The "equal last" predicate for words on infinite alphabets and classes of multitape automata

Christian Choffrut *
httpr://www.liafa.jussieu.fr/∼cc
cc@liafa.jussieu.fr

Serge Grigorieff *
httpr://www.liafa.jussieu.fr/∼seg
seg@liafa.jussieu.fr

September 22, 2008

Abstract

Along with the usual predicates "prefix" and "equal length", the predicate "equal last letter" leads to a first order theory of the free infinitely generated monoid whose definable relations are related to the algebra of relations recognized by different types of multitape automata which are natural extensions of the famous Rabin-Scott multitape automata and the so-called synchronous automata. We investigate these classes of automata and solve decision isssues concerning them.

1 Introduction

The theory of regular languages of a free monoid as developed in the fifties, now belongs to the background of any undergraduate student in Computer Science. Known to a more restricted public but equally interesting is its richness testified by the numerous different characterizations thereafter found by Büchi in terms of second order logic and Schützenbeger in algebraic terms. A great part of theoretical Computer Science is nowadays devoted to extensions of this theory to more general structures such as n-tuples of finite or infinite words, finite or infinite trees, words indexed by arbitrary orderings, traces and message sequence charts with various domains of applications, cf. [2, 4, 3, 20]. The present work is concerned with the extension to infinite alphabets of the theory of sets of n-tuples of words recognized by finite automata, also known as rational relations, and their possible logical characterization of which we briefly review some of the challenges.

*LIAFA, Université Paris 7 & CNRS, 2, pl. Jussieu 75251 Paris Cedex 05

With finite alphabets, the main difference between regular sets of words and regular sets of n-tuples of words is that the former are closed under complement but that the latter are not. As a consequence, a logical characterization of regular sets of n-tuples of words misses and, if it exists, it can only be as a fragment of a theory. However, right from the beginning, attempts were made to express these relations in some logical language, [9]. To our knowledge, only subfamilies of rational relations have received a logical characterization, namely, the co-called *special* relations of Laüchli & Savioz, [14], first introduced by Angluin & Hoover, [1], and the *synchronous* relations of Elgot, Eilenberg and Shepherdson, [8]. A recent account on the subject can be found in [5].

The purpose of the present work will be better understood if we recall the literature. As early as 1969, Eilenberg, Elgot & Shepherdson, proved that, for a finite alphabet Σ, the family of n-ary relations on the free monoid Σ^* which can be recognized by some n-tape automaton with reading heads moving synchronously on the tapes are precisely those relations which are definable in the first order theory of the free monoid Σ^* with the following predicates: unary predicates asserting that a word x ends with some specific letter a, denoted by $\texttt{Last}_a(x)$ and two binary predicates, one asserting that the words x and y have the same length, denoted by $\texttt{EqLen}(x, y)$ and the other one asserting that a word x is a prefix of a word y, denoted by $x \leq_{\texttt{pref}} y$. Their paper ends with some considerations involving the possible extension to infinite alphabets. We tried to clarify these observations and proved and disproved some of the conjectures in [6]. We showed that for words on infinite alphabets, the three predicates characterize relations that can be recognized by some synchronous automaton subject to some additional technical constraint. We also proved that a logical characterization of the relations recognized by truly synchronous automata requires an extra binary predicate `EqLenEqLast` which asserts that two words have the same length along with the same last letter: intuitively this predicate cannot be expressed by the previous three predicates since it leads to an infinite disjunction of conjunctions of the form $\texttt{Last}_a(x) \wedge \texttt{Last}_a(y)$ for all $a \in \Sigma$. Consequently, the two logics are decidable because the truth of a sentence reduces to a problem of accessibility in a finite automaton. In a further work, we disconnected the predicate "equal last letter", denoted by `EqLast`, from the predicate "equal length" and proved that the new logic is strictly more powerful and that it is no longer decidable, [7]. We started investigating the decision properties of the prefix classes and showed that the fragment $\exists\forall\forall$ is undecidable.

We now discuss the main results of the present work. In order to approximate the predicate $\texttt{EqLast}$, we introduce two classes of multitape automata. For the first one, called *multicell synchronous automata*, the reading heads still move simultaneously on the different tapes, but they scan not only the cell on the current position but the d last cells. In particular with $d = 2$ such automata are able to recognize the set of all words made of a single repeated symbol: no synchronous automaton may recognize this set since it may not compare successive symbols. This is an evidence that this new class is strictly more powerful than that of synchronous automata. The second class is simply that of multitape automata with reading heads moving independently, which we call *asynchronous automata* and which is the natural extension to infinite alphabets of the ordinary notion of multitape automaton for finite alphabets. In both cases we study the closure properties. Multicell synchronous relations are closed under all natural operations except projection. Asynchronous relations are closed under the regular operations (set union, componentwise concatenation and Kleene star) but not under intersection nor complementation.

Let's denote by $\mathcal{R}^{syn}$, $\mathcal{R}^{mult}$ and $\mathcal{R}^{asyn}$ the respective classes of synchronous, multicell synchronous and asynchronous relations. Consider the following structures (where "ees" is an acronym for Eilenberg, Elgot & Shepherdson)

$$\begin{array}{lcl}
\mathbf{S}_{\text{ees}} & = & \langle \Sigma^*; \leq_{\texttt{pref}}, \texttt{EqLen}, (\texttt{Last}_a)_{a\in\Sigma}\rangle \\
\mathbf{S}_{\text{sync}} & = & \langle \Sigma^*; \leq_{\texttt{pref}}, \texttt{EqLen}, \texttt{EqLenEqLast}, (\texttt{Last}_a)_{a\in\Sigma}\rangle \\
 & = & \langle \Sigma^*; (R)_{R\in\mathcal{R}^{syn}}\rangle \\
\mathbf{S}_{\text{last}} & = & \langle \Sigma^*; \leq_{\texttt{pref}}, \texttt{EqLen}, \texttt{EqLast}, (\texttt{Last}_a)_{a\in\Sigma}\rangle \\
\mathbf{S}_{\text{mult}} & = & \langle \Sigma^*; (R)_{R\in\mathcal{R}^{mult}}\rangle \\
\mathbf{S}_{\text{asyn}} & = & \langle \Sigma^*; (R)_{R\in\mathcal{R}^{asyn}}\rangle
\end{array}$$

In case Σ is a finite alphabet, the main result in Eilenberg, Elgot & Shepherdson [8] insures that the three first structures are equivalent: they define the same relations, namely those in $\mathcal{R}^{syn}$, and their theories are decidable. Since, for finite Σ, multicell synchronous automata reduce to synchronous automata, the fourth structure is also equivalent to the first three ones and has a decidable theory. As for the structure $\mathbf{S}_{\text{asyn}}$, its theory is undecidable and its definable relations are exactly the arithmetical ones. This is so because concatenation is definable as the intersection of the three asynchronous relations $\{(x, y, z) \mid |z| = |x| + |y|\}$, $\{(x, y, z) \mid x \leq_{\texttt{pref}} z\}$ and $\{(x, y, z) \mid y \text{ is a suffix of } z\}$.
For an infinite alphabet Σ, the picture is totally different.

As said above, in [6] and [7] we proved that the first three structures have increasing expressive power, the first two have decidable theories whereas the third one allows to define concatenation and therefore has an undecidable theory.
Here we prove that the last three structures are equivalent: they define the same family of relations (cf. Theorem 18 which refines Theorem A below).

Theorem A. *Given a relation on the free monoid Σ^* on the infinite alphabet Σ, it is first order definable in $\mathbf{S}_{\mathrm{last}}$ if and only if it is first order definable in $\mathbf{S}_{\mathrm{mult}}$ if and only if it is first order definable in $\mathbf{S}_{\mathrm{async}}$.*

In case Σ is countable, every bijection $f : \mathbb{N} \to \Sigma$ allows to transfer the computability structure on $\mathbb{N}$ to an f-computability structure on Σ and Σ^*. Let's call f-arithmetical any relation on Σ^* which can be obtained from f-computable ones via a series of projections and complementations.
Finally, let's say that a relation R over Σ^* is finitary if there exists a finite subalphabet $\Sigma_0 \subset \Sigma$ such that R is invariant under any bijection of Σ^* stemming from a bijection on the alphabet Σ which is the identity on Σ_0.
These two notions are the ingredients of a characterization of relations definable in our structures (cf. Theorem 21 which refines Theorem B below).

Theorem B. *Suppose Σ is countable. An n-ary relation R on Σ^* is definable in the structures $\mathbf{S}_{\mathrm{last}}$, $\mathbf{S}_{\mathrm{mult}}$, $\mathbf{S}_{\mathrm{async}}$ if and only if it is finitary and f-arithmetical for some (resp. for all) bijection $f : \mathbb{N} \to \Sigma$.*

Thus, the big jump decidable/undecidable occurring for finite alphabets with asynchronous automata, already occurs with the `EqLast` predicate when dealing with infinite alphabets.

Consequently, due to our previous undecidability result mentioned above, the theory of the last four structures on the free monoid with infinite generators is undecidable and the decidable fragments must lie in low level prefix classes. We study for each structure which fragments are undecidable. We should expect that the predicate `EqLast` is less efficient than the multicell predicates and these are less efficient than the asynchronous predicates. This is indeed the case. We proved in [7] that the $\exists\forall\forall$ fragment of $\mathbf{S}_{\mathrm{last}}$ is undecidable . With $\mathbf{S}_{\mathrm{async}}$, most usual decidability results for languages fail for relations, even for finite alphabets. There are two notable exceptions. First, a trivial one: the emptyness problem remains decidable. Second, the equivalence problem, which is undecidable in general, is nevertheless decidable when we consider deterministic asynchronous automata. This is a difficult result using sophisticated algebraic tools about skew fields, cf. Harju & Karhümaki, [11, 12].

The main results concerning the structure $\mathbf{S}_{\text{mult}}$ is that its $\exists^*$-fragment is decidable (we reduce it to the case of finite alphabets) whereas the $\exists\forall$-fragment is undecidable. This latter result is rather surprising and is achieved by an elaborated encoding of Post Correspondence Problem similar to that in [7]. The gain of efficiency relative to the structure $\mathbf{S}_{\text{last}}$ is due to the fact that multicell predicates allow us to express "equal last letter" in an indirect way, thus sparing us one universal quantifier. The results are summarized in the following table.

structure	decidable fragment	undecidable fragment
$\mathbf{S}_{\text{last}}$	$\exists^*$ [7]	$\exists\forall\forall$ [7]
$\mathbf{S}_{\text{mult}}$	$\exists^*$ Thm. 23	$\exists\forall$ Thm. 24
$\mathbf{S}_{\text{async}}$	$\exists x_1 \dots \exists x_n\ R(x_1, \dots, x_n)$ (R is an asynchronous relation)	$\exists x\ R(x,x)$ Prop. 25 $\forall x\ R(x,x)$ [13] (R asynchronous)

Table 1: Decision status of fragments for the different structures

2 Preliminaries

2.1 Infinitely generated free monoids

We denote by Σ^* the free monoid generated by the set Σ. The elements of Σ and Σ^* are *letters* and *words* (or *strings*) respectively. The *empty* word is denoted by ε. The length of a word u is denoted by $|u|$ and the i-th occurrence of u, for all $1 \leq i \leq |u|$, is denoted by $u[i]$. For all $1 \leq i < j$, $u[i \dots j]$ denotes the empty word if $|u| < i$ and the word $u[i] \dots u[k]$ where $k = \min\{j, |u|\}$ otherwise. The *concatenation* of $u \in \Sigma^*$ and $v \in \Sigma^*$ is denoted by uv. We say that u (resp. v) is a *prefix* (resp. *suffix*) of uv and the prefix is *proper* if $v \neq \varepsilon$. The concatenation is extended to subsets of Σ^* in the natural way. In the sequel, unless otherwise stated, Σ refers to a fixed infinite alphabet.

2.2 Finite automata over infinite alphabets

The main objects studied in this work are finite automata on n-tuples of words over non necessarily finite alphabets. This requires an extension of the ordinary notion of finite automata as discussed in [6]. We briefly recall the definitions.

Let $\mathfrak{A}$ be a finite Boolean algebra of subsets of Σ. A (non deterministic) finite automaton on $\mathfrak{A}$ is a quintuple $\mathcal{A} = (Q, \Sigma, E, I, F)$ where Q is the finite set of states, I and F are the sets of *initial* and *terminal* (or *final*) states and $E \subseteq Q \times \mathfrak{A} \times Q$ is the finite set of *transitions*. Given a transition $(q, X, p) \in E$, the set X is called the *label* of the transition. A *run* is a sequence of the form

$$q_0 \xrightarrow{X_1} q_1 \dots q_{n-1} \xrightarrow{X_n} q_n$$

where $q_0 \in I$ and $(q_{i-1}, X_i, q_i) \in E$ for $i = 1, \dots, n$. Its *label* is the concatenation $X_1 \dots X_n \subseteq \Sigma^*$ with the convention that this product is equal to ε whenever $n = 0$. The run is *successful* if $q_n \in F$. The subset *recognized* by the automaton is the union, over all successful runs, of their labels. We denote by $\mathtt{Rec}(\mathfrak{A})$ the family of *regular* subsets of Σ^*, i.e., of subsets recognized by some finite automaton on the algebra $\mathfrak{A}$. The following closure properties of recognizable subsets are easy extensions of well-known results.

Proposition 1. *1. Let $\mathfrak{A}$ be finite Boolean algebras of subsets of Σ. Then for all $L, K \in \mathtt{Rec}(\mathfrak{A})$ we have $L \cup K \in \mathtt{Rec}(\mathfrak{A})$ and $\Sigma^* \setminus L \in \mathtt{Rec}(\mathfrak{A})$*
2. Let $\mathfrak{B}$ be a finite Boolean algebra of subsets of Σ containing $\mathfrak{A}$. If $L \in \mathtt{Rec}(\mathfrak{A})$ then $L \in \mathtt{Rec}(\mathfrak{B})$.

The second assertion of the next proposition is less standard. Its easy proof is left to the reader. It uses the following notation: given two finite Boolean subalgebras of subsets of two alphabets Σ and Δ, $\mathfrak{A} \otimes \mathfrak{B}$ denotes the Boolean subalgebra $\{X \times Y \mid X \in \mathfrak{A}, Y \in \mathfrak{B}\}$ of the alphabet $\Sigma \times \Delta$. Given two subsets $L \subseteq \Sigma^*$ and $M \subseteq \Delta^*$, we define $L \otimes M = (L \times M) \cap (\Sigma \times \Delta)^* = \{(u, v) \in L \times M \mid |u| = |v|\}$.

Proposition 2. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be two finite Boolean algebras of subsets of two alphabets Σ and Δ.*
1. Let h be a length preserving morphism of Σ^* into Δ^* (i.e., $h(\Sigma) \subseteq \Delta$). Assume that, for all $X \in \mathfrak{A}$ and all $Y \in \mathfrak{B}$, we have $h(X) \in \mathfrak{B}$ and $h^{-1}(Y) \in \mathfrak{A}$. Then, for all $L \in \mathtt{Rec}(\mathfrak{A})$ and all $M \in \mathtt{Rec}(\mathfrak{B})$, we have $h(L) \in \mathtt{Rec}(\mathfrak{B})$ and $h^{-1}(M) \in \mathtt{Rec}(\mathfrak{A})$.
2. If $L \in \mathtt{Rec}(\mathfrak{A})$ and $M \in \mathtt{Rec}(\mathfrak{B})$ then $L \otimes M \in \mathtt{Rec}(\mathfrak{A} \otimes \mathfrak{B})$.

2.3 Multitape automata, relations and predicates

We will show how to accomodate one-tape automata over infinite alphabets as just explained above, to multitape automata. This will allow us to introduce two families of multitape automata over infinite alphabets which we call multicell synchronous and asynchronous automata respectively. In the

spirit of [6], the finite subalgebras of the labels are defined via conditions of equality or inequality between the values of the scanned cells. For multicell synchronous automata, the different heads move synchronously along the tapes but have the possibility of scanning the last d cells preceding the current position, see figure 1. This extra feature does not increase the power of the synchronous automata when the alphabet is finite since the information can be stored and updated in the finite memory. The asynchronous automata are a natural extension of their counterpart for finite alphabets as introduced by Rabin & Scott in [16].

3 Multicell synchronous automata

We start with an informal introduction to the model. An n-tape multicell synchronous automaton is a generalization of an n-tape synchronous automaton, see [8]. It possesses n read heads, one for each tape, moving synchronously from left to right on the input. The special feature of multicell automata is that for some fixed integer $d \geq 1$, each head scans d cells on its tape, i.e., at time t it scans the letters at positions $t-d+1, t-d+2, \ldots, t$. Let's call *window* the $n \times d$ array of cells thus inspected. The automaton tests whether certain pairs of cells in the window are equal or different and whether or not some cells of the window are equal to some fixed letter among a finite predefined subset of the alphabet. Based on this piece of information, the automaton changes state and moves one position further.

The input of an n-tape automaton is an n-tuple of words over some fixed alphabet Σ. Words in this tuple may have different lengths, so that too short components lead to "missing letters" in the window. To cope with these missing letters, we introduce a padding symbol # which does not belong to the alphabet Σ. E.g, the tuple of words $(cabca, bba, aabcbc)$ is completed to $(cabca\#, bba\#\#\#, aabcbc)$. We also introduce another new padding symbol $\Box$ outside $\Sigma \cup \{\#\}$ for the virtual cells with negative positions left to the input. Figure 1 is an illustration of such an automaton with $n = d = 3$, at time $t = 5$ (imagine the window sliding along the input). The successive windows scanned by the heads at times $t = 1, \ldots, 6$ are shown on Figure 2.

c	a	b	c	a	$\#$	
b	b	a	$\#$	$\#$	$\#$	
a	a	b	c	b	c	

Figure 1: A configuration

$\Box$	$\Box$	c	$\Box$	c	a	c	a	b	a	b	c	b	c	a	c	a	$\#$
$\Box$	$\Box$	b	$\Box$	b	b	b	b	a	b	a	$\#$	a	$\#$	$\#$	$\#$	$\#$	$\#$
$\Box$	$\Box$	a	$\Box$	a	a	a	a	b	a	b	c	b	c	b	c	b	c

Figure 2: The successive windows

Remark 3. In fact, identifying the padding symbols $\Box$ and $\#$ would not cause any problem.

3.1 Formal definitions

The idea of synchronous automata (even if multicell) is to work with a free monoid, not a direct product of free monoids and thus to formalize the passage from the situation illustrated in Figure 1 to that of Figure 2. We first set $\mathbf{\Sigma} = \Sigma \cup \{\#, \Box\}$ and define the *array alphabet* as the set $\mathbf{\Sigma}^{n\times d}$. Then we consider the mapping $\tau_{n,d} : (\Sigma^*)^n \to (\mathbf{\Sigma}^{n\times d})^*$ which associates to an n-tuple of words $(u_1, \ldots, u_n) \in (\Sigma^*)^n$ a word $\delta^{(1)} \cdots \delta^{(\ell)} \in (\mathbf{\Sigma}^{n\times d})^*$ in such a way that $\ell = \max(|u_1|, \ldots, |u_n|)$ and, for $1 \leq t \leq \ell$, $1 \leq i \leq n$ and $1 \leq j \leq d$, we have

$$\delta_{i,j}^{(t)} = \begin{cases} \Box & \text{if} \quad t-(d-j) \leq 0 \\ u_{i,t-(d-j)} & \text{if} \quad 1 \leq t-(d-j) \leq |u_i| \\ \# & \text{if} \quad |u_i| < t-(d-j) \end{cases}$$

The transformation $\tau_{n,d}$ extends to subsets of $(\Sigma^*)^n$.

As discussed in paragraph 2.2, once we have a free monoid over an infinite alphabet, we need a finite Boolean subalgebra of subsets in order to define a finite automaton. Among the possible candidates, the following one seems natural.

Definition 4. *Given a finite subset $\Sigma_0 \subseteq \Sigma$, called the subset of* constants, *the* algebra of finitary labels $\mathfrak{F}^{n,d}_{\Sigma_0}$, *is the Boolean algebra of subsets of the array alphabet $\mathbf{\Sigma}^{n\times d}$ which are definable over $\mathbf{\Sigma} = \Sigma \cup \{\Box, \#\}$ by Boolean combinations of formulas of the form*

$$x_{i,k} = a \quad \textit{for } a \in \Sigma_0 \cup \{\Box, \#\}, \qquad \textit{and} \qquad x_{i,k} = x_{j,\ell}$$

where the variables $x_{i,k}$ have indexes (i,k) varying in $\{1,\dots,n\}\times\{1,\dots,d\}$.

In other words, it is the Boolean algebra of $(n \times d)$-ary relations on $\mathbf{\Sigma} = \Sigma \cup \{\Box, \#\}$ which are quantifier-free definable in the structure

$$\langle \mathbf{\Sigma}; =, (a)_{a\in\Sigma_0\cup\{\Box,\#\}} \rangle \tag{1}$$

Remark 5. It is worthwhile observing that the atoms of this algebra $\mathfrak{F}^{n,d}_{\Sigma_0}$ are the classes of the equivalence on $\mathbf{\Sigma}^{n\times d}$ defined by $\delta \approx^{\Sigma_0}_{n,d} \eta$ if and only if we have

$$\bigwedge_{\substack{1\le i,j\le n,\\ 1\le k,\ell\le d\\ a\in\Sigma_0\cup\{\Box,\#\}}} (\delta_{i,k} = a \Leftrightarrow \eta_{i,k} = a) \wedge (\delta_{i,k} = \delta_{j,\ell} \Leftrightarrow \eta_{i,k} = \eta_{j,\ell}))$$

With all these ingredients we can define the notion of multicell synchronous relation, without introducing a new type of automaton, since we are able to "reuse" the definition given in §2.2.

Definition 6. *1. A* d-cell, n-tape synchronous, Σ_0*-automaton is a finite automaton on the finite algebra $\mathfrak{F}^{n,d}_{\Sigma_0}$. The automaton is* multicell *(resp.* multitape*) when it is d-cell (resp. n-tape) for some d (resp. n).*
2. A relation $R \subseteq (\Sigma^*)^n$ is d-cell, n-tape, Σ_0-synchronous *if there exists a regular subset $L \in \mathtt{Rec}(\mathfrak{F}^{n,d}_{\Sigma_0})$ such that $R = \tau^{-1}_{n,d}(L)$. If $\mathcal{A}$ recognizes L, we also say by extension that it* recognizes R. *Furthermore, it is* constant-free *if $\Sigma_0 = \emptyset$. The terms multicell and multitape are used in the same way they are used for automata.*

3.2 A useful equivalence

The following is an adaptation to $(\Sigma^*)^n$ of the relation $\approx^{\Sigma_0}_{n,d}$ mentioned after Definition 4. It provides a necessary condition for a relation to be multicell synchronous and will help us show that some relations are not multicell synchronous. We denote by $\sim^{\Sigma_0}_{n,d}$ the equivalence on $(\Sigma^*)^n$ defined by

$(u_1, \ldots, u_n) \sim_{n,d}^{\Sigma_0} (v_1, \ldots, v_n)$ if and only if

$$\left.\begin{array}{l} |u_i| = |v_i| \wedge \\ (u_i[k] = a \Leftrightarrow v_i[k] = a) \wedge \\ (u_i[k] = u_j[\ell] \Leftrightarrow v_i[k] = v_j[\ell]) \end{array}\right\} \quad \begin{array}{l} \textit{for all } 1 \leq i, j \leq n, \\ a \in \Sigma_0 \textit{ and} \\ 1 \leq k, \ell \leq d \text{ s.t. } |k - \ell| < d \end{array} \tag{2}$$

Proposition 7. *Each n-ary d-cell synchronous relation $R \subseteq (\Sigma^*)^n$ is saturated under the equivalence $\sim_{n,d}^{\Sigma_0}$, i.e, $\vec{u} \in R$ and $\vec{u} \sim_{n,d}^{\Sigma_0} \vec{v}$ implies $\vec{v} \in R$*

Proof. If $(u_1, \ldots, u_n) \sim_{n,d}^{\Sigma_0} (v_1, \ldots, v_n)$ holds then we have $\tau_{n,d}(u_1, \ldots, u_n) = \delta^{(1)} \cdots \delta^{(\ell)}$ and $\tau_{n,d}(v_1, \ldots, v_n) = \eta^{(1)} \cdots \eta^{(\ell)}$ for some integer ℓ and $\delta^{(i)} \approx_{n,d}^{\Sigma_0} \eta^{(i)}$ for $i = 1, \ldots, \ell$. Since equivalence classes of $\approx_{n,d}^{\Sigma_0}$ are atoms of the finitary algebra $\mathfrak{F}_{\Sigma_0}^{n,d}$, the words $\delta^{(1)} \cdots \delta^{(\ell)}$ and $\eta^{(1)} \cdots \eta^{(\ell)}$ label the same runs. □

The multitape synchronous automata studied in [6] correspond to $d = 1$. They differ from the general case $d > 1$ by their closure properties. Indeed, they are closed under the Boolean operations, direct product and projection. In fact they are characterized by the fact that the relations which they recognize are precisely those that can be first-order defined in the structure of the free monoid with the predicates "prefix", "equal length" and "equal length along with same last letter".

3.3 Examples of multicell synchronous automata

All the three binary relations in the next examples are recognized by some multicell synchronous automata but cannot be recognized by any synchronous automaton (i.e., $d = 1$) as the reader may verify by applying for example Proposition 7. The predicate $\texttt{EqLast}(u, v)$ is interpreted as saying that u and v end with the same letter.

Example 8. The relation $\{(u, v) \in (\Sigma^*)^2 \mid |v| = |u| + 1 \wedge \texttt{EqLast}(u, v)\}$ is recognized by the following 2-cell synchronous constant-free automaton. Recall that the similar relation $\{(u, v) \in (\Sigma^*)^2 \mid |v| = |u| \wedge \texttt{EqLast}(u, v)\}$ is synchronous.

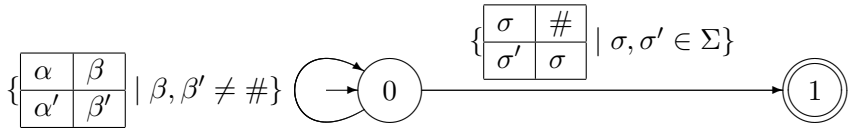


Figure 3: A 2-tape 2-cell synchronous constant-free automaton recognizing $\{(u, v) \in (\Sigma^*)^2 \mid |v| = |u| + 1 \wedge \texttt{EqLast}(u, v)\}$

Example 9. Our second example defines another binary relation. The second component of each pair in the relation is of the form σ^n for some $\sigma \in \Sigma$ and some $n \in \mathbb{N}$, i.e., it is a repetition of the same letter. It will be used in the proof of Theorem 24.

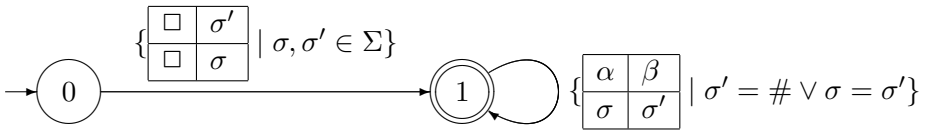
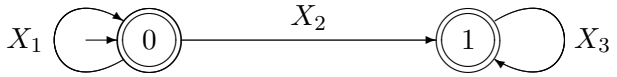


Figure 4: A 2-tape 2-cell synchronous constant-free automaton recognizing $\{(u, \sigma^n) \mid u \in \Sigma^*, \sigma \in \Sigma, n \in \mathbb{N}\}$

Example 10. The last example is a refinement of the previous one. The letter $a \in \Sigma_0$ is fixed. The second component of all pairs in the relation is taken advantage of in order to insure that the first component does not contain two ocurrences of σa for some arbitrary $\sigma \in \Sigma$. It will be used in the proof of Theorem 24.



$$X_1 = \{\begin{array}{|c|}\hline \Box\alpha \\ \hline \Box\beta \\ \hline \end{array} \mid \alpha, \beta \in \mathbf{\Sigma}\} \cup X_3 \ , \ X_2 = \begin{array}{|c|}\hline \sigma a \\ \hline \sigma\sigma \\ \hline \end{array} \ , \ X_3 = \{\begin{array}{|c|}\hline \alpha\beta \\ \hline \sigma\sigma \\ \hline \end{array} \mid \alpha\beta \neq \sigma a\}$$

Figure 5: A relation used in Theorem 24. Here $a \in \Sigma_0$.

3.4 Closure properties

We consider here natural operations on multicell multitape synchronous relations. Union and intersection require the relations to be of the same arity but not necessarily the same window width. The first technical result guarantees that, without loss of generality, we may start up with relations of the same window width.

Proposition 11. *If the relation R is d-cell synchronous then it is also $(d+1)$-cell synchronous.*

Proof. Let $\pi : (\mathbf{\Sigma}^{n,d+1})^* \to (\mathbf{\Sigma}^{n,d})^*$ be the morphism which deletes the first column of each array in $\mathcal{F}_{\Sigma_0}^{n,d+1}$. Then we have $\tau_{n,d} = \pi \circ \tau_{n,d+1}$. Let $L \subseteq$

$(\mathbf{\Sigma}^{n,d})^*$ be some regular language such that $R = \tau_{n,d}^{-1}(L) = \tau_{n,d+1}^{-1}(\pi^{-1}(L))$. By Proposition 2, we have $K = \pi^{-1}(L) \in \mathtt{Rec}(\mathcal{F}_{\Sigma_0}^{n,d+1})$, thus $R = \tau_{n,d+1}^{-1}(K)$ is d-cell synchronous. □

The same type of normalization can be made for the finite alphabet of constants. It is an immediate consequence of Proposition 1.

Proposition 12. *If the relation R is d-cell, n-tape Σ_0-synchronous then it is also d-cell, n-tape Σ_1-synchronous for all $\Sigma_1 \supseteq \Sigma_0$.*

The family of multicell synchronous relations enjoys the same closure properties as the family of ordinary synchronous relations, except closure under projections.

Theorem 13. *1. The family of multicell synchronous relations with fixed arity is closed under the Boolean operations.*
2. The family of multicell synchronous relations (with arbitrary arities) is closed under Cartesian product.

Proof. Let's write simply $\mathfrak{F}^{n,d}$ in place of $\mathfrak{F}_{\Sigma_0}^{n,d}$. Let $L, K \in \mathtt{Rec}(\mathfrak{F}^{n,d})$ such that $R = \tau_{n,d}^{-1}(L)$ and $S = \tau_{n,d}^{-1}(Y)$. Then $R \cup S = \tau_{n,d}^{-1}(L \cup K)$ holds. Now, since R is saturated under τ (i.e., $\vec{x} \in R$ and $\tau(\vec{x}) = \tau(\vec{y})$ implies $\vec{y} \in R$) we have $(\Sigma^*)^n \setminus X = \tau_{n,d}^{-1}((\mathfrak{F}^{n,d})^* \setminus R)$ which shows closure under Boolean operations.

Consider now $L \in \mathtt{Rec}(\mathfrak{F}^{n,d})$ and $K \in \mathtt{Rec}(\mathfrak{F}^{m,d})$ such that $R = \tau_{n,d}^{-1}(L)$ and $S = \tau_{m,d}^{-1}(K)$. Let A (resp. B) be the arrays in $\mathfrak{F}^{n,d}$ (resp. in $\mathfrak{F}^{m,d}$ having their last columns made of occurrences of #. Observe that $\mathfrak{F}^{n,d} \otimes \mathfrak{F}^{m,d} = \mathfrak{F}^{n+m,d}$. Then $R \times S = \tau_{n+m,d}^{-1}(LA^* \otimes KB^*)$, which completes the proof by Proposition 2. □

However, closure under projection fails.

Proposition 14. *Neither the relation*

$$\mathtt{EqLast} = \{(u, v) \in \Sigma^* \mid u \text{ and } v \text{ have the same last letter}\}$$

nor its complement is multicell synchronous though both are projections of 2-cell 3-tape constant-free synchronous relations.

Proof. Suppose first that the relation EqLast is d-cell Σ_0-synchronous for some finite Σ_0. Let $\sigma, \sigma', \sigma'' \in \Sigma \setminus \Sigma_0$ be distinct letters. Using the equivalence introduced in paragraph 3.2, we have $(\sigma^d\sigma', \sigma') \sim_{n,d} (\sigma^d\sigma'', \sigma')$, i.e,

$(\sigma^d\sigma', \sigma') \in \texttt{EqLast}$ if and only if $(\sigma^d\sigma'', \sigma') \in \texttt{EqLast}$, contradiction. An analog argument applies to the complement of `EqLast`.

Concerning the second claim, letting $\pi : (\Sigma^*)^3 \to \Sigma^*$ be the projection on the first two components, it is clear that `EqLast` and its complement are respectively equal to $\pi(A \cap C)$ and $\pi(B \cap C)$ where

$$A = \{(u, v, w) \mid |w| \geq \max(|u|, |v|) \wedge u[|u|] = w[|u|] \wedge v[|v|] = w[|v|]\}$$
$$B = \{(u, v, w) \mid |w| \geq \max(|u|, |v|) \wedge u[|u|] = w[|u|] \Leftrightarrow v[|v] \neq w[|v|]\}$$
$$C = \{(u, v, w) \mid \text{all letters of } w \text{ are the same}\}$$

It is clear that A, B are synchronous relations. Let X be the set of 3×2 arrays with last line (σ, σ) or $(\Box, \sigma)$ with $\sigma \in \Sigma$. It is clear that $X \in \mathfrak{F}_\emptyset^{3,2}$ and that the one state automaton with loop transition labelled by X recognizes C. Thus, C is constant-free 2-cell synchronous and so are $A \cap C$ and $B \cap C$. $\Box$

4 Asynchronous automata

An asynchronous n-tape automaton on an infinite alphabet Σ is nothing more than the natural extension of Rabin and Scott's multitape automata for words over finite alphabets, see [16].

Again, we need a finite Boolean subalgebra of subsets of the product of alphabets. The natural candidate is similar to $\mathfrak{F}_{\Sigma_0}^{n,1}$, except that the extra symbol $\Box$ is now useless and the extra symbol # is replaced by a nonempty subset of $\{1, \ldots, n\}$ to represent the support of the input, i.e., the indexes of its non empty components.

Definition 15. *Given a finite subset $\Sigma_0 \subseteq \Sigma$ and a non empty subset S of $\{1, \ldots, n\}$, the* algebra of finitary labels $\mathfrak{L}_{\Sigma_0}^{n,S}$ *is the Boolean algebra of subsets of Σ^S which are definable over Σ by Boolean combinations of formulas of the form*

$$x_i = a \quad \textit{for } a \in \Sigma_0, \qquad \textit{and} \qquad x_i = x_j$$

where the variables x_i have indexes i varying in S.

An *asynchronous n-tape Σ_0-automaton* on an infinite alphabet Σ is defined by a finite set Q of *states*, a set of *initial* states $I \subseteq Q$, a set of *final* states $F \subseteq Q$ and a set of transitions of the form (q, S, X, P, p) where $q, p \in Q$, $P \subseteq S \subseteq \{1, \ldots, n\}$ and $X \in \mathfrak{L}_{\Sigma_0}^{n,S}$.

The sole transitions (q, S, X, P, p) which occur in a run on an input $(w_1, \ldots, w_n) \in \Sigma^*$ are those for which $S = \{i \mid w_i \neq \varepsilon\}$. Let's describe the dynamic of the device by explaining what instruction the transition

(q, S, X, P, p) performs on input $(w_1, \ldots, w_n) \in \Sigma^*$. Each one of the components in S can be considered as a tape provided with a reading head positioned somewhere on the tape. Based on the occurrences of the components read by the heads in a current position, say $(a_i)_{i\in S} \in \Sigma^S$, if $(a_i)_{i\in S} \in X$, then the device changes from state q to state p and the i-th head moves to the next position of the i-th tape for all $i \in P$. All other heads stay at their current position. The run starts in an initial state, with all the reading heads positioned on the first occurrence of each component in S. If the run enters a final state when the reading heads are positioned on the last occurrence of each component in S, then the word is *recognized*. The set of all n-tuples of words recognized by the automaton is the relation recognized by the automaton. As usual, the n-tuple $(\varepsilon, \ldots, \varepsilon)$ is recognized just in case $I \cap F \neq \emptyset$.

Example 16. The 3-tape automaton of Figure 6 performs the concatenation, i.e., recognizes all the triples of the form (u, v, uv). For any such triple there is a unique accepting run. If $u = v = \varepsilon$ it is reduced to state 2. If $v = \varepsilon \neq u$, it starts in state 1 and ends in state 2. If $u = \varepsilon \neq v$ it starts in state 3 and ends in state 4. If $u, v \neq \varepsilon$ it starts in state 1 and ends in state 4.
In Figure 7, the left 2-tape automata recognize the `EqLast` relation and its complement and the right 2-tape automaton recognizes the relation $R = \{(\varepsilon, \varepsilon)\} \cup \{(\sigma, \sigma^n) \mid \sigma \in \Sigma, n \geq 1\}$.
The labels of the transitions are given as pairs (ϕ, P) rather than triples (S, X, P) : ϕ is a quantifier-free formula which defines X in the structure $\langle \Sigma; =, (a)_{a\in\Sigma_0} \rangle$ and S is any subset of $\{1, 2, 3\}$ which contains P.

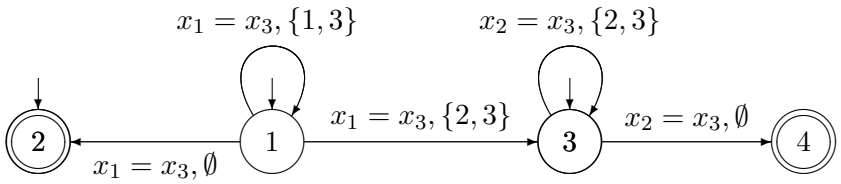


Figure 6: Concatenation: $\{(u, v, uv) \mid u, v \in \Sigma^*\}$

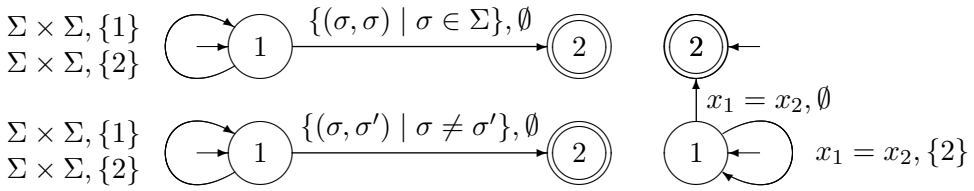


Figure 7: `EqLast`, its complement and $R = \{(\varepsilon, \varepsilon)\} \cup \{(\sigma, \sigma^n) \mid \sigma \in \Sigma, n \geq 1\}$

4.1 Closure properties of asynchronous relations

The asynchronous relations are not closed under projection since the set $\{\sigma^n \mid \sigma \in \Sigma, n \in \mathbb{N}\}$ is the projection of the asynchronous relation R in Figure 7 but is not itself asynchronous. Indeed, if σ, τ are two letters in $\Sigma \setminus \Sigma_0$ then they are in the same sets in $\mathfrak{F}^1_{\Sigma_0}$ (cf. Definition 15), so that any Σ_0-asynchronous automaton which recognizes $\sigma\sigma$ also recognizes $\sigma\tau$.
The usual counterexample for finite alphabet, namely, $\{(a^n, b^n c^n) \mid n \geq 0\} = \{(a^n, b^n c^p) \mid n, p \geq 0\} \cap \{(a^n, b^p c^n) \mid n, p \geq 0\}$ for some fixed $a, b \in \Sigma$, also shows that asynchronous relations are not closed under intersection either.

4.2 Incomparability of multicell synchronous and asynchronous relations

The families of multicell and asynchronous relations are incomparable since $\{\sigma^n \mid \sigma \in \Sigma, n \in \mathbb{N}\}$ is multicell but not asynchronous (see §4.1) and `EqLast` is asynchronous but not multicell by Proposition 14.

Nevertheless, we have the following easy property.

Proposition 17. *Let π be the projection of $(\Sigma^*)^{n\times d}$ onto $(\Sigma^*)^n$ defined by $\pi((x_{i,k})_{i=1,\dots,n,\ k=1,\dots,d}) = (x_{1,1}, \dots, x_{n,1})$. Then any n-ary multicell Σ_0-synchronous relation R is of the form $R = \pi(U \cap T)$ for some Σ_0-synchronous relation U and some Σ_0-asynchronous relation T.*

Proof. Suppose R is recognized by a d-cell Σ_0-synchronous automaton $\mathcal{A}_{mult}$. Let T be the relation recognized by the nd-ary asynchronous automaton $\mathcal{A}_{asyn}$ which works as follows:

(i) In a first phase, for $i = 1, \dots, n$ and $k = 1, \dots, d$, $\mathcal{A}_{asyn}$ positions the head of its $i(d-1)+k$-th tape on cell k. In case the input is empty or too short, its length (which is $< d$) is memorized in the state.

(ii) Afterwards, $\mathcal{A}_{asyn}$ always moves all heads ahead.

(iii) $\mathcal{A}_{asyn}$ mimics $\mathcal{A}_{mult}$, considering the nd cells scanned by its nd heads as an $n \times d$ array scanned by $\mathcal{A}_{mult}$ and changes states accordingly.

The synchronous nd-ary relation U is

$$U = \{(u_{i,k})_{i,k} \in (\Sigma^*)^{n\times d} \mid u_{i,1} = \ldots = u_{i,d} \textit{ for every } i = 1, \ldots, n\}$$

It is clear that the behavior of $\mathcal{A}_{asyn}$ on input $(u_{i,k})_{i,k} \in S$ is that of $\mathcal{A}_{mult}$ on input $(u_{1,1}, \ldots, u_{n,1}) \in (\Sigma^*)^n$. Which shows that $R = \pi(U \cap T)$. □

5 The three structures on Σ^*

Letting Σ_0 be a finite subset of Σ, we consider three structures on Σ^*.

(1) $\mathbf{S}^{\Sigma_0}_{\text{last}} = \langle \Sigma^*; \leq_{\texttt{pref}}, \texttt{EqLen}, \texttt{EqLast}, (\texttt{Last}_a)_{a\in\Sigma_0}\rangle$

(2) $\mathbf{S}^{\Sigma_0}_{\text{mult}} = \langle \Sigma^*; (R)_{R\in\mathcal{R}^{mult}_{\Sigma_0}}\rangle$,

(3) $\mathbf{S}^{\Sigma_0}_{\text{asyn}} = \langle \Sigma^*; (R)_{R\in\mathcal{R}^{asyn}_{\Sigma_0}}\rangle$,

where $\leq_{\texttt{pref}}$, $\texttt{EqLen}$, $\texttt{EqLast}$, $\texttt{Last}_a$ are predicates such that

- $x \leq_{\texttt{pref}} y$ is true if and only if x is a prefix of y,
- $\texttt{EqLen}(x, y)$ is true if and only if x and y have the same length,
- $\texttt{EqLast}(x, y)$ is true if and only if x and y have the same last letter,
- $\texttt{Last}_a(x)$ is true if and only if the last letter of x is a.
- $\mathcal{R}^{mult}_{\Sigma_0}$ is the collection of all multicell Σ_0-synchronous relations,
- $\mathcal{R}^{asyn}_{\Sigma_0}$ is the collection of all Σ_0-asynchronous relations.

To prove that these structures are equivalent, we compare their fragments defined by prefix conditions. Then we study the decidability status of such fragments.

5.1 The three structures are equivalent

The reason why the case of infinite alphabets differs from that of finite alphabets is that we may simulate indices of a given word x. We do not need the ordering of the indices, just the fact that an index is unique (we named such a word injective in [7]).

Let's expand the three structures $\mathbf{S}^{\Sigma_0}_{\text{last}}$, $\mathbf{S}^{\Sigma_0}_{\text{mult}}$ and $\mathbf{S}^{\Sigma_0}_{\text{asyn}}$ with the following predicates and functions which are easily expressible in the structures:

- $|x| \leq |y|$ has the obvious meaning,
- $\texttt{First}_a(x)$ means that the first letter of x is a,

- $\texttt{Pred}(x)$ is the prefix of x of length $|x|-1$ if it exists, else it is ε,
- $x|_y$ (resp. $x|_1$) is the prefix of x of length $|y|$ (resp. 1) if it exists, else it is ε,
- for all finite subsets $A \subseteq \Sigma$, the predicates $\texttt{Last}_A(x)$ and $\texttt{Last}_{\neg A}(x)$ respectively stand for $\bigvee_{a\in A} \texttt{Last}_a(x)$ and $\bigwedge_{a\in A} \neg\texttt{Last}_a(x)$

Let's stress a possible source of confusion in the notation used in point 4 of the next theorem: Σ_0 means a fixed subalphabet of the infinite alphabet whereas $\Sigma_n(\mathbf{S})$ (where $\mathbf{S}$ is some structure) means the family of relations definable in $\mathbf{S}$ by Σ_n formulas (i.e., formulas with $n-1$ alternations of quantifiers which start with existential quantifiers).

Theorem 18. *Let Σ be an infinite alphabet and $\Sigma_0 \subseteq \Sigma$. Consider the above expansions of $\mathbf{S}^{\Sigma_0}_{last}$, $\mathbf{S}^{\Sigma_0}_{mult}$ and $\mathbf{S}^{\Sigma_0}_{asyn}$.*

(1) $\texttt{EqLast}$ *and its complement are constant-free asynchronous and are both $\exists$ definable in $\mathbf{S}^{\Sigma_0}_{mult}$.*

(2) *Every multicell Σ_0-synchronous relation is both $\exists^*\forall$ and $\forall^*\exists$ definable in each structure $\mathbf{S}^{\Sigma_0}_{last}$ and $\mathbf{S}^{\Sigma_0}_{asyn}$.*

(3) *Every Σ_0-asynchronous relation is $\exists^*\forall^*$ definable both in $\mathbf{S}^{\Sigma_0}_{last}$ and in $\mathbf{S}^{\Sigma_0}_{mult}$.*

(4) *For $n \geq 1$* $\begin{cases} \Sigma_n(\mathbf{S}^{\Sigma_0}_{last}) \subseteq \Sigma_n(\mathbf{S}^{\Sigma_0}_{mult}) \subseteq \Sigma_n(\mathbf{S}^{\Sigma_0}_{asyn}) \subseteq \Sigma_{n+2}(\mathbf{S}^{\Sigma_0}_{last}) \\ \Sigma_n(\mathbf{S}^{\Sigma_0}_{mult}) \subseteq \Sigma_{n+1}(\mathbf{S}^{\Sigma_0}_{last}) \end{cases}$

In particular, $\mathbf{S}^{\Sigma_0}_{last}$, $\mathbf{S}^{\Sigma_0}_{mult}$ and $\mathbf{S}^{\Sigma_0}_{asyn}$ have the same first order definable relations.

Proof. 1. Apply Example 16 and Proposition 14.

2. Consider a d-cell n-tape Σ_0-synchronous automaton $\mathcal{A}$ with k states $q_1, \ldots, q_k$.
In case $\Sigma_0 \neq \emptyset$, the simplest way to encode states is to consider a sequence of $\lceil \log k \rceil$ symbols of Σ, each such symbol σ being considered as the Boolean truth value of the assertion $\sigma \in \Sigma_0$. In order not to exclude the case $\Sigma_0 = \emptyset$, we use another encoding: a state q_i is coded by any sequence of k symbols of Σ which contains exactly i distinct letters. Thus, a sequence of states $(q_{i_1}, \ldots, q_{i_N})$ is coded by any k-tuple of words $(s_1, \ldots, s_k)$ such that all s_i's have length N and, for $j = 1, \ldots, N$, the index i_j is equal to the number of different letters among $\{s_1[j], \ldots, s_k[j]\}$.
Let $\theta_i(\vec{s}, t)$ be the Boolean combination of formulas $\texttt{EqLast}(s_j|_t, s_{j'}|_t)$ expressing that there are exactly i different letters among the last letters of

the $s_j|_t$'s ($j = 1, \ldots, k$), i.e. that the state at time $|t|$ is q_i. Let $A(\vec{s})$ be the similar Boolean combinations insuring that the first state is initial and the last one is final.
Let $\mathbf{x}$ be an $n \times d$-array of variables $x_{i,j}$'s. Any $X \in \mathfrak{F}^{n,d}_{\Sigma_0}$ is the set of solutions of some Boolean combination $\phi_X(\mathbf{x})$ of formulas $x_{i,j} = a$, for $a \in \Sigma_0 \cup \{\Box, \#\}$, and $x_{i,j} = x_{i',j'}$. Going from letters in $\Sigma \cup \{\Box, \#\}$ to words in Σ^*, let $\Phi_X(u_1, \ldots, u_n, t)$ be obtained from ϕ by replacing equalities $x_{i,j} = \Box$ and $x_{i,j} = \#$ by condition $x_{i,j} = \varepsilon$ (cf. Remark 3) and replacing equalities $x_{i,j} = a$, for $a \in \Sigma_0$, and $x_{i,j} = x_{i',j'}$ about letters in Σ by conditions $\mathtt{Last}_a(u_i|\mathtt{Pred}^{(j-1)}(t))$ and $\mathtt{EqLast}(u_i|\mathtt{Pred}^{(j-1)}(t), u_{i'}|\mathtt{Pred}^{(j'-1)}(t))$ about the letters in the $n \times d$ window scanning cells $|t| - d + 1, \ldots, |t|$ of $u_1, \ldots, u_n$.
Using an existentially quantified tuple of variables $s_1, \ldots, s_k$ to code the sequence of states in a run, and a universally quantified t to encode (via its length) all computation steps, the existence of an accepting run of $\mathcal{A}$ on input $(u_1, \ldots, u_n)$ can be expressed as follows (recall E is the set of transitions):

$$\begin{aligned}\exists s_1, \ldots, s_k \; \forall t \; (|s_1| = \ldots = |s_k| = \max(|u_1|, \ldots, |u_n|) \; \wedge \; A(\vec{s}|_1) \wedge B(\vec{s}) \\ \wedge (2 \leq |t| \leq |s_1| + 1 \Rightarrow \\ \bigvee_{(q_i, X, q_{i'}) \in E} (\theta_i(\vec{s}|_{\mathtt{Pred}(t)}) \wedge \theta_{i'}(\vec{s}|_t) \wedge \Phi_X(u_1, \ldots, u_n, \mathtt{Pred}(t))))) \quad (3)\end{aligned}$$

This proves that any multicell Σ_0-synchronous relation R is $\exists^*\forall$ definable in $\mathbf{S}^{\Sigma_0}_{\mathrm{last}}$. Such a definition for the complement of R, which is still multicell Σ_0-synchronous, yields a $\forall^*\exists$ definition of R.
Applying Point 1, we get analog definitions of R in $\mathbf{S}^{\Sigma_0}_{\mathrm{asyn}}$.

3. Consider now an n-tape Σ_0-asynchronous automaton $\mathcal{A}$ with states $q_1, \ldots, q_k$ which recognizes a relation R. We reduce to the case where no tuple in R has an empty component, which allows to forget mentioning the support of the input in the transitions.
Due to asynchronicity, we can no more code a computation step by the sole length of some (universally quantified) word. A trickier coding is required. Denote by N the length of the computation. We shall simulate the ordered initial segment $\{1, \ldots, N\}$ where $N = \max(|u_1|, \ldots, |u_n|)$ by considering a word of length N, say $\pi = \gamma_1 \ldots \gamma_N$ to suggest "position", with pairwise different letters (an "injective" word in the terminology used in [6]), cf the figure below.
Letting T be the length of the computation (so $T \geq N$), we encode the

sequence of positions of the i-th head as a length T word h_i (the symbol "h" suggests a reading head). If the i-th head at time t is on cell j of the i-th tape then $h_i[k]$ is equal to γ_j. In the example below, the input words have lengths 4, 3 and 6, the head on the first tape moves at times $1, 2, 4$, that on the second tape moves at times $1, 6$ and that on the third tape moves at times $3, 5, 7, 8, 9$. No head moves during the last transition from state q_{10} to state q_{11}.
States are encoded as above with a k-tuple of words $\vec{s}$ of length T.

$\vec{s}$	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}	q_{11}
h_1	α	β	γ	γ	δ	δ	δ	δ	δ	δ	δ
h_2	α	β	β	β	β	β	γ	γ	γ	γ	γ
h_3	α	α	α	β	β	γ	γ	δ	λ	μ	μ
π	α	β	γ	δ	λ	μ					

Let's express the different conditions on the words $t, h_1, \ldots, h_n, s_1, \ldots, s_k$.

(1) π has length $\max(|u_1|, \ldots, |u_n|)$.

(2) Letters in π are pairwise distinct:

$$\forall \pi' \forall \pi'' \ ((\pi' \leq_{\texttt{pref}} \pi'' \leq_{\texttt{pref}} \pi \wedge \texttt{EqLast}(\pi', \pi'') \Rightarrow \pi' = \pi'')$$

(3) $|s_1| = \ldots = |s_k| = |h_1| = \ldots = |h_n|$, i.e., the s_j's and the h_i's all have the same length which is the computation length.

(4) $A(\vec{s})$, i.e., the first state is initial and the last one is final (cf. the above proof of point 2).

(5) h_i is obtained from some prefix of π by replacing each letter γ_j by a non empty block of letters γ_j. This is characterized by a recursive condition: (i) the first letter of h_i is γ_1, (ii) if γ_j occurs in h_i at rank ≥ 2 then it is preceded by γ_j or γ_{j-1}. Ignoring indexes, this can be expressed as

$$\begin{aligned} &\texttt{EqLast}(\pi|_1, h_i|_1) \\ &\quad \wedge \forall \pi' \ \forall h' \ ((|\pi'| \geq 2 \wedge \pi' \leq_{\texttt{pref}} \pi \wedge h' \leq_{\texttt{pref}} h_i \wedge \texttt{EqLast}(\pi', h')) \\ &\qquad \Rightarrow \ (\texttt{EqLast}(\texttt{Pred}(\pi'), \texttt{Pred}(h')) \vee \texttt{EqLast}(h', \texttt{Pred}(h')))) \end{aligned}$$

(6) the last position of the i-th head is equal to the length of u_i. This means that the last letter of h_i is $\gamma_{|u_i|}$, and, ignoring indexes, can be simply expressed as $\texttt{EqLast}(h_i, \pi|_{u_i})$.

Now, if $1 \leq t \leq T$, we want to find out which letter is scanned at time t by the i-th head. By definition of h_i, at time t the i-th head scans cell j where

γ_j is the t-th letter of h_i. Now, this t-th letter is the last letter of $h_i|_t$ and j is the length of the unique prefix π' of π such that $\texttt{EqLast}(\pi', h_i|_t)$. Thus, the wanted scanned letter is the last letter of $u_i|_{\pi'}$.
Any $X \in \mathfrak{F}^n_{\Sigma_0}$ is the set of solutions of some Boolean combination $\phi_X(x_1, \ldots, x_n)$ of formulas $x_i = a$, for $a \in \Sigma_0$, and $x_i = x_j$. Going from letters in Σ to words in Σ^*, let $\Phi_X(u_1, \ldots, u_n, \pi'_1, \ldots, \pi'_n)$ be obtained from $\phi_X(x_1, \ldots, x_n)$ by replacing equalities $x_i = a$, for $a \in \Sigma_0$, and $x_i = x_j$ about letters in Σ by conditions $\texttt{Last}_a(u_i|_{\pi'})$ and $\texttt{EqLast}(u_i|_{\pi'_i}, u_j|_{\pi'_j})$.
Letting $\Psi(\vec{u}, \vec{s}, \vec{h}, \pi)$ be the conjunction of formulas in (1)-(6) above, the existence of an accepting run of $\mathcal{A}$ on input $(u_1, \ldots, u_n)$ can be expressed as follows (recall E is the set of transitions):

$$\begin{aligned}
&\exists s_1, \ldots, s_k\ \exists h_1, \ldots, h_n\ \exists \pi\ (\Psi(\vec{u}, \vec{s}, \vec{h}, \pi)\ \wedge\ \forall h\ \forall \pi'_1, \ldots, \pi'_n \\
&\qquad ((2 \leq |h| \leq |h_1| \wedge \bigwedge_{i=1,\ldots,n} \pi'_i \leq_{\texttt{pref}} \pi \wedge \texttt{EqLast}(h_i|_{\texttt{Pred}(h)}, \pi'_i)) \\
&\quad \Rightarrow \bigvee_{(q_i, X, q_{i'}) \in E} (\theta_i(\vec{s}|_{\texttt{Pred}(h)}) \wedge \theta_{i'}(\vec{s}|_h) \wedge \Phi_X(u_1, \ldots, u_n, \pi'_1, \ldots, \pi'_n))))
\end{aligned}$$

This proves that any Σ_0-asynchronous relation R is $\exists^*\forall^*$ definable in $\mathbf{S}^{\Sigma_0}_{\text{last}}$. Using point 1, this proves that R is also $\exists^*\forall^*$ definable in $\mathbf{S}^{\Sigma_0}_{\text{mult}}$.

4. All stated inclusions are straightforward consequences of points 1–3. □

5.2 Relation to arithmetical definability

To complete Theorem 18, we characterize the relations first-order definable in any of the considered three structures in case Σ is countable.

The following definition follows a suggestion of Eilenberg & al. in [8], that was worked out in our paper [6].

Definition 19. *A relation $R \subseteq (\Sigma^*)^n$ is Σ_0-finitary if it is invariant under all morphisms $f : \Sigma^* \to \Sigma^*$ which act as permutations on Σ and as the identity on Σ_0.*

The following are basic notions of recursion theory, e.g., [15] and [17]. Letting Δ be a *finite* alphabet, recall that a relation R on $\mathbb{N}$ (resp. on Δ^*) is *arithmetical* if it satisfies any one of the two equivalent conditions (for words on finite alphabets, cf. Seibert, 1991 [19]):

(i) R is first-order definable with addition and multiplication (resp. concatenation),

(ii) R can be obtained from computable relations via a series of projections and complementations,

The notion extends to infinite alphabets in the following way. First it extends to $\mathbb{N}$ by saying that a relation $R \subseteq (\mathbb{N}^*)^n$, i.e., a relation over the set $\mathbb{N}^*$ of finite sequences of integers is arithmetical, if for any computable encoding of finite sequences of integers by integers, the relation can be viewed as an arithmetical relation on $\mathbb{N}$. As for an infinite alphabet Σ, we need an explicit bijection with $\mathbb{N}$.

Definition 20. *Let $f : \mathbb{N}^* \to \Sigma^*$ be a morphsim which defines a bijection from $\mathbb{N}$ to Σ. We say that $R \subseteq (\Sigma^*)^n$ is f-computable (resp. f-arithmetical) if $f^{-1}(R)$ is computable (resp. arithmetical).*

Theorem 21. *Suppose Σ is countable and $R \subseteq (\Sigma^*)^n$. The following conditions are equivalent:*

(1) *R is first-order definable in $\mathbf{S}^{\Sigma_0}_{asyn}$, $\mathbf{S}^{\Sigma_0}_{last}$ and $\mathbf{S}^{\Sigma_0}_{mult}$,*
(2) *R is Σ_0-finitary and f-arithmetical for every bijection $f : \mathbb{N} \to \Sigma$,*
(3) *R is Σ_0-finitary and f-arithmetical for some bijection $f : \mathbb{N} \to \Sigma$.*

Proof. (1)⇒(2). Clearly, $\leq_{\mathtt{pref}}$, `EqLen`, `EqLast` and the $\mathtt{Last}_a$'s, $a \in \Sigma_0$, are Σ_0-finitary and c-arithmetical. Now, the family of Σ_0-finitary relations and that of f-arithmetical relations are both closed under Boolean operations and projections. Thus, they contain all relations definable in $\mathbf{S}^{\Sigma_0}_{\mathrm{last}}$.

The proof of the implication (3)⇒(1) goes through five steps. We consider a finite alphabet $\Delta = \Sigma_0 \cup \{c, d\}$ where c, d are two fixed distinct letters in $\Sigma \setminus \Sigma_0$.

Step 1. R can be encoded as an arithmetical relation $S \subseteq (\Delta^*)^n$.

Consider the infinite prefix code $C = \Sigma_0 \cup \{c^k d \mid n \in \mathbb{N}\}$ and let $\varphi : C^* \to \Sigma^*$ be the bijective morphism such that $\varphi(a) = a$ if $a \in \Sigma_0$ and $\varphi(c^k d) = f(p)$ where p is the $k+1$-st integer in $f^{-1}(\Sigma \setminus \Sigma_0)$. Set $S = \varphi^{-1}(R) = \{(v_1, \ldots, v_n) \mid (\varphi(v_1), \ldots, \varphi(v_n)) \in R\}$. We now show that the relation S is arithmetical. We set $f(s) = c$ and $f(t) = d$ and let $\lambda : \mathbb{N}^* \mapsto \mathbb{N}^*$ be the arithmetical morphism defined on the set $\mathbb{N}$ of generators by

$$\lambda(p) = \begin{cases} p & \text{if } f(p) \in \Sigma_0 \\ s^k t & \text{if } p \text{ is the } k+1\text{-th integer in } \mathbb{N} \setminus f^{-1}(\Sigma_0) \end{cases}$$

Furthermore the following diagram commutes

$$\begin{array}{ccc} \mathbb{N}^* & \xrightarrow{\lambda} & \mathbb{N}^* \\ \downarrow f & & \downarrow f \\ \Sigma^* & \xrightarrow{\varphi^{-1}} & \Delta^* \end{array}$$

We claim that the relation S is f-arithmetical. Indeed, since it is included in $(\Delta^*)^n$ and therefore in $(\Sigma^*)^n$ we may compute its inverse image by f:

$$f^{-1}(S) = f^{-1}(\varphi^{-1}(R)) = \lambda f^{-1}(R)$$

Now $f^{-1}(R)$ is arithmetical and thus so is $\lambda f^{-1}(R) = f^{-1}(S)$. Let P be the finite subset $f^{-1}(\Sigma_0 \cup \{c, d\})$ of $\mathbb{N}$ and let g the restriction of f to P, which is arithmetical. We have $g(f^{-1}(S)) = S$ which shows that S is also arithmetical.

Step 2. S is definable in $\mathbf{S}^{\Delta}_{\mathrm{last}}$.

Indeed, since Δ is a finite alphabet, it is known (Seibert, [19] Theorem 4.4) that any arithmetical relation over Δ^* is definable in the structure $\langle \Delta^*, =, \cdot \rangle$ (where $\cdot$ is concatenation). Thus, S is definable in $\langle \Delta^*, =, \cdot \rangle$. Now, when Σ is infinite, concatenation in Σ^* is definable in $\mathbf{S}^{\emptyset}_{\mathrm{last}}$ (cf. Theorem 5 in our paper [6]). Since $\Delta \subset \Sigma$ is finite, Δ^* is definable in $\mathbf{S}^{\Delta}_{\mathrm{last}}$.

Step 3. R can be recovered from S with no use of φ or f.

Let $\mathfrak{S}_{\Sigma_0}$ (resp $\mathfrak{C}_{\Sigma_0}$) be the family of bijections $h : \Sigma^* \to \Sigma^*$ (resp. $\psi : C^* \to \Sigma^*$) which are the identity on Σ_0. Observe that $\{h \circ \varphi \mid h \in \mathfrak{S}_{\Sigma_0}\} = \mathfrak{C}_{\Sigma_0}$. Since R is Σ_0-finitary, for every $h \in \mathfrak{S}_{\Sigma_0}$, we have $h(R) = R$ hence $h(\varphi(S)) = R$. Thus, $\psi(S) = R$ for every $\psi \in \mathfrak{C}_{\Sigma_0}$. For all integers p define $C_p = \Sigma_0 \cup \{c^k d \mid k < p\}$ and let $\mathfrak{C}^p_{\Sigma_0}$ be the family of injective morphisms $C_p^* \to \Sigma^*$ which are the identity on Σ_0. If $|\vec{v}|$ denotes $\max(|v_1|, \ldots, |v_n|)$, we have

$$R = \bigcup_{\psi \in \mathfrak{C}_{\Sigma_0}} \psi(S) = \bigcup_{\vec{v} \in S} \{\psi(\vec{v}) \mid \psi \in \mathfrak{C}_{\Sigma_0}\} = \bigcup_{\vec{v} \in S} \{\theta(\vec{v}) \mid \theta \in \mathfrak{C}^{|\vec{v}|}_{\Sigma_0}\} \qquad (4)$$

Let Inj_p be the set of length p words in $(\Sigma \setminus \Sigma_0)^*$ whose letters are pairwise different. Such a word $\xi = \sigma_1 \ldots \sigma_{p-1}$ encode the the morphism $\psi_\xi \in \mathfrak{C}^p_{\Sigma_0}$ satisfying $\sigma_k = \psi_\xi(c^k d)$ for $k = 0, \ldots, p-1$. Thus, (4) can be rewritten

$$R = \{\vec{u} \mid \exists \vec{v}\ \exists \xi\ (\vec{v} \in S \wedge \xi \in Inj_{|\vec{v}|} \wedge \bigwedge_{i=1,\ldots,n} u_i = \psi_\xi(v_i)\} \qquad (5)$$

Step 4. Define R in the logic $\mathbf{S}^{\Delta}_{\mathrm{last}}$.

Let's express the right handside of (5) in $\mathbf{S}^{\Delta}_{\mathrm{last}}$.

The conditions that all letters of ξ are distinct and that $|\xi| = |\vec{v}|$ can be clearly expressed. As for equalities $u_i = \psi_\xi(v_i)$, they can be expressed as $\exists\eta\ \exists w\ T(u_i, v_i, \xi, \eta, w)$ where $T(u, v, \xi, \eta, w)$ is the conjunction of the following conditions. Let A_ξ be Σ with all letters in Σ_0 or in ξ removed.

(i) If $v = z_0\ c^{\ell_1}d\ z_1\ c^{\ell_2}d\ z_2 \dots c^{\ell_p}d\ z_p$ with the z_j's in Σ_0^* (recall v is in C^*) then $w = z_0\ \sigma_1^{\ell_1}\tau_1\ z_1\ \sigma_2^{\ell_2}\tau_2\ z_2 \dots \sigma_p^{\ell_p}\tau_p\ z_p$ where σ_k is the $k+1$-st letter of ξ and $\tau_k \in A_\xi$. In particular, $|w| = |v|$. We also require that the τ_k's are pairwise distinct.

(ii) $\eta = z_0\ \tau_1\ z_1\ \tau_2 \dots \tau_p\ z_p$

(iii) For $1 \le j \le |u|$, if $\eta[j] \in \Sigma_0$ then $u[j] = \eta[j]$. Else, $u[j] = w[|w'| - 1]$ where w' is the prefix of w with last letter $\eta[j]$.

We now express these conditions in $\mathbf{S}^{\Delta}_{\mathrm{last}}$.

For condition (i), first, state $|w| = |v|$ and that z_j's are preserved when going from v to w :

$$\forall v_1 \le_{\mathtt{pref}} v\ \forall w_1 \le_{\mathtt{pref}} w\ (|v_1| = |w_1| \Rightarrow \bigwedge_{a\in\Sigma_0} \mathtt{Last}_a(v_1) \Leftrightarrow \mathtt{Last}_a(w_1))$$

Concerning the passage from the $c^k d$'s in v to the $\sigma^k\tau$'s in w we write

$$\begin{array}{l}\forall v_1, v_2, v_3\ \forall w_1, w_2, w_3\ \forall \tau \in \Sigma\ \exists \xi' \le_{\mathtt{pref}} \xi\ \exists \sigma \in \Sigma \setminus \Sigma_0 \\ ((v = v_1 v_2 d v_3 \wedge v_2 \in c^* \wedge \neg\mathtt{Last}_c(v_1) \wedge w = w_1 w_2 \tau w_3 \wedge \textstyle\bigwedge_{i=1,2,3} |v_i| = |w_i|) \\ \qquad \Rightarrow (|\xi'| = |v_2| \wedge \mathtt{EqLast}(\sigma, \xi') \wedge w_2 \in \sigma^* \wedge \neg\mathtt{EqLast}(\sigma, w_1) \wedge \tau \in A_\xi))\end{array}$$

The sole remaining condition from (i) is the requirement that the τ_k'be pairwise distinct in w : express that if the last letters of two distinct prefixes of w are in A_ξ then they are distinct,

For condition (ii), express that

- η and w have the same prefixes in Σ_0^* and the same factors in $A_\xi\Sigma_0^*$,
- η and w have the same letters in A_ξ and they occur in the same order.

Condition (iii) is straightforward to express. Thus, T, hence also R, is definable in $\mathbf{S}^{\Delta}_{\mathrm{last}}$.

Step 5. Define R in the logic $\mathbf{S}^{\Sigma_0}_{\mathrm{last}}$.

We must go from $\mathbf{S}^{\Delta}_{\mathrm{last}} = \mathbf{S}^{\Sigma_0\cup\{c,d\}}_{\mathrm{last}}$ to $\mathbf{S}^{\Sigma_0}_{\mathrm{last}}$. Observe that φ and S depend on c, d, and so does the obtained definition of R. But this is uniform in c, d, i.e., $R = \{\vec{u} \mid \Phi(\vec{u}, c, d)\}$

Since c, d are arbitrary, we have $R = \{\vec{u} \mid \exists c, d \in \Sigma \setminus \Sigma_0 \ (c \neq d \wedge \Phi'(\vec{u}, c, d))\}$ where Φ' is obtained from Φ by replacing any occurrence of the predicates $\mathtt{Last}_c(x)$ and $\mathtt{Last}_d(x)$ by $\mathtt{EqLast}(x, c)$ and $\mathtt{EqLast}(x, d)$. This is a definition in $\mathbf{S}^{\Sigma_0}_{\mathrm{last}}$.

□

5.3 The existential fragment of $\mathbf{S}_{\mathbf{mult}}$ is decidable

Here we study fragments of the first order structure $\mathbf{S}_{\mathrm{mult}}$. First, we show that the existential fragment is decidable. Later, in §5.4 we show that the $\exists\forall$-fragment is not.

Since the class of multicell synchronous relations is closed uner the Boolean operations, a formula of the existential fragment of $\mathbf{S}_{\mathrm{mult}}$ is of the form

$$\exists x_1, \ldots \exists x_n \phi(x_1, \ldots, x_n)$$

where ϕ is defined by a multicell synchronous relation. In other words, the decidability of the existential fragment is equivalent to the decidability of the emptiness problem for multicell relations which we assume specified by some multicell synchronous automaton. We prove this decidability problem by reducing it to the same problem for finite alphabets. Indeed, in that latter case, multicell synchronous relations are equivalent to synchronous relations and the result follows from the decidability problem in the underlying graph of the automaton.

We follow notation introduced in §2.1..

Lemma 22. *Let Σ_0 be a finite subalphabet of an infinite alphabet Σ. Then there exists a subset $\Sigma_1 \subseteq \Sigma \setminus \Sigma_0$ with nd letters such that all equivalence classes of the equivalence defined in paragraph 3.2 has a representative in $\Sigma_0 \cup \Sigma_1$.*

Proof. Consider any subset of nd letters in $\Sigma \setminus \Sigma_0$ and order its elements $a_1 < \ldots < a_{nd}$. Consider an n-tuple $(u_1, \ldots, u_n)$. We show how to define an n-tuple $(v_1, \ldots, v_n)$ such that

$$(u_1, \ldots, u_n) \sim_{n,d} (v_1, \ldots, v_n) \text{ and } (v_1, \ldots, v_n) \in (\Sigma_0 \cup \Sigma_1)^*$$

Set $N = \max\{|u_i| \mid 1, \ldots, n\}$. We show by induction on ℓ that for all $d \leq \ell \leq N$ there exist $(w_1, \ldots, w_n)$ such that

$$\begin{array}{l}(u_1[1 \ldots \ell], \ldots, u_n[1 \ldots \ell]) \sim_{n,d} (w_1, \ldots, w_n) \\ \text{and } (w_1, \ldots, w_n) \in (\Sigma_0 \cup \Sigma_1)^*.\end{array} \tag{6}$$

Consider first $\ell = d$ and the n-tuple $(u_1[1 \dots \ell], \dots, u_n[1 \dots \ell])$. Let $K \leq nd$ be the number of different letters outside Σ_0 which occur in $u_1[1 \dots \ell]$, …, $u_n[1 \dots \ell]$. Define $(w_1, \dots, w_n)$ from $u_1[1 \dots \ell]$, …, $u_n[1 \dots \ell]$ by substituting $a_1, \dots, a_K$ to these letters. Conditions (2) of § 3.2 are clearly satisfied.

Now, assume condition (6) holds for some $d \leq \ell \leq N$. We want to prove that for some $z_1, \dots, z_n \in (\Sigma_0 \cup \Sigma_1)^*$ we have $(u_1[1 \dots \ell+1], \dots, u_n[1 \dots \ell+1]) \sim_{n,d} (z_1, \dots, z_n)$. If $|u_i| \leq \ell$ then set $z_i = w_i$. For all other components, we set $z_i = w_i\sigma_i$ for some letters σ_i that we now define. Condition (6) for $\ell+1$ differs from that for ℓ as concerns the $n \times d$ window with right side at rank $\ell+1$. This window scans letters with ranks $\ell - d + 2, \dots, \ell + 1$. In the words $u_1[\ell - d + 2 \dots \ell], \dots, u_n[\ell - d + 2 \dots \ell])$, there are at most $n(d-1)$ different letters outside Σ_0. Let $\Sigma_2 \subset \Sigma_1$ be the letters of $w_1, \dots, w_n$ in the corresponding positions. Let $I \subseteq \{1, \dots, n\}$ be the set of the indices where the letters $u_i[\ell+1]$ differ from those occurring in $u_1[\ell-d+2 \dots \ell], \dots, u_n[\ell-d+2 \dots \ell])$ and let m be the cardinality of I. Since $\Sigma_1 \setminus \Sigma_2$ contains at least $nd - n(d-1) = n$ letters, it suffices to assign the m first letters of $\Sigma_1 \setminus \Sigma_2$ to the wanted σ_i's, $i \in I$. □

Theorem 23. *There exists an algorithm which, given a formula*

$$\exists x_1, \dots \exists x_n \phi(x_1, \dots, x_n)$$

decides whether or not it holds.

Proof. Because of the Boolean closure of the family of multicell multitape relations, we may assume that the formula ϕ defines a multicell, multitape relation. The problem reduces thus to determining whether or not an d-cell, n-tape Σ_0-synchronous automaton $\mathcal{A}$ recognizes an n-tuple. Let Σ_1 be a family of nd letters in $\Sigma \setminus \Sigma_0$. Let $\mathcal{A}'$ be obtained from $\mathcal{A}$ by substituting for each label of a transition, its intersection with the finite alphabet $\Sigma_0 \cup \Sigma_1$. By the previous lemma, there exists an n-tuple $(w_1, \dots, w_n) \in \Sigma^*$ such that $\tau_{n,d}(w_1, \dots, w_n)$ is recognized by $\mathcal{A}$ if and only if there exists an n-tuple $(w_1, \dots, w_n) \in (\Sigma_0 \cup \Sigma_1)^*$ such that $\tau_{n,d}(w_1, \dots, w_n)$ is recognized by $\mathcal{A}'$. But this latter problem amounts to determining whether or not a synchronous automaton on a given finite alphabet accepts some nonempty relation, which is decidable. □

5.4 The $\exists\forall$-fragment of $\mathbf{S_{mult}}$ is undecidable

Here we show exactly where the bound lies between decidable and undecidable fragments.

Theorem 24. *There exists no algorithm which, given a sentence of the form $\exists x \forall y \phi(x, y)$ where $\phi(x, y)$ is a first-order formula of the theory $\boldsymbol{S}_{mult}$, decides whether or not it holds.*

Proof. The idea of our proof is to encode the Post Correspondence Problem (PCP), which is known to be undecidable. This is done in a way similar to that used in our paper [7]. Recall that

An instance $\pi = \{(u_1, v_1), \ldots, (u_k, v_k)\}$ of PCP for words in $\{a, b\}^*$ is a finite subset of pairs of non empty words in $\{a, b\}^*$.

A non-trivial solution of the PCP for π is a non-empty word w which can be factorized as $w = u_{i_1} \ldots u_{i_r} = v_{i_1} \ldots v_{i_r}$ for some sequence $(i_1, \ldots, i_r) \in \{1, \ldots, k\}^*$.

We consider pairwise distinct markers $\gamma_1, \ldots, \gamma_{r+1}$ and encode these two factorizations as follows. Let $w_1 \ldots w_m$ be the coarsest factorization of w refining the previous two factorizations. For each w_i there are three possible cases: either it is a prefix of an occurrence u_{i_s} solely, or it is a prefix an occurrence of v_{i_ℓ} solely or it is a prefix of an occurrence u_{i_s} and of an occurrence of v_{i_ℓ}. We set $\omega = z_1 w_1 \ldots z_m w_m \gamma_{r+1} a \gamma_{r+1} b$ where for $1 \leq i \leq m$ we have $z_i = \gamma_{i_s} a$ in the first case and $z_i = \gamma_{i_\ell} b$ in the second case and $z_i = \gamma_{i_s} a \gamma_{i_\ell} b$ in the last case. Let's illustrate the process on the instance $\pi = \{(a, aba), (baaab, a)\}$ which has the solution $w = a\ baaab\ a = aba\ a\ aba$. The coarsest factorization which refines both factorizations is $w = a\ ba\ a\ ab\ a$. The encoding of w is $\omega = (\gamma_1 a)(\gamma_1 b)a(\gamma_2 a)ba(\gamma_2 b)a(\gamma_3 b)ab(\gamma_3 a)a(\gamma_4 a)(\gamma_4 b)$.

With this interpretation of the different γ's, a word ω encodes a solution w of the PCP for π if and only if it satisfies the following conditions where $\Gamma = \Sigma \setminus \{a, b\}$:

(i) *Non trivial.* $|\omega| \geq 9$

(ii) *Start.* There exists $\gamma \in \Gamma$ such that $\gamma a \gamma b$ is a prefix of ω.

(iii) *End.* There exists $\gamma' \in \Gamma$ such that $\gamma' a \gamma' b$ is a suffix of ω.

(iv) *Markers.* Every occurrence in ω of a letter in Γ is immediately followed by an occurrence of a or b.

(v) *Markers are distinct.* No factor in Γa or Γb occurs twice in ω.

(vi) *Inductive step of a backward decomposition of ω.*
If $\gamma \in \Gamma$ and $x\gamma a$ and $y\gamma b$ are both prefixes of ω then either $x = \varepsilon$ and $y = \gamma a$ or there exists $(u_i, v_i) \in \pi$ and u_i', v_i' and $\gamma' \in \Gamma$ such that
- $\gamma' a u_i'$ is a suffix of x and $\gamma' b v_i'$ is a suffix of y
- $u_i' \in (\Gamma b \cup \{\varepsilon\}) u_{i,1} \Gamma b u_{i,2} \ldots \Gamma b u_{i,m_i}$ where $u_i = u_{i,1} u_{i,2} \ldots u_{i,m_i}$ and

the $u_{i,j}$'s are $\neq \varepsilon$,
• $v'_i \in v_{i,1}\Gamma a v_{i,2} \dots \Gamma a v_{i,n_i}(\Gamma a \cup \{\varepsilon\})$ where $v_i = v_{i,1}v_{i,2}\dots v_{i,n_i}$ and the $v_{i,j}$'s are different from ε.
Observe that $m_i \leq |u_i|$ and $n_i \leq |v_i|$ since the $u_{i,j}$'s and $v_{i,j}$'s are $\neq \varepsilon$.
Thus, the lengths of u'_i, v'_i are bounded by $3|\pi| = 3\max(|u_1|, |v_1|, \dots, |u_k|, |v_k|)$.

We first consider the following relations where for $u \in \Sigma^*$ and $c = a, b$, $\mu_c(u)$ stands for the word obtained by removing from u all factors of the form γc where $\gamma \in \Gamma$.

$$\begin{aligned}
B &= \{(\omega, \gamma^{|\omega|}) \mid \omega \in \Sigma^*, \gamma \in \Gamma\} \\
B_c &= \{(\omega, \gamma^{|\omega|}) \in B \mid \gamma \in \Gamma \text{ and } \gamma c \text{ occurs at most once in } \omega\} \quad \text{for } c = a, b \\
B_{ab} &= \{(\omega, \gamma^{|\omega|}) \in B \mid \gamma a \text{ occurs in } \omega \text{ if and only if so does } \gamma b\} \\
B_c^u &= \{(\omega, \gamma^{|\omega|}) \in B \mid \text{if a prefix of } \omega \text{ ends with } \gamma c \text{ then it ends with} \\
&\qquad \gamma' u' \gamma c \text{ where } |u'| \leq 3|\pi|,\ \mu_b(u') = u \text{ and } \gamma' \in \Gamma\}
\end{aligned}$$

Denote by $\phi_1(\omega)$ the formula on ω defined by the conditions (i) – (iv) and by $\phi_2(\omega)$ and $\phi_3(\omega)$ the formulas defined by the conditions (v) and (vi) respectively. Then we have

$$\begin{aligned}
\phi_2(\omega) &\equiv \forall y\ ((\omega, y) \in B \Rightarrow (\omega, y) \in B_a \cap B_b) \\
\phi_3(\omega) &\equiv \forall y((\omega, y) \in B \Rightarrow ((\omega, y) \in B_{ab}\ \wedge \bigvee_{(u,v)\in\pi} (\omega, y) \in B_a^u \cap B_b^v)
\end{aligned}$$

Then PCP has a solution if and only if the sentence $\exists x \phi_1(x) \wedge \phi_2(x) \wedge \phi_3(x)$ holds. Let us verify that it belongs to the $\exists\forall$-fragment of the theory. Clearly $\phi_1(x)$ defines a multicell synchronous relation. In order to prove the theorem it suffices to verify that the relations B, B_c for $c = a, b$, B_{ab} and for B_c^u for $c = a, b$ and $u \in \Sigma^* - \{\varepsilon\}$ are multicell too. Examples 9 and 10 show that B and B_c for $c = a, b$ are multicell synchronous relations which settles the case of $\phi_2(x)$. Let Θ_c for $c = a, b$ be the subalphabet of $\mathfrak{F}^{2,2}$ consisting of all arrays of the form $\boxed{\begin{matrix}\gamma c \\ \gamma\gamma\end{matrix}}$ where $\gamma \in \Gamma$. We have $B_{ab} = B \cap \tau_{2,2}^{-1}(M)$ where $M \in \mathtt{Rec}F$ is defined by (we set $F = \mathfrak{F}^{2,2^*}$)

$$M = (F\Theta_a F \cap F\Theta_b F) \cup ((F \setminus F\Theta_a F) \cap (F \setminus F\Theta_b F))$$

Concerning B_c^u for $c = a, b$, it is the inverse image under $\tau_{2,3|\pi|+3}$ of the subset in $\mathtt{Rec}\mathfrak{F}_{\Sigma_0}^{2,3|\pi|+3}$ consisting of all words written on the subalphabet of arrays whose top line does not end in Γc and all words containing an array of the form $\boxed{\begin{matrix}z\gamma c \\ \gamma^{3|\pi|+3}\end{matrix}}$ where $\gamma \in \Gamma$ and $z \in \Sigma^*$ is such that $|z| = 3|\pi| + 1$. □

5.5 Decision issues of the structure $\mathbf{S}_{\mathbf{async}}$

In this last paragraph we prove the assertions of Table 1 about the decision problem in the structure $\mathbf{S}_{\text{async}}$.

Proposition 25. *1. The family of true formulas of the form*

$$\exists x_1 \dots \exists x_n \; R(x_1, \dots, x_n)$$

where $n \geq 1$ and R is an n-ary asynchronous relation over words in an infinite alphabet, is decidable.
2. The family of true formulas of the form $\exists x \; R(x, x)$, where R is a binary asynchronous relation, is undecidable.
3. The family of true formulas of the form $\forall x \; R(x, x)$, where R is a binary asynchronous relation, is undecidable.

Proof. 1. This is the decidability of the emptyness problem, which reduces to an accessibility problem in a finite graph.
2. We adapt an old result stating the undecidability of the disjointness problem for asynchronous relations, see [10] (the fact that the alphabet is infinite is irrelevant). To any instance $\pi = \{(u_1, v_1), \dots, (u_k, v_k)\}$ of the Post correspondence problem, associate the following ternary asynchronous relation R_π : $R_\pi(x, y)$ is true if (x, y) is a non empty product of pairs of words in π, i.e., x, y are of the form $x = u_{i_1} \dots u_{i_p}$ and $y = v_{i_1} \dots v_{i_p}$. It is clear that $\exists x \; R(x, x)$ is true if and only if the instance π of the PCP has a solution.
3. We shall use the following result from [13] (again, the fact that the alphabet is infinite is irrelevant). Given a binary asynchronous relation R, it is undecidable to test whether $R(x, x)$ holds for every x in the domain of R. That is, the family of true formulas $\forall x \; ((\exists z \; R(x, z)) \Rightarrow R(x, x))$ is undecidable. Now, observe that $\{x \mid \exists z \; R(x, z)\}$ is regular, so that $\{(x, y) \mid (\exists z \; R(x, z)) \Rightarrow R(x, y)\}$ is also an asynchronous relation. Thus, the family of true formulas $\forall x \; S(x, x)$, S an asynchronous binary relation, is also undecidable. □

5.6 Multicell asynchronous automata

A natural common generalization of the ideas underlying multicell synchronous automata and asynchronous automata is to enrich asynchronous automata as follows: on each tape, the head scans d consecutive cells.

This notion of multicell asynchronous automaton is strictly more general than the two notions considered before. For instance, $\{(\sigma', \sigma^n \sigma') \mid \sigma, \sigma' \in \Sigma, n \in \mathbb{N}\}$ is multicell asynchronous but neither multicell synchronous nor asynchronous.

Closure of multicell asynchronous relations under intersection and complement fail as they do with asynchronous relations. Closure under projection fails as it does with multicell synchronous relations.

It is routine to extend Theorem 18 to the structure with multicell asynchronous relations.

References

[1] D. Angluin and D.N. Hoover. Regular prefix relations. *Mathematical Systems Theory*, 17(3):167–191, 1984.

[2] M. Benedikt, L. Libkin, T. Schwentick, and L. Ségoufin. Definable relations and first-order query languages over strings. *Journal of the Association of Computing Machinery*, 50:694–751, 2003.

[3] A. Bès. An application of the Feferman-Vaught theorem to automata and logics for words over an infinite alphabet. *Logical Methods in Computer Science*, to appear.

[4] M. Bojanczyk, A. Muscholl, T. Schwentick, L. Segoufin and C. David. Two-variable logic on words with data. *In Proceedings of LICS'06*, 7–16, 2006.

[5] C. Choffrut. Relations over words and logic: a chronology. *Bulletin of the EATCS*, 89:159–163, June 2006.

[6] C. Choffrut and S. Grigorieff. Finite n-tape automata over possibly infinite alphabets: Extending a theorem of eilenberg et al. *Theoretical Computer Science*, 2008. Available online 31 July 2008.

[7] C. Choffrut and S. Grigorieff. The decision problem for some logics for finite words on infinite alphabets. 2008. Notes of scientific seminars of POMI, Special volume dedicated to Yuri Matiyasevich's 60th birthday.

[8] S. Eilenberg, C.C. Elgot, and J.C. Shepherdson. Sets recognized by n-tape automata. *Journal of Algebra*, 3:447–464, 1969.

[9] C. C. Elgot and J. E. Mezei. On Relations Defined by Finite Automata. *IBM Journal*, 10:47–68, 1965.

[10] P. C. Fischer and A. L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2:88–101, 1968.

[11] T. Harju and J. Karhumäki. Decidability of the Multiplicity Equivalence of Multitape Finite Automata. *In Proceedings of STOC 1990*, 477-481.

[12] T. Harju and J. Karhumäki. The Equivalence Problem of Multitape Finite Automata. *Theoretical Computer Science*, 78(2):347-355, 1991.

[13] H. Johnson. Rational equivalence relations. *Theoretical Computer Science*, 47:39–60, 1986.

[14] H. Läuchli and C. Savioz. Monadic second order definable relations on the binary tree. *Journal of Symbolic Logic*, 52(1):219–226, 1987.

[15] P. Odifreddi. *Classical Recursion Theory*, volume 125. North-Holland, 1989.

[16] M. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop*, pages 125–144, 1959.

[17] H. Rogers. *Theory of recursive functions and effective computability.* McGraw-Hill, 1967.

[18] J. Sakarovitch. *Eléments de théorie des automates.* Vuibert Informatique, 2003.

[19] S. Seibert. Quantifier hierarchies over word relations. *In Proceedings of CSL'91.* Lecture Notes in Computer Sciences, 626:477-481, 1992.

[20] W. Thomas. Infinite trees and automaton-definable relations over the ω-words. *Theoretical Computer Science*, 103:143–159, 1992.

Contents