Synchronizing Relations on Words

Diego Figueira and Leonid Libkin

University of Edinburgh, UK

Abstract

While the theory of languages of words is very mature, our understanding of *relations* on words is still lagging behind. And yet such relations appear in many new applications such as verification of parameterized systems, querying graph-structured data, and information extraction, for instance. Classes of well-behaved relations typically used in such applications are obtained by adapting some of the equivalent definitions of regularity of words for relations, leading to non-equivalent notions of recognizable, regular, and rational relations.

The goal of this paper is to propose a systematic way of defining classes of relations on words, of which these three classes are just natural examples, and to demonstrate its advantages compared to some of the standard techniques for studying word relations. The key idea is that of a *synchronization* of a pair of words, which is a word over an extended alphabet. Using it, we define classes of relations via classes of regular languages over a fixed alphabet, just $\{1, 2\}$ for binary relations. We characterize some of the standard classes of relations on words via finiteness of parameters of synchronization languages, called shift, lag, and shiftlag. We describe these conditions in terms of the structure of cycles of graphs underlying automata, thereby showing their decidability. We show that for these classes there exist canonical synchronization languages, and every class of relations can be effectively re-synchronized using those canonical representatives. We also give sufficient conditions on synchronization languages, defined in terms of injectivity and surjectivity of their Parikh images, that guarantee closure under intersection and complement of the classes of relations they define.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Word Relations, Regular, Rational, Recognizable

Digital Object Identifier 10.4230/LIPIcs.STACS.2014.518

1 Introduction

Foundations of formal language theory have been largely developed in the 1960s and 1970s, and used heavily in practically all areas of computer science. The field itself stayed somewhat dormant for a while, but that changed over the past 10–15 years due to new application areas requiring techniques that could not have been foreseen 30 or 40 years earlier. Among consumers of results in formal language theory are verification (for instance, automata-based approaches to model-checking are now part of standard industrial verification tools [7, 22]) and data management (standards for describing and querying XML documents, for instance, are rooted in both word and tree automata [24, 28], and emerging graph data models are borrowing many formal language concepts [3]).

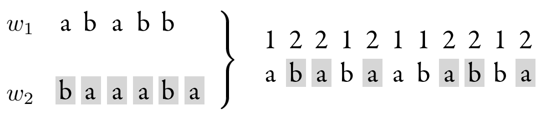
Of interest to us in this paper are *relations on words.* That is, for a given finite alphabet $\mathbb{A}$, we deal with binary relations $R \subseteq \mathbb{A}^* \times \mathbb{A}^*$. Their study goes back to Elgot, Mezei, Nivat in the 1960s [15, 25] with much subsequent work done later (see, e.g., surveys [8, 13]). The standard notions of regularity that generate the same class of languages —recognizability by finite monoids, definability by automata, or by regular expressions— give rise to different classes of relations, called *recognizable, regular, and rational* relations. Their properties may differ significantly from properties of regular languages: for instance, rational relations are not closed under intersection and it is even undecidable whether the intersection of two such languages is non-empty. Recognizable relations are just unions of products of regular languages; examples of regular relations are prefix, equality, or equal length of words; and examples of rational relations are suffix, subword (for instance, *bb* is a subword of *aabbaa*), and subsequence (*bb* is a subsequence of *abaaba*: letters need not be consecutive).

There has been renewed interest in relations on words as of late. One motivation comes from verification of safety and liveness properties of parameterized systems, where such relations describe transitions [1, 10, 20, 29]. Another comes from graph databases, which are actively studied as a suitable model for RDF data, social networks data, and others [3]. Paths in graph databases are described by their labels, and need to be compared, for instance, for their degree of similarity, e.g., their edit distance [4, 6, 23]. Yet another example is the study of formal models underlying IBM's tools for information extraction [16].

Many of the basic questions that arise in these new applications, however, are not the kind of questions that had been addressed previously. Just to give an example, it is well known that checking nonemptiness of the intersection of a rational relation and a regular relation is an undecidable problem. But what about *really used* rational relations such as subword, suffix, subsequence (as opposed to artificial codings of the halting problem) – can we test if their intersection with regular relations is nonempty? However natural these questions are, they were answered only recently [5].

An even more basic question relates to the very choice and structure of the main classes of relations: recognizable, regular, and rational. They appeared in a somewhat ad hoc way, just as analogs of different ways of defining regularity of languages, but is there another way to explain these, and perhaps other classes as well? This is the main point of our paper: we argue that there is a natural way to study relations on words, and we do it by explaining how positions in words are *synchronized.*

As an example of synchronization, consider words $w_1 = ababb$ and $w_2 = baaaba$. We can represent this pair as a single word over $\{a, b\}$, by shuffling w_1 and w_2, i.e., interspersing letters of w_1 among letters of w_2. For each position in the shuffle, we remember which word it came from – this is indicated by the symbols 1 or 2 above the letters in the figure.



When we read the letters marked i, for $i = 1, 2$ we get the word w_i. The word over $\{1, 2\}$ provides a *synchronization* of the pair (w_1, w_2) – in our example, 12212112212. We show that the commonly occurring classes of relations over words follow the same principle:

1. to decide whether (w_1, w_2) is in the relation, one runs an automaton over the shuffle;
2. classes of relations are then determined by the classes of allowed synchronizations.

For instance, recognizable relations are given by synchronizations from 1^*2^*, length-preserving regular relations by synchronizations from $(12)^*$, arbitrary regular relations by synchronizations from $(12)^*(1^*|2^*)$, and rational relations by synchronizations from $(1|2)^*$.

For relations, we have proper inclusions *recognizable* $\subsetneq$ *regular* $\subsetneq$ *rational* [8], making them very different from languages. This immediately raises the question: since every recognizable language is regular, and yet 1^*2^* is *not* contained in $(12)^*(1^*|2^*)$, there must be multiple ways of synchronizing relations to obtain even known classes. What are these ways, and how can they be characterized? And will those characterizations lead to new naturally appearing classes?

These are the questions we answer. We define three parameters of regular languages in $(1|2)^*$: the *shift* says how often we switch between 1s and 2s, the *lag* says how big the difference between the numbers of 1 and 2 is allowed to get, and *shiftlag* combines the two in a certain way. Then finite shift characterizes recognizability, while finite shiftlag characterizes regularity of relations. Finite lag, which appears to be a natural measure then, captures another known class of relations.

We provide automata characterizations of classes of synchronization languages in terms of the structure of cycles in the graph representations of automata. All these turn out to be decidable. This shows one advantage of dealing with relations in terms of their synchronizations. For instance, it is known that checking whether a given rational relation is regular, is an undecidable problem (assuming the input is a transducer, i.e., an automaton with output [8]). However, if the input to the problem is a synchronization language, then it *is* decidable whether the relations it describes are all regular.

Another advantage of describing relations by their synchronizations is the ability to find classes closed under intersection or complementation (rational relations, for instance, are not). We do it by imposing decidable conditions on Parikh images of synchronization languages to guarantee closure properties of classes of relations they give rise to.

We also look at re-synchronization of relations. For each class of relations, there may be many different regular synchronizing languages over $\{1, 2\}$. We show that in the standard cases, there exist canonical synchronizing languages, and relations can be effectively resynchronized using those canonical languages.

2 Recognizable, regular, and rational relations

We start with some basic notations. Throughout the paper, $\mathbb{A}$ stands for a finite alphabet, $\mathbb{N} = \{1, 2, \dots\}$ for the set of positive natural numbers, and $\mathbb{N}_0$ for $\mathbb{N} \cup \{0\}$. The set of all words over $\mathbb{A}$ is denoted by $\mathbb{A}^*$, and the length of w in $\mathbb{A}^*$ is denoted by $|w|$. If $w = a_1 \dots a_n$, then $w[i, j]$ stands for the subword $a_i \dots a_j$; in particular, $w[i]$ is the letter a_i.

Recall that there are three standard ways of defining regular languages:

- Recognizability by finite monoids: the set $\mathbb{A}^*$, equipped with the concatenation operation (denoted by '$\cdot$', whose unit is the empty word 'ε') is a monoid. A set $L \subseteq \mathbb{A}^*$ is recognizable if there is a finite monoid M and a homomorphism $\langle \mathbb{A}^*, \cdot, \varepsilon \rangle \to M$ so that $L = f^{-1}(M_0)$ for some $M_0 \subseteq M$.
- Definability by finite automata, say NFAs.
- Definability by regular (sometimes called rational) expressions, i.e., those built from the empty word and alphabet letters using union, concatenation, and the Kleene star.

Classical formal language theory tells us that these definitions generate the same class of languages, known as regular languages. We now adapt them to binary relations on words.

Recognizable relations Since $\langle \mathbb{A}^*, \cdot, \varepsilon \rangle$ is a monoid, $\mathbb{A}^* \times \mathbb{A}^*$ has the structure of a monoid too. We can thus define *recognizable relations* as sets $R \subseteq \mathbb{A}^* \times \mathbb{A}^*$ for which there is a finite monoid M and a morphism $f : \mathbb{A}^* \times \mathbb{A}^* \to M$ such that $R = f^{-1}(M_0)$ for some $M_0 \subseteq M$. This class will be denoted by REC.

Regular relations Let $\perp \notin \mathbb{A}$ be a new alphabet letter. A pair (w_1, w_2) of words from $\mathbb{A}^*$ can be encoded by a single word of length $\max(|w_1|, |w_2|)$ over the alphabet $(\mathbb{A} \cup \{\perp\}) \times (\mathbb{A} \cup \{\perp\})$: its ith letter is the pair containing the ith letter of w_1 and the ith letter of w_2, with $\perp$ used when i is greater than the length of w_1 or w_2. For example, the encoding for the words of the figure of page 519 is $(a, b)(b, a)(a, a)(b, a)(b, b)(\perp, a)$. A regular relation

R is given by an automaton over this alphabet: it contains pairs (w_1, w_2) whose encodings are accepted by the automaton. The class of regular relations is denoted by REG.

Rational relations There are two equivalent ways of defining them. One uses regular expressions, which are now built from pairs in $(\mathbb{A} \cup \{\varepsilon\}) \times (\mathbb{A} \cup \{\varepsilon\})$ using the same operations of union, concatenation, and Kleene star. Alternatively, rational relations can be defined by means of 2-tape automata, that have 2 heads for the tapes and one additional control; at every step, based on the state and the letters it is reading, the automaton can enter a new state and move some (not necessarily all) tape heads. The class of rational relations is denoted by RAT.

Relations in REC are exactly the finite unions of products of regular languages over $\mathbb{A}$ [8, 15]. Examples of relations in REG $\setminus$ REC are prefix, equality, or equal length. Examples of relations in RAT $\setminus$ REG are suffix, given by $\big(\bigcup_{a\in\mathbb{A}}(\varepsilon, a)\big)^* \cdot \big(\bigcup_{a\in\mathbb{A}}(a, a)\big)^*$; subword: $\big(\bigcup_{a\in\mathbb{A}}(\varepsilon, a)\big)^* \cdot \big(\bigcup_{a\in\mathbb{A}}(a, a)\big)^* \cdot \big(\bigcup_{a\in\mathbb{A}}(\varepsilon, a)\big)^*$, and subsequence: $\big(\bigcup_{a\in\mathbb{A}}(\varepsilon, a) \cup (a, a)\big)^*$.

Note that unlike in the case of languages, where the three notions coincide, we have REC $\subsetneq$ REG $\subsetneq$ RAT. The classes REC and REG are closed under intersection; however the class of rational relations is not. In fact, one can find $R \in$ REG and $S \in$ RAT so that $R \cap S \notin$ RAT. However, if $R \in$ REC and $S \in$ RAT, then $R \cap S \in$ RAT.

Relations in REC and REG inherit all the closure/decidability properties of regular languages. If $R \in$ RAT, then each of its projections is a regular language, and can be effectively constructed. Hence, the nonemptiness problem is decidable for RAT. However, testing nonemptiness of the intersection of two rational relations is undecidable. We refer to [8, 12, 27] for basic information on these relations and their decision problems.

3 Synchronizations of relations

We now formalize the idea of synchronizations informally described in the introduction. We write $\mathbf{k}$ for the set $\{1, \ldots, k\}$. A *synchronization* of a pair (w_1, w_2) of words in $\mathbb{A}^*$ is a word over $\mathbf{2} \times \mathbb{A}$ so that the projection on $\mathbb{A}$ of positions labeled i is exactly w_i, for $i = 1, 2$ (see the figure on page 519). Every word w in $(\mathbf{2} \times \mathbb{A})^*$ is a synchronization of a uniquely determined pair (w_1, w_2), where w_i is the sequence of $\mathbb{A}$-letters corresponding to the symbol i in the first position of $\mathbf{2} \times \mathbb{A}$. We denote such (w_1, w_2) by $[\![w]\!]$ and extend it to languages $S \subseteq (\mathbf{2} \times \mathbb{A})^*$ by $[\![S]\!] = \{[\![w]\!] \mid w \in S\}$.

For two words $u = a_1 \cdots a_n \in \mathbb{A}^*$ and $v = b_1 \cdots b_n \in \mathbb{B}^*$, we write $u \otimes v$ for the word $(a_1, b_1) \cdots (a_n, b_n) \in (\mathbb{A} \times \mathbb{B})^*$. The main idea of our approach to relations on words comes from two different ways of viewing words in $(\mathbf{2} \times \mathbb{A})^*$.

- Every word $w \in (\mathbf{2} \times \mathbb{A})^*$ is a synchronization of a pair $[\![w]\!] = (w_1, w_2)$.
- Every word $w \in (\mathbf{2} \times \mathbb{A})^*$ is of the form $u \otimes v$ with $u \in \mathbf{2}^*$ and $v \in \mathbb{A}^*$.

This makes it possible to define relations consisting of pairs $[\![w]\!]$ with restricted synchronizations, i.e., $w = u \otimes v$ and u belongs to a given language $L \subseteq \mathbf{2}^*$.

Formally, if $L \subseteq \mathbf{2}^*$, we say that $u \otimes v$ is *L-controlled* if $u \in L$; a language is L-controlled if all its words are. We now look at relations given by L-controlled synchronizations, i.e., for a regular language $L \subseteq \mathbf{2}^*$, let

$$\textsc{Rel}(L) \;=\; \{[\![S]\!] \mid S \text{ is a regular } L\text{-controlled language}\} \tag{1}$$

If $\mathcal{C}$ is a class of relations over $\mathbb{A}^*$, then $L \subseteq \mathbf{2}^*$ is a *synchronization* for $\mathcal{C}$ if $\textsc{Rel}(L) \subseteq \mathcal{C}$, that is, all relations given by L-controlled synchronizations belong to $\mathcal{C}$. We remark that a similar approach to defining relations was used in [18], although the questions considered were completely different.

Procedurally, each relation in $\text{Rel}(L)$ is obtained as follows:

1. Choose an automaton over $\mathbf{2} \times \mathbb{A}$;
2. consider words $u \otimes v$ it accepts so that $u \in L$,
3. view v as a synchronization of (w_1, w_2) and add the pair to the relation.

This view suggests natural candidates for capturing classes REC, REG, and RAT. For REC, relations are unions of products of regular languages, so synchronizations are of the form 1^*2^*: one starts by going over the first word, and then over the second. For REG, they are from $(12)^*(1^*|2^*)$: we first go over two words letter-by-letter, and then write out the rest of the longer word. For RAT, there are no restrictions. Indeed, we can show the following.

▶ Proposition 1.

(I) $\text{Rel}(1^*2^*) =$ REC.

(II) $\text{Rel}((12)^* \cdot (1^*|2^*)) =$ REG.

(III) $\text{Rel}((1|2)^*) =$ RAT.

It is easy to see that $\text{Rel}(L)$ is closed under union, alphabetic morphisms, and inverse alphabetic morphisms, and that $L_1 \subseteq L_2$ implies $\text{Rel}(L_1) \subseteq \text{Rel}(L_2)$.

▶ Remark. One may ask why we need to take both S and L regular in the definition (1) of $\text{Rel}(L)$. The reason why S needs to be regular is that even with regular L (e.g., 1^*), $\text{Rel}(L)$ would otherwise contain non-rational relations (e.g., $\{(a^n b^n, \varepsilon) \mid n \in \mathbb{N}\}$). If, on the other hand, L is not regular, strange things may happen. For instance, it could be that all relations in $\text{Rel}(L)$ are finite, although L is infinite. Indeed, take L as the set of all words 1^p for prime p. Note that there is no infinite regular L-controlled language, since it would imply that an infinite number of distinct primes is semi-linear. Thus, all regular L-controlled languages are finite, and $\text{Rel}(L)$ is the set of all finite relations on $\mathbb{A}^* \times \{\varepsilon\}$ so that the first component is of prime length.

4 Synchronizations for recognizable, regular, and rational relations

We have seen examples of languages characterizing the classes of recognizable, regular, and rational relations, but those are not unique. There are trivial examples such as $\text{Rel}(1^*2^*) = \text{Rel}(2^*1^*) =$ REC, and $\text{Rel}((12)^*(1^*|2^*)) = \text{Rel}((21)^*(1^*|2^*)) =$ REG, but others as well, e.g., $\text{Rel}(1^*2^*1^*2^*)$ equals REC, and $\text{Rel}(((12)^*1(12)^*2)^*(1^*|2^*)) =$ REG.

What kind of parameters guarantee that $L \subseteq \mathbf{2}^*$ synchronizes relations in a class $\mathcal{C}$, for the classes we study here? That is, what parameters guarantee that with the synchronization language L, we are guaranteed that the resulting relations are in $\mathcal{C}$?

We now answer this question, but first we need some definitions. Given a word w over some finite alphabet, and a letter a in the alphabet, we define $\#_a(w)$ as the number of occurrences of a in w. Given a word $w \in \mathbf{2}^*$, a position $i \leq |w|$, and $\delta \in \mathbb{N}$, we say i is

- δ*-lagged* if $|\#_1(w[1,i]) - \#_2(w[1,i])| = \delta$;
- $\geq\delta$*-lagged* if $|\#_1(w[1,i]) - \#_2(w[1,i])| \geq \delta$;
- $\leq\delta$*-lagged* if $|\#_1(w[1,i]) - \#_2(w[1,i])| \leq \delta$.

That is, these parameters show by how much the numbers of 1s and 2s in $w \in \mathbf{2}^*$ differ.

A *shift* of w is a position $i \in \{1, \ldots, |w| - 1\}$ so that $w[i] \neq w[i+1]$. Two shifts $i < j$ are *consecutive* if there is no shift l so that $i < l < j$.

Let *shift*(w) be the number of shifts of w, let *lag*(w) be the maximum lag of a position in w, and let *shiftlag*(w) be the maximum $n \in \mathbb{N}$ so that w contains n consecutive shifts which are $>n$-lagged. We lift these notions to languages by taking maxima, e.g., $\mathit{shift}(L) = \max_{w \in L} \mathit{shift}(w)$, and likewise for $\mathit{lag}(L)$ and $\mathit{shiftlag}(L)$. If words of arbitrarily large lag (shift, or shiftlag) occur in L, we write $\mathit{shift}(L) = \infty$ (and likewise for the other parameters).

Observe that finite shift and finite lag imply that shiftlag is finite, but the converse is not true: for $L = (12)^*1^*$ we have $\mathit{shiftlag}(L) < \infty$ and yet $\mathit{lag}(L) = \mathit{shift}(L) = \infty$.

It turns out that finiteness of the shiftlag parameter corresponds to synchronizing regular languages, and finiteness of shift corresponds to synchronizing recognizable languages. An arbitrary regular $L \subseteq \mathbf{2}^*$ is guaranteed to synchronize rational languages.

As for the finite lag, it corresponds to a class of languages that is known as well. The class REG^{bld} of *bounded length discrepancy* relations [17, 27] is defined as follows. Recall the definition of rational relations using two-tape automata. For a rational relation to be in REG^{bld} it is required that there be $\delta \geq 0$ so that in accepting runs of such automata, the heads for the two tapes are never more than δ positions apart. It also follows from [17, 27] that REG^{bld} is the class $\bigcup_{k \in \mathbb{N}_0} \textsc{Rel}(L_k)$, for $L_k = (12)^*(1^k|2^k)$. Note that $\textsc{Rel}(L_0)$ is the class of length preserving relations. A closely related class $R_{\leq} = \{(w_1, w_2) \in \mathbb{A}^* \times \mathbb{A}^* \mid |w_1| \leq |w_2|\}$ [21] can be equally defined by $\textsc{Rel}((12|2)^*)$.

Now we can state the characterization result.

▶ **Theorem 1.** *Let $L \subseteq \mathbf{2}^*$ be a regular language. Then:*

(I) *L synchronizes regular relations iff $\mathit{shiftlag}(L) < \infty$,*
(II) *L synchronizes recognizable relations iff $\mathit{shift}(L) < \infty$,*
(III) *L synchronizes relations in REG^{bld} iff $\mathit{lag}(L) < \infty$,*
(IV) *L synchronizes rational relations.*

Proof idea. For the 'if' direction of (1), one can easily show that for any regular language L with $\mathit{shiftlag}(L) < n$ there is some δ so that $L \subseteq L'$ for $L' = L_{\leq\delta\text{-lag}} \cdot (1^*|2^*)^n$, where $L_{\leq\delta\text{-lag}}$ is the (regular) language of all words with $\leq\delta$-lagged positions. On the other hand, it is easy to show that $\textsc{Rel}(L') = \mathsf{REG}$. Since $L \subseteq L'$, by applying monotonicity, we then have $\textsc{Rel}(L) \subseteq \mathsf{REG}$.

For the 'only if' direction of (1), suppose that $\mathit{shiftlag}(L) = \infty$. Note that this means that for every $s, \delta \in \mathbb{N}$ there is some $w \in L$ that has s consecutive shifts $>\delta$-lagged. Let $S \subseteq (\mathbf{2} \times \{a, b\})^*$ consist of all words $u \otimes v \in (\mathbf{2} \times \{a, b\})^*$ so that $u \in L$, and for every $i \in \{1, \dots, |v|\}$, we have $v[i] = a$ if i is a shift of u, and $v[i] = b$ otherwise. One can show that S is an L-controlled relation so that $[\![S]\!] \in \mathsf{RAT} \setminus \mathsf{REG}$. ◀

We conclude the section with a couple of examples of applications of the main result. First, we show that $\textsc{Rel}((112)^*) \not\subseteq \mathsf{REG}$. Indeed, note that for every s, δ, the word $w = (112)^{\delta+s}$ is in $(112)^*$ and the last s shifts of w are $\geq\delta$-lagged. Hence, there must be some L-controlled regular language $S \subseteq (\mathbf{2} \times \mathbb{A})^*$ so that $[\![S]\!]$ is *not* a regular relation.

As another example, we get more ways of synchronizing regular relations: given $L_1 = (1^k \cdot 2^k)^*$, $L_2 = (1^* \cdot 2^*)^k$ for some fixed k, we have $\textsc{Rel}(L_i) \subseteq \mathsf{REG}$ (in fact, $\textsc{Rel}(L_2) \subseteq \mathsf{REC}$).

Finally, we consider the (r/s)-synchronized relations [27, p.660] studied in [11]. This class can be defined as $\textsc{Rel}(L_{r/s})$, where

$$L_{r/s} = (1^r 2^s)^* \Big(\bigcup_{r' < r} (1^{r'} 2^*) \mid \bigcup_{s' < s} (1^* 2^{s'}) \Big). \tag{2}$$

It is easy to see that $\mathit{shiftlag}(L_{r/s}) = \infty$ whenever $r \neq s$, and hence that (r/s)-synchronized relations (with $r \neq s$) are not in REG.

4.1 Automata theoretic characterizations

We characterized classes of relations via conditions imposed on their synchronization languages: finite shift, lag, or shiftlag. Now we show that these conditions themselves can be characterized using automata, or more precisely, the underlying labeled graphs of automata. It turns out that the structure of the cycles provides the desired characterizations.

Since in this section we deal with synchronization languages, we consider automata over the alphabet $\{1, 2\}$. For a given NFA A, we consider the *transition graph* G_A of A as the usual representation of the transition relation, where G_A is a directed graph where states are vertices and edges are labeled by transitions. Given a cycle C of G_A, we define $\#_a(C)$ as the number of edges in C labeled with transitions reading letter a. In a *heterogeneous* cycle C we have $\#_1(C) > 0$ and $\#_2(C) > 0$; otherwise a cycle is *homogeneous*. A cycle C is *balanced* if $\#_1(C) = \#_2(C)$, otherwise it is *unbalanced* (these definitions are closely related to the notions of balanced/unbalanced oriented cycles in digraphs, cf. [19]). Note that all balanced cycles are also heterogeneous.

Recall that the trim automaton is the result of removing all states which are not reachable from the initial state, and all states from which no final state is reachable.

▶ **Theorem 2.** *For any trim NFA A over the alphabet* $\mathbf{2}$*, and its transition graph* G_A*,*

(I) *$\mathit{shiftlag}(L(A)) = \infty$ iff*
- *G_A contains a heterogeneous unbalanced cycle, or*
- *G_A contains a path from a homogeneous to a heterogeneous cycle,*

(II) *$\mathit{shift}(L(A)) = \infty$ iff G_A has a heterogeneous cycle,*

(III) *$\mathit{lag}(L(A)) = \infty$ iff G_A has an unbalanced cycle.*

Proof idea. The ‘if’ directions of all items are straightforward. For the ‘only if’ direction of item (1), it can be shown that for $n = 2|Q| + 1$ (where $|Q|$ is the number of states of A), any accepting run of A on $w \in L(A)$ so that $\mathit{shiftlag}(w) \geq n$ must induce a path on the transition graph G_A of A containing either a heterogeneous unbalanced cycle, or a homogeneous cycle followed by a heterogeneous cycle. Once this is verified, the statement follows. Note that since $\mathit{shiftlag}(w) \geq n$, w must contain n consecutive $>n$-lagged shifts $1 \leq a_1 < a_2 < \cdots < a_n \leq |w|$ in w. Since a_1 is $>n$-lagged, there must be an unbalanced cycle C_1 contained in the path induced by the run ρ restricted to $w[1, a_1]$. Since there is a sufficiently large number of shifts, there must be some heterogeneous cycle C_2 contained in the path induced by the run ρ restricted to $w[a_1, |w|]$. Of course, we have that there is a path from C_1 to C_2 in G_A, showing (1). ◀

▶ **Corollary 3.** *Checking whether* $\textsc{Rel}(L(A)) \subseteq \mathsf{REG}$*,* $\textsc{Rel}(L(A)) \subseteq \mathsf{REC}$ *or* $\textsc{Rel}(L(A)) \subseteq \mathsf{REG}_2^{bld}$ *can be done in polynomial time in the size of A.*

Note that Corollary 3 does *not* mean that it is decidable whether a relation $R \in \mathsf{RAT}$ is in REG (in fact, this problem is undecidable [8, Theorem 8.4-(vi)]). What one can check is whether it has a “safe” control, in the sense that it synchronizes regular relations. Hence, for any relation R controlled by $L(A)$, if $\textsc{Rel}(L(A)) \subseteq \mathsf{REG}$ then $R \in \mathsf{REG}$, but the opposite does not necessarily hold. For example, if we take $L' = (1|2)^*$, we have that $\textsc{Rel}(L') \not\subseteq \mathsf{REG}$ but the universal relation $\mathbb{A}^* \times \mathbb{A}^*$ is obviously in REG.

5 Resynchronizing relations

We saw that different languages in $\mathbf{2}^*$ can generate the same class relations, and yet for the commonly used classes, we have synchronization languages that somehow look canonical:

for instance, $(12)^*(1^*|2^*)$ for REG. Thus, we now address the question whether we can resynchronize relations using those canonical synchronization languages, and if so, can we do it effectively?

To pose this formally, suppose two different languages $S, S' \subseteq (\mathbf{2} \times \mathbb{A})^*$ controlled by $L, L' \subseteq \mathbf{2}^*$ respectively represent the same relation, i.e., $[\![S]\!] = [\![S']\!]$. Then we say that S is an *L-resynchronization* of S'. Given a class $\mathcal{C}$ of regular languages over $\mathbf{2}$, we say that $L_0 \in \mathcal{C}$ is a *canonical representative* of $\mathcal{C}$ if for every $L \in \mathcal{C}$ and every L-controlled language S there exists an L_0-resynchronization of S. In other words, for every $L \in \mathcal{C}$ and $R \in \textsc{Rel}(L)$, there is an L_0-controlled $S' \in (\mathbf{2} \times \mathbb{A})^*$ so that $[\![S']\!] = R$. If, in addition, there is a recursive procedure that constructs such an L_0-resynchronization of S, then we say that L_0 is an *effective canonical representative* of $\mathcal{C}$.

Let RL_{all} be the class of all regular languages over $\mathbf{2}$, and let $\mathsf{RL}^{fin}_{param}$ stand for class of regular languages $L \subseteq \mathbf{2}^*$ with finite parameter *param*, where *param* is lag, or shift, or shiftlag. We also let $\mathsf{RL}_{lag\leq\delta}$ denote the class of all regular languages $L \subseteq \mathbf{2}^*$ with $lag(L) \leq \delta$.

▶ **Example 4.** *Take, for example, $L_1 = (1122)^*1^*2^*$ and $L_2 = (12)^*(1^*|2^*)$, and a L_1-controlled relation S_1. Since $shiftlag(L_1) < \infty$, $[\![S_1]\!] \in$ REG by Theorem 1. Further, since by Proposition 1-(2) $\textsc{Rel}(L_2) =$ REG, there must be some L_2-controlled relation S_2 so that $[\![S_2]\!] = [\![S_1]\!]$. In other words S_2 is the L_2-resynchronization of S_1. Since $\textsc{Rel}(L_2) =$ REG in fact L_2 is a canonical representative of $\mathsf{RL}^{fin}_{shiftlag}$.*

▶ **Theorem 5** (Resynchronization theorem)**.**
(I) *$(12)^*(1^*|2^*)$ is an effective canonical representative of $\mathsf{RL}^{fin}_{shiftlag}$;*
(II) *1^*2^* is an effective canonical representative of $\mathsf{RL}^{fin}_{shift}$;*
(III) *there is no canonical representative of RL^{fin}_{lag};*
(IV) *$(12)^*(1^{\leq\delta}|2^{\leq\delta})$ is an effective canonical representative of $\mathsf{RL}_{lag\leq\delta}$;*
(V) *$\mathbf{2}^*$ is an effective canonical representative of RL_{all}.*
If the relations are given as NFA, the synchronization procedures are in exponential time.

Proof idea. We only give the proof sketch for (1), the other items being easier.

The *strongly connected components* (henceforth SCC) of G_A are its maximal strongly connected subgraphs. An SCC is *heterogeneous* if it contains a heterogeneous cycle; an SCC is *homogeneous* if it contains a cycle and all the cycles it contains are homogeneous; otherwise, an SCC without cycles (that is, a single vertex) is an *edgeless* SCC. The *condensation* of G_A (written $con(G_A)$) is the labeled directed acyclic graph (henceforth labeled DAG) induced by the SCC's of G_A. This is the labeled DAG whose nodes are the SCC's of G_A, and there is an edge labeled $(q, (i, a), q')$ from vertex v to vertex v' iff $v \neq v'$, q belongs to the SCC v in G_A, q' belongs to the SCC v' in G_A, and there is an edge labeled $(q, (i, a), q')$ from q to q' in G_A (in other words, $(q, (i, a), q')$ is a transition of A).

Let $S \subseteq (\mathbf{2} \times \mathbb{A})^*$ be an L-controlled regular language with $shiftlag(L) < \infty$. Let A be an NFA recognizing S with statespace Q, initial state q_0 and set of final states Q_F.

Note that since the projection of S onto $\mathbf{2}$ is inside L, we can apply Theorem 2-(1) to A, obtaining that there are no paths from homogeneous SCC's to heterogeneous SCC's in G_A (and there are no heterogeneous cycles C with $\#_1(C) \neq \#_2(C)$). Let Q_{hom} be the set of all vertices of G_A that are reachable from a vertex of a homogeneous SCC. Note that Q_{hom} includes all vertices in homogeneous SCC's, plus some vertices from edgeless SCC's. Also, note that the subgraph of G_A induced by Q_{hom} has no heterogeneous cycles. Let $Q_{het} = Q \setminus Q_{hom}$. Hence, Q_{het} includes all vertices in heterogeneous SCC's and some vertices in edgeless SCC's. Also, by the property before, the subgraph of G_A induced by Q_{het} is

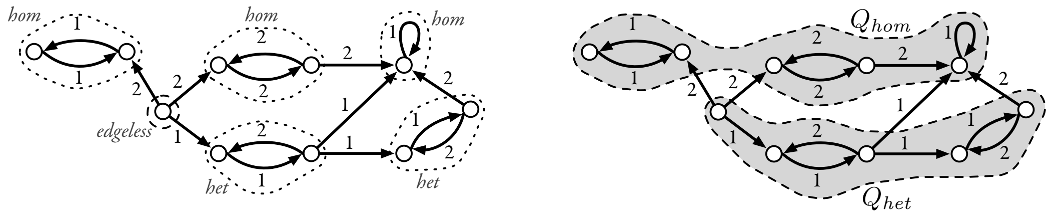


Figure 1 Example of G_A with the subgraphs induced by Q_{hom} and Q_{het}. For simplicity we assume that $\mathbb{A} = \{a\}$ and we hence omit the letter a when depicting edges labeled by (i, a).

connected. Figure 1 contains an example. Further, any path P in G_A is of the form (1) $P \cdot (q, \tau, q') \cdot P'$, (2) P, or (3) P', where

- P is a (possibly empty) path of the subgraph of G_A induced by Q_{het},
- P' is a (possibly empty) path of the subgraph of G_A induced by Q_{hom},
- $q \in Q_{het}$, $q' \in Q_{hom}$, and τ is a transition of A.

Let A^{het} be A restricted to Q_{het}, and let A^{hom} be A restricted to Q_{hom}. For every pair of states $q_{het} \in Q_{het}$ and $q_{hom} \in Q_{hom}$, let $L_{q_{het},q_{hom}}$ be the union of all

$$L(A^{het}[q_0, q_{het}]) \cdot \{(i, a)\} \cdot L(A^{hom}[q_{hom}, q_f])$$

for every $q_f \in Q_F$ and $(i, a) \in \mathbf{2} \times \mathbb{A}$ so that $(q_{het}, (i, a), q_{hom})$ is a transition of A. Let $L_{hom} = \bigcup_{q_f \in Q_F} L(A^{hom}[q_0, q_f])$ and $L_{het} = \bigcup_{q_f \in Q_F} L(A^{het}[q_0, q_f])$. It follows that

$$S = L_{hom} \cup L_{het} \cup \bigcup_{q_{het} \in Q_{het}, q_{hom} \in Q_{hom}} L_{q_{het},q_{hom}} .$$

We show that we can build, in exponential time, a $(12)^*(1^*|2^*)$-controlled automaton for each of these languages. Since the case of $L_{q_{het},q_{hom}}$ is more general than L_{hom} and L_{het}, we will only prove this case.

Note that by definition of A^{het} and A^{hom}, and since G_A has no unbalanced heterogeneous cycles, for every $q_{het} \in Q_{het}, q_{hom} \in Q_{hom}, q_f \in Q_F$ we have that $\mathit{lag}(L(A^{het}[q_0, q_{het}])) < \infty$ and $\mathit{shift}(L(A^{hom}[q_{hom}, q_f])) < \infty$. This implies that $\mathit{lag}(L(A^{het}[q_0, q_{het}])) \leq n$, and $\mathit{shift}(L(A^{hom}[q_{hom}, q_f])) \leq n$, for $n = |A|$.

By the already shown item (2), there exists a (1^*2^*)-controlled automaton $A^{hom}_{q_{hom},q_f}$ so that $[\![L(A^{hom}[q_0, q_{hom}])]\!] = [\![L(A^{hom}_{q_0,q_{hom}})]\!]$. By item (4), there exists a $(12)^*(1^{\leq n}|2^{\leq n})$-controlled automaton $A^{het}_{q_0,q_{het}}$ so that $[\![L(A^{het}[q_0, q_{het}])]\!] = [\![L(A^{het}_{q_0,q_{het}})]\!]$. These automata can be built in exponential time.

Indeed, a $(12)^*(1^*|2^*)$-controlled automaton for $L_{q_{het},q_{hom}}$ can be built from $A^{het}_{q_0,q_{het}}$ and all the $A^{hom}_{q_{hom},q_f}$'s for all $q_f \in Q_F$ in polynomial time, and thus the statement follows. This is shown by a variant of (2), showing that from any (1^*2^*)-controlled automaton one can build, in polynomial time, an equivalent automaton (in the sense of the relation it represents) that is $(12)^*(1^*|2^*)$-controlled. ◀

6 Closure via Parikh images

It is well known that the class REG is effectively closed under Boolean operations. Although RAT is a natural generalization of REG, it is not a Boolean algebra (let alone an effective one), not being closed under intersection or complement [8]. Even testing whether a rational relation is regular, or whether it has an empty intersection with a regular relation is undecidable [8].

Since regular relations are characterized via finite shiftlag, it is natural to ask whether infinite shiftlag somehow describes "dangerous" classes of relations. That is, does this mean for example that for any $L \subseteq \mathbf{2}^*$ with $\mathit{shiftlag}(L) = \infty$ the intersection problem is undecidable for $\textsc{Rel}(L)$? The answer to this question is negative: take for instance $L = (122)^*$ with $\mathit{shiftlag}(L) = \infty$. However, it is not hard to see that $\textsc{Rel}(L)$ is effectively closed under intersection.

This raises the question of whether there are classes $\mathcal{C} \subseteq \mathsf{RAT}$ that are natural, expressive, and well-behaved, that is, so that

- $\mathsf{REC} \subsetneq \mathcal{C}$,
- $\mathcal{C}$ is effectively closed under union, intersection and complementation (i.e., is an *effective Boolean algebra*); and
- $\mathcal{C}$ corresponds to a natural condition on the language.

Note that REG is one such example. Here we address the question from our perspective in terms of control languages. The idea is to show sufficient conditions of synchronization languages L so that $\textsc{Rel}(L)$ is effectively closed under intersection, or an effective boolean algebra. We state those in terms of Parikh images of languages.

Recall that the *Parikh image* of a word $w \in \mathbf{k}^*$, written $\Pi(w)$, is the vector of $\mathbb{N}_0^k$ whose ith component contains $\#_i(w)$, the number of occurrences of i in w. The Parikh image of a language L is $\Pi(L) = \{\Pi(w) \mid w \in L\}$. It is well known that for regular and context-free languages L, sets $\Pi(L)$ are exactly the semi-linear sets in $\mathbb{N}_0^k$, see [26].

A language $L \subseteq \mathbf{k}^*$ is

- *Parikh-injective* if the function $\Pi : L \to \mathbb{N}_0^k$ is injective, and
- *Parikh-surjective* if the function $\Pi : L \to \mathbb{N}_0^k$ is surjective.

▶ **Example 6.**

- *$(12)^*(1^*|2^*)$ and 1^*2^* are Parikh-injective, while $(1|2)^*$ is not.*
- *It can easily be shown that $L = w_1^* \cdot w_2^* \cdots w_\ell^* \subseteq \mathbf{k}^*$ is Parikh-injective if $\ell \le k$ and $\{\Pi(w_1), \dots, \Pi(w_\ell)\}$ generate a linear subspace of $(\mathbb{N}_0)^k$ of dimension ℓ. For example, $(122)^*(112)^*$ is Parikh-injective.*
- *$(12)^*(1^*|2^*)$, 1^*2^*, and $(1|2)^*$ are Parikh-surjective, but $(122)^*(112)^*$ is not Parikh-surjective.*
- *It is easy to see that $L_{r/s}$ as defined in (2) is Parikh-injective and Parikh-surjective for any choice of r, s. For example, if $r = 2$, $s = 1$, we have $L_{r/s} = (122)^*(22^*|1^*2|1^*)$, which is Parikh-injective and Parikh-surjective, since every element of $(\mathbb{N}_0)^2$ is covered, and there is only one way to reach any element of $(\mathbb{N}_0)^2$.*

We now analyze the (effective) closure of classes $\textsc{Rel}(L)$ under Boolean operations. It turns out that closure under union is free, but for closure under intersection and complement, the newly introduced criteria serve as sufficient conditions.

▶ **Theorem 7.** *Let $L \subseteq \mathbf{2}^*$ be a regular language. Then*

(I) $\textsc{Rel}(L)$ *is effectively closed under union, alphabetic morphisms, and inverse alphabetic morphisms;*

(II) *If L is Parikh-injective, then* $\textsc{Rel}(L)$ *is effectively closed under intersection;*

(III) *if L is both Parikh-injective and Parikh-surjective, then* $\textsc{Rel}(L)$ *is effectively closed under complement.*

Proof idea. We prove only item (3). Let $S \subseteq (\mathbf{2} \times \mathbb{A})^*$ be an L-controlled relation. We show that $[\![S]\!]^c = [\![S^c \cap (L \otimes \mathbb{A}^*)]\!]$, where S^c, $[\![S]\!]^c$ denote the complement of S, $[\![S]\!]$ respectively, and $L \otimes \mathbb{A}^*$ denotes the set of all words $u \otimes v$ where $|u| = |v|$, $u \in L$ and $v \in \mathbb{A}^*$.

[$\subseteq$] Suppose $(u,v) \notin [\![S]\!]$. We show that there must be some $w \in S^c \cap (L \otimes \mathbb{A}^*)$ so that $(u,v) = [\![w]\!]$. By Parikh surjectivity and injectivity, there is exactly one word $w' \in L$ so that $\Pi(w') = (|u|,|v|)$. Let $w = u' \otimes v' \in (\mathbf{2} \times \mathbb{A})^*$ be the only word so that $u' = w'$ and $[\![w]\!] = (u,v)$. Note that $w \notin S$ and that its projection onto the first component (*i.e.*, w') is in L. Therefore, $w \in S^c \cap (L \otimes \mathbb{A}^*)$.

[$\supseteq$] Assume $w \in S^c \cap (L \otimes \mathbb{A}^*)$ and suppose that $[\![w]\!] \in [\![S]\!]$. Then, there is some $w' \in S$ so that $[\![w']\!] = [\![w]\!]$. It cannot be that $w' = w$, as it would be in contradiction with $w \in S^c \cap (L \otimes \mathbb{A}^*)$. Since L is Parikh-injective, and w, w' are L-controlled, $w = w'$, as otherwise $[\![w']\!] \neq [\![w]\!]$. This contradicts $w \in S^c \cap (L \otimes \mathbb{A}^*)$. Thus, $[\![w]\!] \notin [\![S]\!]$ and $[\![S]\!]^c \supseteq [\![S^c \cap (L \otimes \mathbb{A}^*)]\!]$. ◀

▶ **Corollary 8.** *If $L \subseteq \mathbf{2}^*$ is Parikh-injective and Parikh-surjective, then* $\text{Rel}(L)$ *is an effective boolean algebra, closed under alphabetic morphisms and inverse alphabetic morphisms.*

Observe that in this context, REG and REC are simply two examples of the (infinitely) many such well-behaved classes.

▶ **Example 9.**

- REC *and* REG *are effective boolean algebras because they correspond to* $\text{Rel}(1^*2^*)$ *and* $\text{Rel}((12)^*(1^*|2^*))$*, where* 1^*2^*, $(12)^*(1^*|2^*)$ *are Parikh-injective and Parikh-surjective.*
- $\text{Rel}((122)^*(112)^*)$ *is effectively closed under intersection.*
- *It was shown in [11] that the class of* (r/s)*-synchronized relations is an effective Boolean algebra. Our results provide an alternative proof, since* $L_{r/s}$ *is Parikh-injective and Parikh-surjective.*

Observation. Note that Theorem 7 cannot be generalized to finite unions of Parikh-injective languages, since for example $\text{Rel}(L)$ for $L = ((12)^*1^*)|(1^*(12)^*)$ is not closed under intersection. In fact, its intersection problem is undecidable. This follows from the fact that $\text{Rel}(L)$ contains the suffix relation and all regular relations (where the first component is longer than the second). By [5, Theorem V.1], this problem is undecidable.

7 Future work

We presented a new way of looking at relations on words, and this new perspective opens up several directions. An obvious one is to extend results to k-ary relations, for $k > 2$. We know that exact analogs of Proposition 1, Theorem 1, and Theorem 2 continue to hold.

Another natural extension is to look for other classes of relations, say analogs of context-free languages. In particular, one can look at a generalization of rational relations, the *pushdown relations* of [14], which are those recognized by multi-tape automata with a stack or, equivalently, by a context-free grammar. We have some preliminary results in this direction but more work is needed.

We also would like to use the structural approach to look for better behaved classes of relational word transducers for verification purposes, and for classes of relations that can be effectively used in querying graph data. Finally, we would like to use it to identify classes of well behaved relations over *data words* [9] and study logics over them, extending the approach of [5, 6] with data.

Acknowledgment. Work partially supported by EPSRC grants G049165 and J015377.

References

1 P.A. Abdulla, B. Jonsson, M. Nilsson, and M. Saksena. A survey of regular model checking. In *CONCUR'03*, pages 35–48.

2 R. Alur and P. Madhusudan. Visibly pushdown languages. In *STOC'04*, pages 202–211.

3 R. Angles and C. Gutiérrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1), 2008.

4 K. Anyanwu and A.P. Sheth. ρ-queries: enabling querying for semantic associations on the semantic web. In *WWW'03*, pages 690–699.

5 P. Barceló, D. Figueira, and L. Libkin. Graph logics with rational relations. *LMCS*, 9(3:1), 2013.

6 P. Barceló, L. Libkin, A. W. Lin, and P. Wood. Expressive languages for path queries over graph-structured data. *ACM Trans. Database Syst.*, 37(4):31, 2012.

7 M. Ben-Ari. *Principles of the Spin model checker.* Springer, 2008.

8 J. Berstel. *Transductions and Context-Free Languages.* B. G. Teubner, 1979.

9 M. Bojańczyk. Automata for data words and data trees. In *RTA'10*, pages 1–4.

10 A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *CAV'00*, pages 403–418.

11 O. Carton. The growth ratio of synchronous rational relations is unique. *TCS*, 376(1-2):52–59, 2007.

12 O. Carton, C. Choffrut, and S. Grigorieff. Decision problems among the main subfamilies of rational relations. *RAIRO Theor. Inf. and Appl.*, 40(2):255–275, 2006.

13 C. Choffrut. Relations over words and logic: A chronology. *Bull. of the EATCS*, 89:159–163, 2006.

14 C. Choffrut and K. Culik II. Properties of finite and pushdown transducers. *SIAM J. Comput.*, 12(2):300–315, 1983.

15 C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Dev.*, 9(1):47–68, January 1965.

16 R. Fagin, B. Kimelfeld, F. Reiss, and S. Vansummeren. A formal framework for information extraction. In *PODS'13*, pages 37–48.

17 C. Frougny and J. Sakarovitch. Synchronized rational relations of finite and infinite words. *TCS*, 108(1):45–82, 1993.

18 T. Harju, A. Mateescu, A. Salomaa. shuffle on trajectories: the Schützenberger product and related operations. *MFCS'98*, pages 503–511.

19 P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* Oxford University Press, 2004.

20 B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In *TACAS'00*, pages 220–234.

21 J. Leguy. Transductions rationnelles décroissantes. *ITA*, 15(2):141–148, 1981.

22 K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

23 R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

24 F. Neven. Automata, Logic, and XML. In *CSL'02*, pages 2–26.

25 M. Nivat. Transduction des langages de Chomsky. *Ann. Inst. Fourier*, 18:339–455, 1968.

26 R. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.

27 J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.

28 T. Schwentick. Automata for XML – a survey. *JCSS*, 73(3):289–315, 2007.

29 A. W. To and L. Libkin. Algorithmic metatheorems for decidable LTL model checking over infinite systems. In *FOSSACS'10*, pages 221–236.