Regular Separability and Intersection Emptiness Are Independent Problems

Ramanathan S. Thinniyam
Max Planck Institute for Software Systems (MPI-SWS), Germany
thinniyam@mpi-sws.org

Georg Zetzsche
Max Planck Institute for Software Systems (MPI-SWS), Germany
georg@mpi-sws.org

Abstract

The problem of *regular separability* asks, given two languages K and L, whether there exists a regular language S that includes K and is disjoint from L. This problem becomes interesting when the input languages K and L are drawn from language classes beyond the regular languages. For such classes, a mild and useful assumption is that they are full trios, i.e. closed under rational transductions.

All the results on regular separability for full trios obtained so far exhibited a noteworthy correspondence with the intersection emptiness problem: In each case, regular separability is decidable if and only if intersection emptiness is decidable. This raises the question whether for full trios, regular separability can be reduced to intersection emptiness or vice-versa.

We present counterexamples showing that neither of the two problems can be reduced to the other. More specifically, we describe full trios $\mathcal{C}_1$, $\mathcal{D}_1$, $\mathcal{C}_2$, $\mathcal{D}_2$ such that (i) intersection emptiness is decidable for $\mathcal{C}_1$ and $\mathcal{D}_1$, but regular separability is undecidable for $\mathcal{C}_1$ and $\mathcal{D}_1$ and (ii) regular separability is decidable for $\mathcal{C}_2$ and $\mathcal{D}_2$, but intersection emptiness is undecidable for $\mathcal{C}_2$ and $\mathcal{D}_2$.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Formal languages and automata theory

Keywords and phrases Regular separability, intersection emptiness, decidability

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.51

Related Version `https://arxiv.org/pdf/1908.04038.pdf`

Acknowledgements We thank Lorenzo Clemente and Wojciech Czerwiński for fruitful discussions.

1 Introduction

The *intersection emptiness problem* for language classes $\mathcal{C}$ and $\mathcal{D}$ asks for two given languages K from $\mathcal{C}$ and L from $\mathcal{D}$, whether $K \cap L = \emptyset$. If $\mathcal{C}$ and $\mathcal{D}$ are language classes associated with classes of infinite-state systems, then intersection emptiness corresponds to verifying safety properties in concurrent systems where one system of $\mathcal{C}$ communicates with a system of $\mathcal{D}$ via messages or shared memory [6]. The question of separability is to decide whether two given languages are not only disjoint, but whether there exists a finite, easily verifiable, certificate for disjointness (and thus for safety). Specifically, the $\mathcal{S}$ *separability problem* for a fixed class $\mathcal{S}$ of separators and language classes $\mathcal{C}$ and $\mathcal{D}$ asks, for given languages K from $\mathcal{C}$ and L from $\mathcal{D}$, whether there exists a language $S \in \mathcal{S}$ with $K \subseteq S$ and $S \cap L = \emptyset$.

There is extensive literature dealing with the separability problem, with a range of different separators considered. One line of work concerns separability of regular languages by separators from a variety [1] of regular languages. Here, the investigation began with a more general problem, *computing pointlikes* (equivalently, the *covering problem*) [2, 20, 38], but

[1] By which we mean the algebraic notion.

later also concentrated on separability (e.g. [32, 33, 34, 35, 36, 37]). Moreover, separability has been studied for regular tree languages, where separators are either piecewise testable tree languages [21] or languages of deterministic tree-walking automata [5]. For non-regular input languages, separability has been investigated with *piecewise testable languages* (PTL) [11] and generalizations thereof [42] as separators. Separability of subsets of trace monoids [7] and commutative monoids [9] by recognizable subsets has been studied as well.

A natural choice for the separators is the class of *regular languages*. On the one hand, they have relatively high separation power and on the other hand, it is usually verifiable whether a given regular language is in fact a separator. For instance, they generalize piecewise testable languages but are less powerful than context-free languages (CFL). Since the intersection problem for CFL is undecidable, it is not easy to check if a given candidate CFL is a separator.

This has motivated a recent research effort to understand for which language classes $\mathcal{C}, \mathcal{D}$ regular separability is decidable [29, 9, 8]. An early result was that regular separability is undecidable for CFL (by this we mean that both input languages are context-free) [39, 25]. This was recently strengthened to undecidability already for visibly pushdown languages [28] and one-counter languages [29]. On the positive side, it was shown that regular separability is decidable for several subclasses of vector addition systems (VASS): for one-dimensional VASS [29], for commutative VASS languages [9], and for Parikh automata (equivalently, $\mathbb{Z}$-VASS) [8]. Moreover, it is decidable for languages of well-structured transition systems [10]. Furthermore, decidability still holds in many of these cases if one of the inputs is a general VASS language [12]. However, if both inputs are VASS languages, decidability of regular separability remains a challenging open problem.

Of course, if one of the input languages is regular, checking regular separability degenerates into checking intersection with a regular language. Thus, the problem becomes interesting when both input languages are non-regular. Many language classes beyond the regular languages constitute *full trios*, meaning that they are closed under rational transductions. This is typically the case for classes that originate from non-deterministic infinite-state systems [16] and from various types of grammars [16, 13].

In the case of full trios, the available results exhibit a striking correspondence between regular separability and the intersection problem: Wherever decidability of regular separability has been clarified for a full trio, it is decidable if and only if intersection is decidable. Of the abovementioned languages classes, the context-free languages [3], languages of (one-dimensional) VASS [22], one-counter automata [3], Parikh automata [27], and well-structured transition systems [18] each constitute a full trio (visibly pushdown languages and commutative VASS languages do not form full trios). In fact, in the case of well-structured transition systems, it even turned out that two languages are regular-separable if and only if they are disjoint [10]. Moreover, deciding regular separability usually involves non-trivial refinements of the methods for deciding intersection. Without the restriction of being a full trio, there is an example of a language class where the intersection problem is decidable, but regular separability is not: the visibly pushdown languages for a fixed alphabet partition [28].

In light of these observations, there was a growing interest in whether there is a deeper connection between regular separability and intersection emptiness in the case of full trios. In other words: *Is regular separability just intersection emptiness in disguise?* It is conceivable that for full trios, regular separability and intersection emptiness are mutually reducible. An equivalence in this spirit already exists for separability by PTL: For full trios $\mathcal{C}$ and $\mathcal{D}$, separability by PTL for $\mathcal{C}$ and $\mathcal{D}$ is decidable if and only if the simultaneous unboundedness problem is decidable for $\mathcal{C}$ and for $\mathcal{D}$ [11]. These two problems, in turn, are equivalent to computing downward closures [41]. A further analogous equivalence is that full trios are closed under intersection if and only if they are closed under the shuffle operator [19].

Contribution. We show that regular separability and intersection emptiness are independent problems for full trios: Each problem can be decidable while the other is undecidable. Specifically, we present full trios $\mathcal{C}_1$, $\mathcal{D}_1$, $\mathcal{C}_2$, $\mathcal{D}_2$, so that (i) for $\mathcal{C}_1$ and $\mathcal{D}_1$, regular separability is undecidable, but intersection emptiness is decidable and (ii) for $\mathcal{C}_2$ and $\mathcal{D}_2$, regular separability is decidable, but intersection emptiness is undecidable. Some of these classes have been studied before (such as the higher-order pushdown languages), but some have not (to the best of our knowledge). However, they are all natural in the sense that they are defined in terms of machine models and have decidable emptiness and membership problems. We introduce two new classes defined by counter systems that accept based on certain numerical predicates. These predicates are specified either using reset vector addition systems or higher-order pushdown automata.

2 Preliminaries

We use Σ (sometimes Γ) to denote a finite set of letters and Σ^* to denote the set of finite strings (aka words) over the alphabet Σ. To distinguish between expressions over natural numbers and expressions involving words, we use typewriter font to denote letters, e.g. $\mathtt{a}$, $\mathtt{0}$, $\mathtt{1}$, etc. For example, $\mathtt{0}^n$ is the word consisting of an n-fold repetition of the letter $\mathtt{0}$, whereas 0^n is the number zero. The empty string is denoted ε. If $S \subseteq \mathbb{N}$ we write $\mathtt{a}^S$ for the set $\{\mathtt{a}^n \mid n \in S\} \subseteq \mathtt{a}^*$ and 2^S for the set $\{2^n \mid n \in S\} \subseteq \mathbb{N}$.

We define the map $\nu\colon \{\mathtt{0},\mathtt{1}\}^* \to \mathbb{N}$ which takes every word to the number which it denotes in binary representation: We define $\nu(\varepsilon) = 0$ and $\nu(w\mathtt{1}) = 2 \cdot \nu(w) + 1$ and $\nu(w\mathtt{0}) = 2 \cdot \nu(w)$ for $w \in \{\mathtt{0},\mathtt{1}\}^*$. For example, $\nu(\mathtt{110}) = 6$. Often we are only concerned with words of the form $\{\mathtt{0}\} \cup \mathtt{1}\{\mathtt{0},\mathtt{1}\}^*$. For subsets $L \subseteq \{\mathtt{0},\mathtt{1}\}^*$, we define $\nu(L) = \{\nu(w) \mid w \in L\}$.

Languages are denoted by L, L', K etc. and the language of a machine M is denoted by $\mathsf{L}(M)$. Classes of languages are denoted by $\mathcal{C}$, $\mathcal{D}$, etc.

▶ **Definition 2.1.** An *asynchronous transducer* $\mathcal{T}$ is a tuple $\mathcal{T} = (Q, \Gamma, \Sigma, E, q_0, F)$ with a set of finite states Q, finite output alphabet Γ, finite input alphabet Σ, a finite set of edges $E \subseteq Q \times \Gamma^* \times \Sigma^* \times Q$, initial state $q_0 \in Q$ and set of final states $F \subseteq Q$. We write $p \xrightarrow{v|u} q$ if $(p, v, u, q) \in E$ and the machine reads u in state p, outputs v and moves to state q. We also write $p \xrightarrow[*]{w|w'} q$ if there are states $q_0, q_1, \ldots, q_n$ and words $u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_n$ such that $p = q_0$, $q = q_n$, $w' = u_1 u_2 \cdots u_n$, $w = v_1 v_2 \cdots v_n$ and $q_i \xrightarrow{v_i|u_i} q_{i+1}$ for all $0 \leq i \leq n$.

The *transduction* $T \subseteq \Gamma^* \times \Sigma^*$ generated by the transducer $\mathcal{T}$ is the set of tuples $(v, u) \in \Gamma^* \times \Sigma^*$ such that $q_0 \xrightarrow[*]{v|u} q_f$ for some $q_f \in F$. Given a language $L \subseteq \Sigma^*$, we define $TL := \{v \in \Gamma^* \mid \exists u \in L\ (v, u) \in T\}$. A transduction $T \subseteq \Gamma^* \times \Sigma^*$ is *rational* if it is generated by some asynchronous transducer.

A *language* is a subset of Γ^* for some alphabet Γ. A *language class* is a collection of languages, together with some way to finitely represent these languages, for example using machine models or grammars. We call a language class a *full trio* if it is effectively closed under rational transductions. This means, given a representation of L in $\mathcal{C}$ and an asynchronous transducer for $T \subseteq \Gamma^* \times \Sigma^*$, the language TL belongs to $\mathcal{C}$ and one can compute a representation of TL in $\mathcal{C}$.

The following equivalent definition of full trios is well known (see Berstel [3]):

▶ **Lemma 2.2.** *A language class is closed under rational transductions if and only if it is effectively closed under (i) homomorphic image, (ii) inverse homomorphic image, and (iii) intersection with regular languages.*

We are interested in decision problems where the representation of a language L (or possibly multiple languages) is the input. In particular, we study the following problems.

▶ **Problem 2.3** (Intersection Emptiness). For languages classes $\mathcal{C}_1$ and $\mathcal{C}_2$, the *intersection emptiness problem*, briefly $\mathsf{IE}(\mathcal{C}_1, \mathcal{C}_2)$, is defined as follows:
Input: Languages L_1 from $\mathcal{C}_1$ and L_2 from $\mathcal{C}_2$.
Question: Is $L_1 \cap L_2$ empty?

▶ **Problem 2.4** (Regular Separability). For languages classes $\mathcal{C}_1$ and $\mathcal{C}_2$, the *regular separability problem*, briefly $\mathsf{RS}(\mathcal{C}_1, \mathcal{C}_2)$, is defined as follows:
Input: Languages L_1 from $\mathcal{C}_1$ and L_2 from $\mathcal{C}_2$.
Question: Is there a regular language R such that $L_1 \subseteq R$ and $L_2 \cap R = \emptyset$?
We will write $L|K$ to denote that L and K are regular-separable.

▶ **Problem 2.5** (Emptiness). The *emptiness problem* for a language class $\mathcal{C}$, briefly $\mathsf{Empty}(\mathcal{C})$, is defined as:
Input: A language L from $\mathcal{C}$.
Question: Is $L = \emptyset$, i.e. is L empty?

▶ **Problem 2.6** (Infinity). The *infinity problem* for a language class $\mathcal{C}$, briefly $\mathsf{Inf}(\mathcal{C})$, is defined as:
Input: A language L from $\mathcal{C}$.
Question: Does L contain infinitely many words?

3 Incrementing automata

The counterexamples we construct are defined using special kinds of automata that can only increment a counter, which we will define formally below. The acceptance condition requires that the counter value satisfies a specific *numerical predicate*, in addition to reaching a final state. By a *predicate class*, we mean a class $\mathcal{P}$ of predicates over natural numbers (i.e. subsets $P \subseteq \mathbb{N}$) such that there is a way to finitely describe the members of $\mathcal{P}$. As an example, if $\mathcal{C}$ is a language class, then a subset $S \subseteq \mathbb{N}$ is a *pseudo-$\mathcal{C}$ predicate* if $S = \nu(L)$ for some $L \in \mathcal{C}$ and $L \subseteq \{\mathtt{0}, \mathtt{1}\}^*$. Now the class of all pseudo-$\mathcal{C}$ predicates constitutes a predicate class, because a pseudo-$\mathcal{C}$ predicate can be described using the finite description of a language in $\mathcal{C}$. The class of all pseudo-$\mathcal{C}$ predicates is denoted $\mathsf{pseudo}\mathcal{C}$.

▶ **Definition 3.1.** Let $\mathcal{P}$ be a predicate class. An *incrementing automaton over $\mathcal{P}$* is a five-tuple $\mathcal{M} = (Q, \Sigma, E, q_0, F)$ where Q is a finite set of *states*, Σ is its *input alphabet*, $E \subseteq Q \times \Sigma^* \times \{0, 1\} \times Q$ a finite set of *edges*, $q_0 \in Q$ an initial state and F is a finite set of *acceptance pairs* (q, P) where $q \in Q$ is a state and P belongs to $\mathcal{P}$.

A *configuration* of $\mathcal{M}$ is a pair $(q, n) \in Q \times \mathbb{N}$. For two configurations (q, n), (q', n'), we write $(q, n) \xrightarrow{w} (q', n')$ if there are configurations $(q_1, n_1), \ldots, (q_\ell, n_\ell)$ with $q_1 = q$ and $q_\ell = q'$ and edges (q_i, w_i, m_i, q_{i+1}) with $n_{i+1} = n_i + m_i$ for $1 \leq i < \ell$ and $w = w_1 \cdots w_\ell$. The *language accepted by $\mathcal{M}$* is

$$\mathsf{L}(\mathcal{M}) = \{w \in \Sigma^* \mid (q_0, 0) \xrightarrow{w} (q, m) \text{ for some } (q, P) \text{ in } F \text{ with } m \in P\}.$$

The collection of all languages accepted by incrementing automata over $\mathcal{P}$ is denoted $\mathcal{I}(\mathcal{P})$.

It turns out that even with no further assumptions on the predicate class $\mathcal{P}$, the language class $\mathcal{I}(\mathcal{P})$ has some nice closure properties.

▶ **Lemma 3.2.** *Let $\mathcal{P}$ be a predicate class. The languages of incrementing automata over $\mathcal{P}$ are precisely the finite unions of languages of the form $T\mathtt{a}^P$ where $P \in \mathcal{P}$ and $T \subseteq \Sigma^* \times \{\mathtt{a}\}^*$ is a rational transduction. In particular, the class of languages accepted by incrementing automata over $\mathcal{P}$ is a full trio.*

Proof. For every accepting pair (q, P) of $\mathcal{M}$, we construct a transducer $T_{q,P}$, which has the same set of states as $\mathcal{M}$, accepting state set $\{q\}$ and for each edge (q', w, m, q'') of $\mathcal{M}$ the transducer reads $\mathtt{a}$ if $m = 1$ or ε if $m = 0$ and outputs w. Then $\mathsf{L}(\mathcal{M})$ is the finite union of all $T_{q,P}(\mathtt{a}^P)$.

Conversely, since the languages accepted by incrementing automata over $\mathcal{P}$ are clearly closed under union, it suffices to show that $T\mathtt{a}^P$ is accepted by an incrementing automaton over $\mathcal{P}$. We may assume that T is given by a transducer in which every edge is of the form $(q, w, \mathtt{a}^m, q')$ with $m \in \{0, 1\}$. Let $\mathcal{M}$ have the same state set as T and turn every edge $(q, w, \mathtt{a}^m, q')$ into an edge (q, w, m, q') for $\mathcal{M}$. Finally, for every final state q of T, we give $\mathcal{M}$ an accepting pair (q, P). Then clearly $\mathsf{L}(\mathcal{M}) = T\mathtt{a}^P$.

This implies that the class of incrementing automata over $\mathcal{P}$ is a full trio: If $L \subseteq \Sigma^*$ is accepted by a incrementing automata over $\mathcal{P}$, then we can write $L = T_1\mathtt{a}^{P_1} \cup \cdots \cup T_\ell\mathtt{a}^{P_\ell}$ with $T_1, \ldots, T_\ell \subseteq \Sigma^* \times \mathtt{a}^*$. If $T \subseteq \Gamma^* \times \Sigma^*$ is a rational transduction, then $TL = (TT_1)\mathtt{a}^{P_1} \cup \cdots \cup (TT_\ell)\mathtt{a}^{P_\ell}$ and since TT_i is again a rational transduction for $1 \leq i \leq \ell$, the lanuage TL is accepted by some incrementing automaton over $\mathcal{P}$. ◀

It is obvious that the class $\mathcal{I}(\mathcal{P})$ does not always have a decidable emptiness problem: Emptiness is decidable for $\mathcal{I}(\mathcal{P})$ if and only if it is decidable whether a given predicate from $\mathcal{P}$ intersects a given arithmetic progression, i.e. given P and $m, n \in \mathbb{N}$, whether $(m+n\mathbb{N}) \cap P \neq \emptyset$. For all the predicate classes $\mathcal{P}$ we consider, emptiness for $\mathcal{I}(\mathcal{P})$ will always be decidable.

4 Decidable Intersection and Undecidable Regular Separability

In this section, we present a language class $\mathcal{C}$ so that the intersection emptiness problem $\mathsf{IE}(\mathcal{C}, \mathcal{C})$ is decidable for $\mathcal{C}$, but the regular separability problem $\mathsf{RS}(\mathcal{C}, \mathcal{C})$ is undecidable for $\mathcal{C}$. The definition of $\mathcal{C}$ is based on reset vector addition systems.

Reset Vector Addition Systems. A *reset vector addition system* (reset VASS) is a tuple $\mathcal{V} = (Q, \Sigma, n, E, q_0, F)$, where Q is a finite set of *states*, Σ is its *finite input alphabet*, $n \in \mathbb{N}$ is its number of counters, $E \subseteq Q \times \Sigma^* \times \{1, \ldots, n\} \times \{0, 1, -1, \mathsf{r}\} \times Q$ is a finite set of *edges*, $q_0 \in Q$ is its *initial state*, and $F \subseteq Q$ is its set of *final states*. A *configuration* of $\mathcal{V}$ is a tuple $(q, m_1, \ldots, m_n)$ where $q \in Q$ and $m_1, \ldots, m_n \in \mathbb{N}$. We write $(q, m_1, \ldots, m_n) \xrightarrow{w} (q', m_1', \ldots, m_n')$ if there is an edge (q, w, k, x, q') such that for every $j \neq k$, we have $m_j' = m_j$ and

- if $x \in \{-1, 0, 1\}$, then $m_k' = m_k + x$,
- if $x = \mathsf{r}$, then $m_k' = 0$.

If there are configurations $c_1, \ldots, c_\ell$ and words $w_1, \ldots, w_{\ell-1}$ with $c_i \xrightarrow{w_i} c_{i+1}$ for $1 \leq i < \ell$, and $w = w_1 \cdots w_\ell$, then we also write $c_1 \xrightarrow{w} c_\ell$. The *language accepted by* $\mathcal{V}$ is defined as

$$\mathsf{L}(\mathcal{V}) = \{w \in \Sigma^* \mid (q_0, 0, \ldots, 0) \xrightarrow{w} (q, m_1, \ldots, m_n) \text{ for some } q \in F \text{ and } m_1, \ldots, m_n \in \mathbb{N}\}.$$

The class of languages accepted by reset VASS is denoted $\mathcal{R}$.

Our language class will be $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$, i.e. incrementing automata with access to predicates of the form $\nu(L)$ where $L \subseteq \{\mathtt{0}, \mathtt{1}\}^*$ is the language of a reset VASS.

▶ **Theorem 4.1.** $\mathsf{RS}(\mathcal{I}(\mathsf{pseudo}\mathcal{R}),\mathcal{I}(\mathsf{pseudo}\mathcal{R}))$ *is undecidable and* $\mathsf{IE}(\mathcal{I}(\mathsf{pseudo}\mathcal{R}),\mathcal{I}(\mathsf{pseudo}\mathcal{R}))$ *is decidable.*

Note that $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$ is a full trio (Lemma 3.2) and since intersection is decidable, in particular its emptiness problem is decidable: For $L \subseteq \Sigma^*$, one has $L \cap \Sigma^* = \emptyset$ if and only if $L = \emptyset$. Moreover, note that we could not have chosen $\mathcal{R}$ as our example class: Since reset VASS are well-structured transition systems, regular separability is decidable for them [10].

Before we begin with the proof of Theorem 4.1, let us mention that instead of $\mathcal{R}$, we could have chosen any language class $\mathcal{D}$, for which (i) $\mathcal{D}$ is closed under rational transductions, (ii) $\mathcal{D}$ is closed under intersection, (iii) $\mathsf{Empty}(\mathcal{D})$ is decidable and (iv) $\mathsf{Inf}(\mathcal{D})$ is undecidable. For example, we could have also used lossy channel systems instead of reset VASS.

We now recall some results regarding $\mathcal{R}$ from literature.

▶ **Lemma 4.2.** *Emptiness is decidable for $\mathcal{R}$.*

The lemma follows from the fact that reset VASS are well-structured transition systems [14], for which the coverability problem is decidable [1, 17] and the fact that a reset VASS has a non-empty language if and only if a particular configuration is coverable.

The following can be shown using standard product constructions, please see the full version [40].

▶ **Lemma 4.3.** *$\mathcal{R}$ is closed under rational transductions, union, and intersection.*

We now show that regular separability is undecidable for $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$. We do this using a reduction from the infinity problem for $\mathcal{R}$, whose undecidability is an easy consequence of the undecidability of boundedness of reset VASS.

The boundedness problem for reset VASS is defined below and was shown to be undecidable by Dufourd, Finkel, and Schnoebelen [14] (and a simple and more general proof was given by Mayr [30]). A configuration $(q, x_1, \ldots, x_n)$ is *reachable* if there is a $w \in \Sigma^*$ with $(q_0, 0, \ldots, 0) \xrightarrow{w} (q, x_1, \ldots, x_n)$. A reset VASS $\mathcal{V}$ is called *bounded* if there is a $B \in \mathbb{N}$ such that for every reachable $(q, x_1, \ldots, x_n)$, we have $x_1 + \cdots + x_n \leq B$. Hence, the *boundedness problem* is the following.
Input: A reset VASS $\mathcal{V}$.
Question: Is $\mathcal{V}$ bounded?

▶ **Lemma 4.4.** *The infinity problem for $\mathcal{R}$ is undecidable.*

Proof. From an input reset VASS $\mathcal{V} = (Q, \Sigma, n, E, q_0, F)$, we construct a reset VASS $\mathcal{V}'$ over the alphabet $\Sigma' = \{\mathtt{a}\}$ as follows. In every edge of $\mathcal{V}$, we replace the input word by the empty word ε. Moreover, we add a fresh state s, which is the only final state of $\mathcal{V}'$. Then, we add an edge $(q, \varepsilon, 1, 0, s)$ for every state q of $\mathcal{V}$. Finally, we add a loop $(s, \mathtt{a}, i, -1, s)$ for every $i \in \{1, \ldots, n\}$. This means $\mathcal{V}'$ simulates a computation of $\mathcal{V}$ (but disregarding the input) and can spontaneously jump into the state s, from where it can decrement counters. Each time it decrements a counter in s, it reads an $\mathtt{a}$ from the input. Thus, clearly, $\mathsf{L}(\mathcal{V}') \subseteq \mathtt{a}^*$. Moreover, we have $\mathtt{a}^m \in \mathsf{L}(\mathcal{V}')$ if and only if there is a reachable configuration $(q, x_1, \ldots, x_n)$ of $\mathcal{V}$ with $x_1 + \cdots + x_n \geq m$. Thus, $\mathsf{L}(\mathcal{V}')$ is finite if and only if $\mathcal{V}$ is bounded. ◀

Note that infinity is already undecidable for languages that are subsets of $\mathtt{1}\mathtt{0}^*$. This is because given L from $\mathcal{R}$, a rational transduction yields $L' = \{\mathtt{1}\mathtt{0}^{|w|} \mid w \in L\}$ and L' is infinite if and only if L is.

Our reduction from the infinity problem works because the input languages have a particular shape, for which regular separability has a simple characterization.

▶ **Lemma 4.5.** *Let $S_0, S_1 \subseteq \mathbb{N}$ and $\mathbb{N} \setminus 2^{\mathbb{N}} \subseteq S_1$. Then $\mathtt{a}^{S_0}$ and $\mathtt{a}^{S_1}$ are regular-separable if and only if S_0 is finite and disjoint from S_1.*

Proof. If S_0 is finite and disjoint from S_1, then clearly $\mathtt{a}^{S_0}$ is a regular separator. For the "only if" direction, consider any infinite regular language $R \subseteq \mathtt{a}^*$. It has to include an arithmetic progession, meaning that there exist $m, n \in \mathbb{N}$ with $\mathtt{a}^{m+n\mathbb{N}} \subseteq R$. Hence, for sufficiently large ℓ, the language $\{\mathtt{a}^x \mid 2^\ell < x < 2^{\ell+1}\} \subseteq S_1$ must intersect with R. In other words, no infinite R can be a regular separator of $\mathtt{a}^{S_0}$ and $\mathtt{a}^{S_1}$ i.e. S_0 must be finite (and disjoint from S_1). ◀

▶ **Lemma 4.6.** *Regular separability is undecidable for $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$.*

Proof. We reduce the infinity problem for $\mathcal{R}$ (which is undecidable by Lemma 4.4) to regular separability in $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$. Suppose we are given L from $\mathcal{R}$. Since $\mathcal{R}$ is effectively closed under rational transductions, we also have $K = \{\mathtt{1}\mathtt{0}^{|w|} \mid w \in L\}$ in $\mathcal{R}$. Note that K is infinite if and only if L is infinite. Then $\nu(K) \subseteq 2^{\mathbb{N}}$ and $K_1 := \mathtt{a}^{\nu(K)}$ belongs to $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$. Let $K_2 = \mathtt{a}^{\mathbb{N}\setminus 2^{\mathbb{N}}} = \mathtt{a}^{\nu(\mathtt{1}\{\mathtt{0},\mathtt{1}\}^*\mathtt{1}\{\mathtt{0},\mathtt{1}\}^*)}$, which also belongs to $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$, because $\mathtt{1}\{\mathtt{0},\mathtt{1}\}^*\mathtt{1}\{\mathtt{0},\mathtt{1}\}^*$ is regular and thus a member of $\mathcal{R}$.

By Lemma 4.4, K_1 and K_2 are regular-separable if and only if K_1 is finite and disjoint from K_2. Since $K_1 \cap K_2 = \emptyset$ by construction, we have regular separability if and only if K_1 is finite, which happens if and only if K is finite. ◀

For Theorem 4.1, it remains to show that intersection is decidable for $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$. We do this by expressing intersection non-emptiness in the logic $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$, which is the positive Σ_1 fragment of Presburger arithmetic extended with pseudo-$\mathcal{R}$ predicates. Moreover, we show that this logic has a decidable truth problem.

We begin with some notions from first-order logic (please see [15] for syntax and semantics of first-order logic). First-order formulae will be denoted by $\phi(\bar{x}), \psi(y)$ etc. where $\bar{x}$ is a tuple of (possibly superset of the) free variables and y is a single free variable. For a formula $\phi(\bar{x})$, we denote by $[\![\phi(\bar{x})]\!]$ the set of its solutions (in our case, the domain is $\mathbb{N}$).

Our decision procedure for $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ is essentially the same as the procedure to decide the first-order theory of automatic structures [4], except that instead of regular languages, we use $\mathcal{R}$. For $\bar{w} = (w_1, w_2, \ldots, w_k) \in (\Sigma^*)^k$, the *convolution* $w_1 \otimes w_2 \otimes \ldots \otimes w_k$ is a word over the alphabet $(\Sigma \cup \{\Box\})^k$ where $\Box$ is a padding symbol not present in Σ. If $w_i = w_{i1}w_{i2}\ldots w_{im_i}$ and $m = \max\{m_1, m_2, \ldots, m_k\}$ then

$$w_1 \otimes w_2 \otimes \ldots \otimes w_k := \begin{bmatrix} w'_{11} \\ w'_{21} \\ \vdots \\ w'_{k1} \end{bmatrix} \ldots \begin{bmatrix} w'_{1m} \\ w'_{2m} \\ \vdots \\ w'_{km} \end{bmatrix} \in ((\Sigma \cup \{\Box\})^k)^*$$

where $w'_{i1} \cdots w'_{im} = \Box^{m-m_i} w_i$ for $1 \leq i \leq k$. We say that a k-ary (arithmetic) relation $R \subseteq \mathbb{N}^k$ is a *pseudo-$\mathcal{R}$ relation* if the set of words $L_R = \{w_1 \otimes w_2 \otimes \cdots \otimes w_k \mid (\nu(w_1), \ldots, \nu(w_k)) \in R\}$ belongs to $\mathcal{R}$. In our decision procedure for $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$, we will show inductively that every formula defines a pseudo-$\mathcal{R}$ relation.

▶ Remark 4.7. Note that our definition of the convolution deviates from the usual one that pads words on the right [4, 26]. This is because we want pseudo-$\mathcal{R}$ predicates to be pseudo-$\mathcal{R}$ relations. By our definition of ν, this means the least significant bit will always be on the right. Since we also want the ternay addition relation $\{(x, y, z) \in \mathbb{N}^3 \mid x + y = z\}$ to be a pseudo-$\mathcal{R}$ relation, we need to align the words in the convolution at the least significant bit and thus pad on the left.

Formally, we consider the theory $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ where $(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ is the structure with domain $\mathbb{N}$ of natural numbers, the constant symbol 1 and the binary symbols $+$ and $\leq$ taking their canonical interpretations and $\mathsf{pseudo}\mathcal{R}$ is a set of predicate symbols, one for each pseudo-$\mathcal{R}$ predicate. By Σ_1^+ we mean the fragment of first order formulae obtained by using only the boolean operations $\wedge, \vee$ and existential quantification.

▶ **Definition 4.8.** Let $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ be the set of first order logic formulae given by the following grammar:

$$\phi(\bar{x}, \bar{y}, \bar{z}) := S(x) \mid t_1 \leq t_2 \mid \phi_1(\bar{x}, \bar{y}) \wedge \phi_2(\bar{x}, \bar{z}) \mid \phi_1(\bar{x}, \bar{y}) \vee \phi_2(\bar{x}, \bar{z}) \mid \exists y\ \phi'(y, \bar{x})$$

where S is from $\mathsf{pseudo}\mathcal{R}$ and t_1, t_2 are terms obtained from using variables, 1 and $+$.

▶ **Lemma 4.9.** *The truth problem for* $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ *is decidable.*

Proof. It is clear that by introducing new existentially quantified variables, one can transform each formula from $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ into an equivalent formula that is generated by the simpler grammar

$$\begin{aligned}\phi(\bar{x}, \bar{y}, \bar{z}) := &S(x) \mid x + y = z \mid x = 1 \mid \\ &\phi_1(\bar{x}, \bar{y}) \wedge \phi_2(\bar{x}, \bar{z}) \mid \phi_1(\bar{x}, \bar{y}) \vee \phi_2(\bar{x}, \bar{z}) \mid \exists y\ \phi'(y, \bar{x})\end{aligned}$$

We want to show that given any input sentence ψ from $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$, we can decide if it is true or not. If the sentence has no variables, then it is trivial to decide. Otherwise, $\psi = \exists \bar{x}\ \phi(\bar{x})$ for some formula $\phi(\bar{x})$. We claim that the solution set $R = [\![\phi(\bar{x})]\!]$ is a pseudo-$\mathcal{R}$ relation and a reset VASS for L_R can be effectively computed. Assuming the claim, the truth of ψ reduces to the emptiness of $[\![\phi(\bar{x})]\!]$ or equivalently the emptiness of L_R, which is decidable by Lemma 4.2.

We prove the claim by structural induction on the defining formula $\phi(\bar{x})$, please see the full version [40] for details. ◀

▶ Remark 4.10. The truth problem for $\Pi_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ is undecidable by reduction from the infinity problem for $\mathcal{R}$. Given $L \subseteq \mathtt{10}^*$, let $R_L = \nu(L) \subseteq \mathbb{N}$ be the predicate corresponding to L. Now the downward closure $D := \{x \in \mathbb{N} \mid \exists y \colon x \leq y \wedge R_L(y)\}$ is definable in $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ and therefore $K := \nu(D)$ belongs to $\mathcal{R}$ by the proof of Lemma 4.9. Then the Π_1^+-sentence $\forall x \colon R_K(x)$ is true if and only if L is infinite.

Having established that $\Sigma_1^+(\mathbb{N}, +, \leq, 1, \mathsf{pseudo}\mathcal{R})$ is decidable, we are ready to show that intersection emptiness is decidable for $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$.

▶ **Lemma 4.11.** *The intersection problem is decidable for* $\mathcal{I}(\mathsf{pseudo}\mathcal{R})$.

Proof. Given $L_1, L_2 \in \mathcal{I}(\mathsf{pseudo}\mathcal{R})$, by Lemma 3.2, we know that both L_1 and L_2 are finite unions of languages of the form $T\mathtt{a}^S$, where S is a pseudo-$\mathcal{R}$ predicate. Therefore, it suffices to decide the emptiness of intersections of the form $T_1\mathtt{a}^{S_1} \cap T_2\mathtt{a}^{S_2}$ where S_1 and S_2 are pseudo-$\mathcal{R}$ predicates. Note that $T_1\mathtt{a}^{S_1} \cap T_2\mathtt{a}^{S_2} = \emptyset$ iff $T_2^{-1}T_1\mathtt{a}^{S_1} \cap \mathtt{a}^{S_2} = \emptyset$. Since $T_2^{-1}T_1$ is again a rational transduction, it suffices to check emptiness of languages of the form $T\mathtt{a}^{S_1} \cap \mathtt{a}^{S_2}$ where $T \subseteq \mathtt{a}^* \times \mathtt{a}^*$ is a rational transduction. Notice that we can construct an automaton $\mathcal{A}$ over the alphabet $\Sigma' = \{\mathtt{b}, \mathtt{c}\}$ with the same states as the transducer $\mathcal{M}_T$ for T and where for any transition $p \xrightarrow{\mathtt{a}^m \mid \mathtt{a}^n} q$ of $\mathcal{M}_T$ we have a transition $p \xrightarrow{\mathtt{b}^m \mathtt{c}^n} q$ in $\mathcal{A}$. It is clear that $(\mathtt{a}^x, \mathtt{a}^y) \in T$ iff there exists a word $w \in \mathsf{L}(\mathcal{A})$ such that w contains exactly x

occurrences of $\mathtt{b}$ and y occurrences of $\mathtt{c}$. Now it follows from Parikh's theorem [31] that the set $\{(x,y)\in\mathbb{N}\times\mathbb{N}\mid(\mathtt{a}^x,\mathtt{a}^y)\in T\}$ is semilinear, meaning that there are numbers $n_0,\ldots,n_k$ and $m_0,\ldots,m_k$ such that $(\mathtt{a}^x,\mathtt{a}^y)\in T$ if and only if

$$\exists z_1\exists z_2\ldots\exists z_k\ (x=n_0+\sum_{i=i}^{k}z_in_i)\ \wedge\ (y=m_0+\sum_{i=i}^{k}z_im_i).$$

In particular, there is a formula $\phi_T(x,y)$ in $\Sigma_1^+(\mathbb{N},+,\le,1,\mathsf{pseudo}\mathcal{R})$ such that $(\mathtt{a}^x,\mathtt{a}^y)\in T$ if and only if $\phi_T(x,y)$ is satisfied. We can now write a formula $\phi_2(x)$ in $\Sigma_1^+(\mathbb{N},+,\le,1,\mathsf{pseudo}\mathcal{R})$ such that $\phi_2(x)$ is satisfied if and only if $\mathtt{a}^x\in T\mathtt{a}^{S_2}$:

$$\phi_2(x):=\exists y\ \big(\phi_T(x,y)\ \wedge\ S_2(y)\big)$$

In the same way, the formula $\phi_1(x):=S_1(x)$ defines $\mathtt{a}^{S_1}$. Now set $\phi=\exists x\ \big(\phi_1(x)\ \wedge\ \phi_2(x)\big)$. Then ϕ is true if and only if $T\mathtt{a}^{S_2}\cap\mathtt{a}^{S_1}\neq\emptyset$. Decidability of $\mathsf{IE}(\mathcal{I}(\mathsf{pseudo}\mathcal{R}),\mathcal{I}(\mathsf{pseudo}\mathcal{R}))$ follows from Lemma 4.9. ◀

5 Decidable Regular Separability and Undecidable Intersection

In this section, we present language classes $\mathcal{C}$ and $\mathcal{D}$ so that $\mathsf{IE}(\mathcal{C},\mathcal{D})$ is undecidable, but $\mathsf{RS}(\mathcal{C},\mathcal{D})$ is decidable. These classes are constructed using higher-order pushdown automata, which we define first.

We follow the definition of [23]. Higher-order pushdown automata are a generalization of pushdown automata where instead of manipulating a stack, one can manipulate a stack of stacks (order-2), a stack of stacks of stacks (order-3), etc. Therefore, we begin by defining these higher-order stacks. While for ordinary (i.e. order-1) pushdown automata, stacks are words over the stack alphabet Γ, order-$(k+1)$ stacks are sequences of order-k stacks. Let Γ be an alphabet and $k\in\mathbb{N}$. The set of *order-k stacks* $\mathcal{S}_k^\Gamma$ is inductively defined as follows:

$$\mathcal{S}_0^\Gamma=\Gamma,\qquad \mathcal{S}_{k+1}^\Gamma=\{[s_1\cdots s_m]_{k+1}\mid m\ge1,\ s_1,\ldots,s_m\in\mathcal{S}_k^\Gamma\}.$$

For a word $v\in\Gamma^+$, the stack $[\cdots[[v]_1]_2\cdots]_k$ is also denoted $[\![v]\!]_k$. The function top yields the topmost symbol from Γ. This means, we have $\mathsf{top}([s_1\cdots s_m]_1)=s_m$ and $\mathsf{top}([s_1\cdots s_m]_k)=\mathsf{top}(s_m)$ for $k>1$.

Higher-order pushdown automata operate on higher-order stacks by way of instructions. For the stack alphabet Γ and for order-k stacks, we have the instruction set $I_k^\Gamma=\{\mathsf{push}_i,\mathsf{pop}_i\mid 1\le i\le k\}\cup\{\mathsf{rew}_\gamma\mid\gamma\in\Gamma\}$. These instructions act on $\mathcal{S}_k^\Gamma$ as follows:

$$\begin{aligned}
[s_1\cdots s_m]_1\cdot\mathsf{rew}_\gamma &= [s_1\cdots s_{m-1}\gamma]_1 & \\
[s_1\cdots s_m]_k\cdot\mathsf{rew}_\gamma &= [s_1\cdots s_{m-1}(s_m\cdot\mathsf{rew}_\gamma)]_k & \text{if } k>1\\
[s_1\cdots s_m]_i\cdot\mathsf{push}_i &= [s_1\cdots s_m s_m]_i & \\
[s_1\cdots s_m]_k\cdot\mathsf{push}_i &= [s_1\cdots s_m\ (s_m\cdot\mathsf{push}_i)]_k & \text{if } k>i\\
[s_1\cdots s_m]_i\cdot\mathsf{pop}_i &= [s_1\cdots s_{m-1}]_i & \text{if } m\ge2\\
[s_1\cdots s_m]_k\cdot\mathsf{pop}_i &= [s_1\cdots s_{m-1}\ (s_m\cdot\mathsf{pop}_i)]_k & \text{if } k>i
\end{aligned}$$

and in all other cases, the result is undefined. For a word $w\in(I_k^\Gamma)^*$ and a stack $s\in\mathcal{S}_k^\Gamma$, the stack $s\cdot w$ is defined inductively by $s\cdot\varepsilon=s$ and $s\cdot(wx)=(s\cdot w)\cdot x$ for $x\in I_k^\Gamma$.

An *(order-k) higher-order pushdown automaton* (short HOPA) is a tuple $\mathcal{A}=(Q,\Sigma,\Gamma,\bot,E,q_0,F)$, where Q is a finite set of *states*, Σ is its *input alphabet*, Γ is its *stack alphabet*, $\bot\in\Gamma$ is its *stack bottom symbol*, E is a finite subset of $Q\times\Sigma^*\times\Gamma\times(I_k^\Gamma)^*\times Q$

whose elements are called *edges*, $q_0 \in Q$ is its *initial state*, and $F \subseteq Q$ is its set of *final states*. A *configuration* is a pair $(q, s) \in Q \times \mathcal{S}_k^\Gamma$. When drawing a higher-order pushdown automaton, an edge (q, u, γ, v, q') is represented by an arc $q \xrightarrow{u|\gamma|v} q'$. An arc $q \xrightarrow{u|v} q'$ means that for each $\gamma \in \Gamma$, there is an edge (q, u, γ, v, q').

For configurations $(q, s), (q', s')$ and a word $u \in \Sigma^*$, we write $(q, s) \xrightarrow{u}_{\mathcal{A}} (q', s')$ if there are edges $(q_1, u_1, \gamma_1, v_1, q_2), (q_2, u_2, \gamma_2, v_2, q_3), \ldots, (q_{n-1}, u_{n-1}, \gamma_{n-1}, v_{n-1}, q_n)$ in E and stacks $s_1, \ldots, s_n \in \mathcal{S}_k^\Gamma$ with $\mathsf{top}(s_i) = \gamma_i$ and $s_i \cdot v_i = s_{i+1}$ for $1 \le i \le n-1$ such that $(q, s) = (q_1, s_1)$ and $(q', s') = (q_n, s_n)$ and $u = u_1 \cdots u_n$. The *language accepted by* $\mathcal{A}$ is defined as

$$\mathsf{L}(\mathcal{A}) = \{w \in \Sigma^* \mid (q_0, [\![\bot]\!]_k) \xrightarrow{w}_{\mathcal{A}} (q, s) \text{ for some } q \in F \text{ and } s \in \mathcal{S}_k^\Gamma\}.$$

The languages accepted by order-k pushdown automata are called *order-k pushdown languages*. By $\mathcal{H}$, we denote the class of languages accepted by an order-k pushdown automaton for some $k \in \mathbb{N}$. In our example of classes with decidable regular separability and undecidable intersection, one of the two classes is $\mathcal{H}$. The other class will again be defined using incrementing automata.

▶ **Definition 5.1.** Let $\mathcal{C}$ be a language class. A predicate $P \subseteq \mathbb{N}$ is a *power-$\mathcal{C}$ predicate* if $P = \mathbb{N} \setminus 2^{\mathbb{N}} \cup \{2^{\nu(w)} \mid w \in L\}$ for some language L from $\mathcal{C}$. The class of power-$\mathcal{C}$ predicates is denoted $\mathsf{power}\mathcal{C}$.

Our example of classes with decidable regular separability but undecidable intersection is $\mathcal{H}$ on the one hand and $\mathcal{I}(\mathsf{power}\mathcal{H})$ on the other hand. It is well-known that $\mathcal{H}$ is a full trio (see, e.g., [16]). Moreover, $\mathcal{I}(\mathsf{power}\mathcal{H})$ is a full trio according to Lemma 3.2.

▶ **Theorem 5.2.** *$\mathsf{RS}(\mathcal{H}, \mathcal{I}(\mathsf{power}\mathcal{H}))$ is decidable, whereas $\mathsf{IE}(\mathcal{H}, \mathcal{I}(\mathsf{power}\mathcal{H}))$ is undecidable.*

Note that decidable regular separability implies that $\mathcal{I}(\mathsf{power}\mathcal{H})$ has a decidable emptiness problem: For $L \subseteq \Sigma^*$, one has $\Sigma^* | L$ if and only if $L = \emptyset$. Moreover, note that we could not have chosen $\mathcal{H}$ as our counterexample, because regular separability is undecidable for $\mathcal{H}$ (already for context-free languages) [39, 25].

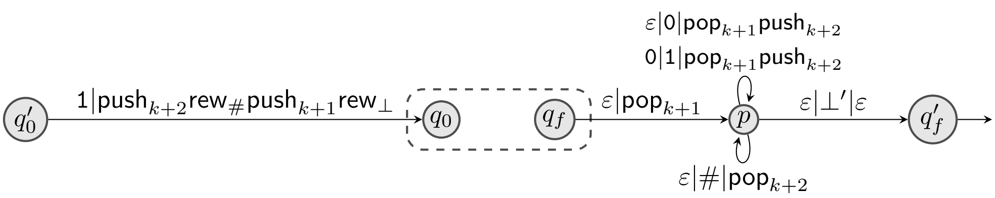
For showing Theorem 5.2, we rely on two ingredients. The first is that infinity is decidable for higher-order pushdown languages. This is a direct consequence of a result of Hague, Kochems and Ong [23], showing that the more general simultaneous unboundedness problem [41] and diagonal problem [11] are decidable for higher-order pushdown automata.

▶ **Lemma 5.3** ([23]). *$\mathsf{Inf}(\mathcal{H})$ is decidable.*

The other ingredient is that turning binary representations into unary ones can be achieved in higher-order pushdown automata.

▶ **Lemma 5.4.** *If $L \subseteq \{0, 1\}^*$ is an order-k pushdown language, then $L' = \{10^{\nu(w)} \mid w \in L\}$ is an order-$(k + 2)$ pushdown language.*

Proof. Let $\mathcal{A}$ be an order-k HOPA accepting $L \subseteq \{0, 1\}^*$. We construct an order-$(k + 2)$ HOPA $\mathcal{A}'$ for L'. We may clearly assume that $\mathcal{A}$ has only one final state q_f. The following diagram describes $\mathcal{A}'$:



The HOPA $\mathcal{A}'$ starts in the configuration $(q_0', [\![\bot']\!]_{k+2})$ and in moving to q_0, it reads $\mathtt{1}$ and goes to $(q_0, [[\![\bot']\!]_{k+1}[[\![\#]\!]_k[\![\bot]\!]_k]_{k+1}]_{k+2})$. In the part in the dashed rectangle, $\mathcal{A}'$ simulates $\mathcal{A}$. However, instead of reading an input symbol $a \in \{\mathtt{0},\mathtt{1}\}$, $\mathcal{A}$ stores that symbol on the stack. In order not to interfere with the simulation of $\mathcal{A}$, this is done by copying the order-k stack used by $\mathcal{A}$ and storing a in the copy below. This is achieved as follows. For every edge $p \xrightarrow{a|\gamma|v} q$ with $v \in (I_k^\Gamma)^*$, $\mathcal{A}'$ instead has an edge

$$p \xrightarrow{\varepsilon|\gamma|\mathsf{push}_1\mathsf{rew}_a\mathsf{push}_{k+1}\mathsf{pop}_1 v} q$$

This pushes the input symbol a on the (topmost order-k) stack, makes a copy of the topmost order-k stack, removes the a from this fresh copy, and then excutes v. Edges $p \xrightarrow{\varepsilon|\gamma|v}$ (i.e. ones that read ε from the input) are kept.

When $\mathcal{A}'$ arrives in q_f, it has a stack $[[\![\bot']\!]_{k+1}[[\![\#]\!]_k s_1 \cdots s_m s]_{k+1}]_{k+2}$, where s is the order-k stack reached in the computation of $\mathcal{A}$, and $s_1, \ldots, s_m$ store the input word $w \in \Sigma^*$ read by $\mathcal{A}$, meaning $\mathsf{top}(s_1)\cdots\mathsf{top}(s_m) = w$. When moving to p, $\mathcal{A}'$ removes s so as to obtain $[[\![\bot']\!]_{k+1}[[\![\#]\!]_k s_1 \cdots s_m]_{k+1}]_{k+2}$ as a stack.

In p, $\mathcal{A}'$ reads the input word $\mathtt{0}^{\nu(w)}$ as follows. While in p, the stack always has the form

$$t = [[\![\bot']\!]_{k+1} t_1 \cdots t_\ell]_{k+2}, \tag{1}$$

where each t_i is an order-$(k+1)$ stack of the form $[[\![\#]\!]_k s_1 \cdots s_m]_{k+1}$ for some order-k stacks $s_1, \ldots, s_m \in \mathcal{S}_k^\Gamma$. To formulate an invariant that holds in state p, we define a function μ on the stacks as in (1). First, if $t_i = [[\![\#]\!]_k s_1 \cdots s_m]_{k+1}$, then let $\mu(t_i) = \nu(\mathsf{top}(s_1)\cdots\mathsf{top}(s_m))$. Next, let $\mu(t) = \mu(t_1) + \cdots + \mu(t_\ell)$. It is not hard to see that the loops on p preserve the following invariant: If $\mathtt{0}^r$ is the input word read from configuration (p,t) to (p,t'), then $\mu(t) = r + \mu(t')$. To see this, consider a one step transition $(p,t) \xrightarrow{\varepsilon|\mathtt{0}|\mathsf{pop}_{k+1}\mathsf{push}_{k+2}} (p,t')$. If

$$t = [[\![\bot']\!]_{k+1} t_1 \cdots t_\ell]_{k+2} = [[\![\bot']\!]_{k+1} t_1 \cdots t_{\ell-1}[[\![\#]\!]_k s_1 \cdots s_m]_{k+1}]_{k+2}$$

then

$$t' = [[\![\bot']\!]_{k+1} t_1 \cdots t_{\ell-1}[[\![\#]\!]_k s_1 \cdots s_{m-1}]_{k+1}[[\![\#]\!]_k s_1 \cdots s_{m-1}]_{k+1}]_{k+2}.$$

If $w = \mathsf{top}(s_1)\ldots\mathsf{top}(s_m)$ then $w = w'\mathtt{0}$ where $w' = \mathsf{top}(s_1)\cdots\mathsf{top}(s_{m-1})$ since we popped s_m off the stack. Moreover,

$$\mu(t') = \sum_{i=1}^{\ell-1}\mu(t_i) + 2\nu(w') = \sum_{i=1}^{\ell-1}\mu(t_i) + \nu(w) = \mu(t)$$

Similarly we see that if the transition taken is $\mathtt{0}|\mathtt{1}|\mathsf{pop}_{k+1}\mathsf{push}_{k+2}$ then we get $\mu(t) = \mu(t')+1$. By induction on the length of the run, we get $\mu(t) = r + \mu(t')$ when $\mathtt{0}^r$ is read.

Now observe that when $\mathcal{A}'$ first arrives in p with stack t, then by construction we have $\ell = 1$ and $\mu(t) = \mu(t_1) = \nu(w)$. Moreover, when $\mathcal{A}'$ moves on to q_f' with a stack as in (1), then $\ell = 0$ and thus $\mu(t) = 0$. Thus, the invariant implies that if $\mathcal{A}'$ reads $\mathtt{0}^r$ while in p, then $r = \nu(w)$. This means, $\mathcal{A}'$ has read $\mathtt{1}\mathtt{0}^{\nu(w)}$ in total.

Finally, from a stack t as in (1), $\mathcal{A}'$ reaches q_f' in finitely many steps, please see the full version [40] for details. ◀

▶ **Lemma 5.5.** *The problem* $\mathsf{IE}(\mathcal{H}, \mathcal{I}(\mathsf{power}\mathcal{H}))$ *is undecidable.*

Proof. We reduce intersection emptiness for context-free languages, which is well-known to be undecidable [24], to $\mathsf{IE}(\mathcal{H}, \mathcal{I}(\mathsf{power}\mathcal{H}))$. Let $K_1, K_2 \subseteq \{\mathtt{0},\mathtt{1}\}^*$ be context-free. Since $K_1 \cap K_2 \neq \emptyset$ if and only if $\mathtt{1}K_1 \cap \mathtt{1}K_2 \neq \emptyset$ and $\mathtt{1}K_i$ is context-free for $i = 0, 1$, we may assume that $K_1, K_2 \subseteq \mathtt{1}\{\mathtt{0},\mathtt{1}\}^*$. This implies $K_1 \cap K_2 \neq \emptyset$ if and only if $\nu(K_1) \cap \nu(K_2) \neq \emptyset$.

Let $P_2 = \mathbb{N} \setminus 2^{\mathbb{N}} \ \cup \ 2^{\nu(K_2)}$. Then $P_2 \subseteq \mathbb{N}$ is a power-$\mathcal{H}$ predicate, because $\mathcal{H}$ includes the context-free languages. Thus, the language $L_2 = \{\mathtt{10}^n \mid n \in P_2\}$ belongs to $\mathcal{I}(\mathsf{power}\mathcal{H})$ and

$$L_2 = \{\mathtt{10}^n \mid n \in \mathbb{N} \setminus 2^{\mathbb{N}}\} \cup \{\mathtt{10}^{2^{\nu(w)}} \mid w \in K_2\}.$$

Moreover, let $L_1 := \{\mathtt{10}^{2^{\nu(w)}} \mid w \in K_1\}$. Since $L_1 = \{\mathtt{10}^{\nu(\mathtt{10}^{\nu(w)})} \mid w \in K_1\}$ and K_1 is an order-1 pushdown language, applying Lemma 5.4 twice yields that L_1 is an order-5 pushdown language and thus belongs to $\mathcal{H}$. Now clearly $L_1 \cap L_2 \neq \emptyset$ if and only if $\nu(K_1) \cap \nu(K_2) \neq \emptyset$, which is equivalent to $K_1 \cap K_2 = \emptyset$. ◀

For showing decidability of regular separability, we use the following well-known fact (please see the full version [40] for a proof).

▶ **Lemma 5.6.** *Let $L = \bigcup_{i=1}^{m} L_i$ and $K = \bigcup_{i=1}^{n} K_i$. Then $K|L$ if and only if $L_i|K_j$ for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$.*

The last ingredient for our decision procedure is the following simple but powerful observation from [12] (for the convenience of the reader, a proof can be found in [40]).

▶ **Lemma 5.7.** *Let $K \subseteq \Gamma^*$, $L \subseteq \Sigma^*$ and $T \subseteq \Sigma^* \times \Gamma^*$ be a rational transduction. Then $L|TK$ if and only if $T^{-1}L|K$.*

The following now completes the proof of Theorem 5.2.

▶ **Lemma 5.8.** *The problem $\mathsf{RS}(\mathcal{H}, \mathcal{I}(\mathsf{power}\mathcal{H}))$ is decidable.*

Proof. Suppose we are given $L_1 \subseteq \Sigma^*$ from $\mathcal{H}$ and $L_2 \subseteq \Sigma^*$ from $\mathcal{I}(\mathsf{power}\mathcal{H})$. Then we can write $L_2 = \bigcup_{i=1}^{n} T_i\mathtt{a}^{P_i}$, where for $1 \leq i \leq n$, $T_i \subseteq \Sigma^* \times \mathtt{a}^*$ is a rational transduction and $P_i \subseteq \mathbb{N}$ is a power-$\mathcal{H}$ predicate. Since $L_1|L_2$ if and only if $L_1|T_i\mathtt{a}^{P_i}$ for every i (Lemma 5.6), we may assume $L_2 = T\mathtt{a}^P$ for $T \subseteq \Sigma^* \times \mathtt{a}^*$ rational and $P \subseteq \mathbb{N}$ a power-$\mathcal{H}$ predicate. According to Lemma 5.7, $L_1|T\mathtt{a}^P$ if and only if $T^{-1}L_1|\mathtt{a}^P$. Since T^{-1} is also a rational transduction and $\mathcal{H}$ is a full trio, we may assume that L_1 is in $\mathcal{H}$ with $L_1 \subseteq \mathtt{a}^*$ and $L_2 = \mathtt{a}^P$.

By Lemma 4.5, we know that $L_1|\mathtt{a}^P$ if and only if L_1 is finite and disjoint from $\mathtt{a}^P$. We can decide this as follows. First, using Lemma 5.3 we check whether L_1 is finite. If it is not, then we know that $L_1|L_2$ is not the case.

If L_1 is finite, then we can compute a list of all words in L_1: We start with $F_0 = \emptyset$ and then successively compute finite sets $F_i \subseteq L_1$. For each $i \in \mathbb{N}$, we check whether $L_1 \subseteq F_i$, which is decidable because $L_1 \cap (\mathtt{a}^* \setminus F_i)$ is in $\mathcal{H}$ and emptiness is decidable for $\mathcal{H}$. If $L_1 \not\subseteq F_i$, then we enumerate words in $\mathtt{a}^*$ until we find $\mathtt{a}^m$ with $\mathtt{a}^m \in L_1$ (membership in L_1 is decidable) and $\mathtt{a}^m \notin F_i$. Then, we set $F_{i+1} = F_i \cup \{\mathtt{a}^m\}$. Since L_1 is finite, this procedure must terminate with $F_i = L_1$. Now we have $L_1|\mathtt{a}^P$ if and only if $F_i \cap \mathtt{a}^P = \emptyset$. The latter can be checked because $\mathsf{power}\mathcal{H}$ predicates are decidable. ◀

6 Conclusion

We have presented a language class $\mathcal{C}_1$ for which intersection emptiness is decidable but regular separability is undecidable in Section 4. Similarly, in Section 5 we constructed $\mathcal{C}_2, \mathcal{D}_2$ for which intersection emptiness is undecidable but regular separability is decidable. All three language classes enjoy pleasant language theoretic properties in that they are full trios and have a decidable emptiness problem.

Let us provide some intuition on why these examples work. The underlying observation is that intersection emptiness of two sets is insensitive to the shape of their members: If $f\colon X \to Y$ is any injective map and S disjoint from the image of f, then for $A, B \subseteq X$, we have $A \cap B = \emptyset$ if and only if $(f(A) \cup S) \cap f(B) = \emptyset$. Regular separability, on the other hand, is affected by such distortions: For example, if $K, L \subseteq \mathtt{1}\{\mathtt{0},\mathtt{1}\}^*$ are infinite, then $\mathtt{a}^{\mathbb{N}\setminus 2^{\mathbb{N}}} \cup \mathtt{a}^{2^{\nu(K)}}$ and $\mathtt{a}^{2^{\nu(L)}}$ are never regular-separable, even if K and L are. Hence, roughly speaking, the examples work by distorting languages (using encodings as numbers) so that intersection emptiness is preserved, but regular separability reflects infinity of the input languages. We apply this idea to language classes where intersection is decidable, but infinity is not (Theorem 4.1) or the other way around (Theorem 5.2). All this suggests that regular separability and intersection emptiness are fundamentally different problems.

Moreover, our results imply that any simple combinatorial decision problem that characterizes regular separability has to be incomparable with intersection emptiness. Consider for example the *infinite intersection problem* as a candidate. It asks whether two given languages have an infinite intersection. Note that if L and K are languages from $\mathcal{C}$ and $\mathcal{D}$, respectively, then $L \cap K \neq \emptyset$ if and only if $L\#^*$ and $K\#^*$ (where $\#$ is a symbol not present in L or K) have infinite intersection. Moreover, if $\mathcal{C}$ and $\mathcal{D}$ are full trios, then they effectively contain $L\#^*$ and $K\#^*$, respectively. This implies a counterexample with decidable regular separability and undecidable infinite intersection.

While the example from Section 4 is symmetric (meaning: the two language classes are the same) and natural, the example in Section 5 is admittedly somewhat contrived: While pseudo-$\mathcal{C}$ predicates rely on the common conversion of binary into unary representations, power-$\mathcal{C}$ predicates are a bit artificial. It would be interesting if there were a simpler symmetric example with decidable regular separability and undecidable intersection.

References

1. Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 313–321. IEEE, 1996.
2. Jorge Almeida. Some algorithmic problems for pseudovarieties. *Publ. Math. Debrecen*, 54(1):531–552, 1999.
3. Jean Berstel. *Transductions and context-free languages.* Springer-Verlag, 2013.
4. Achim Blumensath and Erich Gradel. Automatic structures. In *Proceedings of LICS 2000*, pages 51–62. IEEE, 2000.
5. Mikołaj Bojańczyk. It is Undecidable if Two Regular Tree Languages can be Separated by a Deterministic Tree-walking Automaton. *Fundam. Inform.*, 154(1-4):37–46, 2017.
6. Ahmed Bouajjani, Javier Esparza, and Tayssir Touili. A generic approach to the static analysis of concurrent programs with procedures. *International Journal of Foundations of Computer Science*, 14(04):551–582, 2003.
7. Christian Choffrut and Serge Grigorieff. Separability of rational relations in $A^* \times \mathbb{N}^m$ by recognizable relations is decidable. *Information processing letters*, 99(1):27–32, 2006.

8 Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Regular Separability of Parikh Automata. In *Proceedings of ICALP 2017*, pages 117:1–117:13, 2017.

9 Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Separability of Reachability Sets of Vector Addition Systems. In *Proceedings of STACS 2017*, pages 24:1–24:14, 2017.

10 Wojciech Czerwiński, Slawomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular Separability of Well-Structured Transition Systems. In *Proceedings of CONCUR 2018*, pages 35:1–35:18, 2018. doi:10.4230/LIPIcs.CONCUR.2018.35.

11 Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, Marc Zeitoun, and Georg Zetzsche. A Characterization for Decidable Separability by Piecewise Testable Languages. *Discrete Mathematics & Theoretical Computer Science*, 19(4), 2017.

12 Wojciech Czerwiński and Georg Zetzsche. An Approach to Regular Separability in Vector Addition Systems, 2019. In preparation.

13 Jürgen Dassow and Gheorghe Păun. *Regulated Rewriting in Formal Language Theory*. Springer, Heidelberg, 1989.

14 Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset Nets Between Decidability and Undecidability. In *Proceedings of ICALP 1998*, pages 103–115, 1998.

15 Herbert B Enderton. *A mathematical introduction to logic*. Elsevier, 2001.

16 Joost Engelfriet. Context-Free Grammars with Storage. *CoRR*, abs/1408.0683, 2014. arXiv:1408.0683.

17 Alain Finkel and Philippe Schnoebelen. Fundamental Structures in Well-Structured Infinite Transition Systems. In *Proceedings of LATIN 1998*, pages 102–118, 1998.

18 Gilles Geeraerts, Jean-François Raskin, and Laurent Van Begin. Well-structured languages. *Acta Informatica*, 44(3-4):249–288, 2007.

19 Seymour Ginsburg and Sheila Greibach. Principal AFL. *Journal of Computer and System Sciences*, 4(4):308–338, 1970.

20 Samuel J.v. Gool and Benjamin Steinberg. Pointlike sets for varieties determined by groups. *Advances in Mathematics*, 348:18–50, 2019.

21 Jean Goubault-Larrecq and Sylvain Schmitz. Deciding Piecewise Testable Separability for Regular Tree Languages. In *Proceedings of ICALP 2016*, pages 97:1–97:15, 2016.

22 Sheila A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7(3):311–324, 1978.

23 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and Downward Closures of Higher-order Pushdown Automata. In *POPL 2016*, pages 151–163, New York, NY, USA, 2016. ACM.

24 Juris Hartmanis. Context-free languages and Turing machine computations. In *Proceedings of Symposia in Applied Mathematics*, volume 19, pages 42–51, 1967.

25 Harry B. Hunt III. On the Decidability of Grammar Problems. *Journal of the ACM*, 29(2):429–447, 1982. doi:10.1145/322307.322317.

26 Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. In *Logic and Computational Complexity*, volume 960 of *LNCS*, pages 367–392, Berlin, Heidelberg, 1995. Springer.

27 Felix Klaedtke and Harald Rueß. Monadic Second-Order Logics with Cardinalities. In *Proceedings of ICALP 2003*, volume 2719 of *LNCS*, pages 681–696, Springer, Heidelberg, 2003. Springer.

28 Eryk Kopczyński. Invisible Pushdown Languages. In *Proceedings of LICS 2016*, pages 867–872, New York, NY, USA, 2016. ACM.

29 Sławomir Lasota and Wojciech Czerwiński. Regular Separability of One Counter Automata. *Logical Methods in Computer Science*, 15, 2019. Extended version of LICS 2017 paper.

30 Richard Mayr. Undecidable problems in unreliable computations. *Theoretical Computer Science*, 297(1-3):337–354, 2003.

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

32 Thomas Place. Separating Regular Languages with Two Quantifiers Alternations. In *Proceedings of LICS 2015*, pages 202–213, 2015.

33 Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating Regular Languages by Piecewise Testable and Unambiguous Languages. In *Proceedings of MFCS 2013*, pages 729–740, 2013.

34 Thomas Place and Marc Zeitoun. Separation and the Successor Relation. In *Proceedings of STACS 2015*, pages 662–675, 2015.

35 Thomas Place and Marc Zeitoun. Separating Regular Languages with First-Order Logic. *Logical Methods in Computer Science*, 12(1), 2016.

36 Thomas Place and Marc Zeitoun. Concatenation Hierarchies: New Bottle, Old Wine. In *Proceedings of CSR 2017*, pages 25–37, 2017.

37 Thomas Place and Marc Zeitoun. Separation for dot-depth two. In *Proceedings of LICS 2017*, pages 1–12, 2017.

38 Thomas Place and Marc Zeitoun. The Covering Problem. *Logical Methods in Computer Science*, 14(3), 2018.

39 Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2), 1976.

40 Ramanathan S Thinniyam and Georg Zetzsche. Regular Separability and Intersection Emptiness are Independent Problems. *arXiv preprint*, 2019. `arXiv:1908.04038`.

41 Georg Zetzsche. An Approach to Computing Downward Closures. In *Proceedings of ICALP 2015*, pages 440–451, 2015.

42 Georg Zetzsche. Separability by piecewise testable languages and downward closures beyond subwords. In *Proceedings of LICS 2018*, pages 929–938, 2018.