2, \dots\} \to i_2$. Firstly, in the ordering $\bar{\Gamma}$, each element $c_1 \in \bar{\Gamma}_1$ that is not the first or last one has a successor and a predecessor in $\bar{\Gamma}_2$. Indeed, the limit transitions in $\Delta^\diamond$ do not make it possible to leave i_2 by a right-limit transition, or to arrive in f_2 or i_1 after following a left-limit transition. Secondly, the ordering $\bar{\Gamma}$ is complete. This comes from the fact that there does not exist any cut $c \in \widehat{|w|}$ that is mapped onto a triple σ such that $\mathit{first}(\sigma)$ is the destination of a limit transition in $\Delta^\diamond$ and $\mathit{last}(\sigma)$ is the source of a right-limit transition in $\Delta^\diamond$. Finally, for all $c_2 <_{\bar{\Gamma}} c_2' \in \bar{\Gamma}$, there exists $c_1 \in \bar{\Gamma}_1$ such that $c_2 <_{\bar{\Gamma}} c_1 <_{\bar{\Gamma}} c_2'$, since c_2 and c_2' cannot be consecutive in $\bar{\Gamma}$, and $\bar{\Gamma}$ does not admit an infinite dense subordering that only contains elements of $\bar{\Gamma}_2$. This last property is ensured by the presence of the state i_1 and the transition (f_2, ε, i_1).

We now show that those properties imply that the ordering $\bar{\Gamma}$ is isomorphic to $\bar{\Gamma}_1 \cup \widehat{\bar{\Gamma}_1}^*$, i.e., that there exists a one-to-one correspondence between these orderings that preserves order. This together with Definition 3 establishes that the word read by ρ belongs to $L_1 \diamond L_2$. Consider the mapping $\tau : \bar{\Gamma} \to \bar{\Gamma}_1 \cup \widehat{\bar{\Gamma}_1}^*$ defined as $\tau(c_1) = c_1$ for all $c_1 \in \bar{\Gamma}_1$ and $\tau(c_2) = \big(\{c_1 \in \bar{\Gamma}_1 : c_1 <_{\bar{\Gamma}} c_2\}, \{c_1 \in \bar{\Gamma}_1 : c_2 <_{\bar{\Gamma}} c_1\}\big)$ for all $c_2 \in \bar{\Gamma}_2$. The fact that this mapping is an order-preserving bijection is a consequence of the following properties:

- Each element of $\bar{\Gamma}_1$ does not belong to $\bar{\Gamma}_2$, and is mapped by τ onto itself.
- For all elements $c_2, c_2' \in \bar{\Gamma}_2$ such that $c_2 <_{\bar{\Gamma}} c_2'$, one has $\tau(c_2) <_{\bar{\Gamma}} \tau(c_2')$. This follows from the definition of τ and the existence of an element $c_1 \in \bar{\Gamma}_1$ between any two elements of $\bar{\Gamma}_2$.
- For all $(K, L) \in \widehat{\bar{\Gamma}_1}^*$, there exists $c_2 \in \bar{\Gamma}_2$ such that $\tau(c_2) = (K, L)$ and $c_1 <_{\bar{\Gamma}} c_2 <_{\bar{\Gamma}} c_1'$ for all $c_1 \in K$ and $c_1' \in L$. Indeed, if K (resp. L) has a greatest element K_{max} (resp. a least element L_{min}), then c_2 can be chosen equal to the successor of K_{max} (resp. the predecessor of L_{min}) in $\bar{\Gamma}$. Otherwise, (K, L) is a gap in $\bar{\Gamma}_1$, and the completeness property of $\bar{\Gamma}$ ensures the existence of c_2. □

5 Conclusions and Perspectives

The contribution of this paper is to introduce an extension of automata on linear orderings, in which the transition relation can contain successor transitions with an empty label. The semantics of those automata has been defined so as to minimize the impact of epsilon transitions; in particular, following infinite combinations of epsilon transitions does not suffice to enable limit transitions. We have established that moving from plain automata to epsilon automata on linear orderings does not increase their expressive power, by providing an algorithm that removes epsilon transitions without modifying the language accepted by the automaton. We have illustrated the usefulness of our extended form of automata on linear orderings by using them for correcting a minor error in [1].

We mainly expect epsilon automata on linear orderings to be useful for theoretical developments, for which the high cost of eliminating epsilon transitions is not prohibitive. There are projects however to exploit those automata to develop an effective decision procedure for some logics, in particular the first-order theory of order over $\mathbb{R}$ and $\mathbb{Q}$ with uninterpreted predicates [2]. For such applications, it will become necessary to find efficient strategies for restricting the set of limit transitions that are handled by the epsilon transition elimination procedure.

Acknowledgments. The authors wish to thank the anonymous reviewers for their insightful comments and constructive suggestions. This work is partially supported by the FNRS-DFG PDR Weave (SMT-ART) grant 40019202.

Disclosure of Interests. The authors do not have competing interests to declare that are relevant to the content of this article.

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., Hansel, G., Michaux, C., Villemaire, R.: Logic and p-recognizable sets of integers. Bulletin of the Belgian Mathematical Society **1**(2), 191–238 (1994)
4. Bruyère, V., Carton, O.: Automata on linear orderings. Journal of Computer and System Sciences **73**(1), 1–24 (2007)
5. 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, Stanford (1962)
6. Choueka, Y.: Finite automata, definable sets, and regular expressions over ω^n-tapes. Journal of Computer and System Sciences **17**(1), 81–97 (1978)
7. 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)
8. Nivat, M., Perrin, D.: Ensembles reconnaissables de mots biinfinis. In: Proc. 14th Annual ACM Symposium on Theory of Computing (STOC). pp. 47–59. ACM (1982)
9. Rabinovich, A.: Temporal logics over linear time domains are in PSPACE. Information and Computation **210**, 40–67 (2012)
10. Rispal, C., Carton, O.: Complementation of rational sets on countable scattered linear orderings. International Journal of Foundations of Computer Science **16**(04), 767–786 (2005)
11. Rosenstein, J.G.: Linear orderings. Academic press (1982)
12. Wojciechowski, J.: Finite automata on transfinite sequences and regular expressions. Fundamenta Informaticae **8**(3–4), 379–396 (1985)