# Description of scientific achievements

## 1 Name

Wojciech Karol Czerwiński

## 2 Diplomas and scientific degrees

| | |
|---|---|
| 2009–2013 | Ph.D. studies in Computer Science, University of Warsaw |
| 21.03.2013 | Ph.D. degree in Computer Science<br>thesis title: *Partially-commutative context-free graphs*<br>advisor: Sławomir Lasota |
| 2002–2008 | M.Sc. studies in Mathematics, University of Warsaw |
| 26.09.2009 | M.Sc. degree in Mathematics<br>thesis title: *Khintchin type inequalities*<br>advisor: Rafał Latała |
| 2002–2007 | M.Sc. studies in Computer Science, University of Warsaw |
| 27.09.2007 | M.Sc. degree in Computer Science<br>thesis title: *Simulation between games*<br>advisor: Damian Niwiński |

## 3 Employment in academia

### 3.1 Current position

since 10.2013 assistant professor (ori. *adiunkt*) at the Institute of Informatics, University of Warsaw

### 3.2 Previous positions

07.2012 – 06.2013 researcher at University of Bayreuth

## 4 Main scientific achievement

### 4.1 Title of the achievement

Separability and reachability problems in infinite-state systems.

### 4.2 Publications that are part of the achievement

1. Wojciech Czerwiński, Wim Martens, Tomás Masopust
   *Efficient Separability of Regular Languages by Subsequences and Suffixes.*
   In Proc. of ICALP '13, pages 150–161.

2. Wojciech Czerwiński, Sławomir Lasota
   *Regular separability of one counter automata.*
   In Proc. of LICS '17, pages 1–12.

3. Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, Charles Paperman
   *Separability of Reachability Sets of Vector Addition Systems.*
   In Proc. of STACS '17, pages 24:1–24:14.

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

5. Wojciech Czerwiński, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdziński, Ranko Lazić, Paweł Parys
*Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games.*
In Proc. of SODA '19, pages 2333–2349.

6. Wojciech Czerwiński, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, Filip Mazowiecki
*The Reachability Problem for Petri Nets is Not Elementary.*
Accepted to STOC '19.

## 4.3 Description of the achievement

Six above mentioned works are all in the topics of separability or reachability in infinite-state systems.

### 4.3.1 Introduction

**Basic computation models** Investigation of models of computation is an important and active area of theoretical computer science since 70-ties. Such models, called infinite-state systems, are usually infinite, directed graphs with edges labelled by letters from some finite alphabet. Infinite-state systems have to be finitely describable, as otherwise one cannot design any algorithms, which take them as input. The most popular, and also fundamental, models of computation are several extensions of finite automata: Turing machines, pushdown automata and automata with counters (where increments, decrements and sometimes zero tests are allowed). Many more other models have been considered for decades, they differ on structures they have and operations, which they can apply. Most of them use a few basic constructions: stack, counter, infinite data domain, clock but in many different combinations, modifications or restrictions. This is why it is so important to understand the basics, they can influence everything else.

Several fundamental questions have been asked for various infinite-state systems. Most basic one is probably the reachability problem: given two nodes of the system decide whether there is a path between them. This problem does not take into account labels of on the edges. Based on labels one can define languages of infinite-state systems. Given two nodes $u$ and $v$ one can define language $L(u, v)$ in a natural way: as the set of words, which are labelings of some path from $u$ to $v$. Reachability problem can be then reformulated as a question whether language $L(u, v)$ for given $u$ and $v$ is empty. However one can ask many other natural questions, which have been considered often in the literature: 1) is $L(u, v)$ finite?, 2) is $L(u, v)$ universal?, i.e. does it contain all words, 3) is $L(u, v)$ regular?, 4) does $L(u, v)$ belong to another fixed language family $\mathcal{F}$?, etc. Asking these questions leads to better understanding of the considered computational models.

**Separability problem** In my work I have often considered another problem, which was not so popular till recently, namely *separability problem*. For two languages $K, L \subseteq \Sigma^*$ over alphabet $\Sigma$ language $S \subseteq \Sigma^*$ *separates $K$ and $L$* if $S$ includes $K$ and is disjoint from $L$. For two families $\mathcal{F}$, $\mathcal{G}$ of languages over $\Sigma$ the *$\mathcal{F}$-separability problem for $\mathcal{G}$* asks for two given languages $K, L \in \mathcal{G}$ whether there exists a language in $\mathcal{F}$, which separates $K$ and $L$. We say simply *$\mathcal{F}$-separability problem* if $\mathcal{G}$ is clear from the context. For classes $\mathcal{G}$ closed under complement the $\mathcal{F}$-separability problem generalizes the *$\mathcal{F}$-membership problem*, which asks whether given $L \in \mathcal{G}$ belongs to $\mathcal{F}$. Indeed $L \in \mathcal{F}$ if and only if $L$ and $\bar{L}$, i.e. the complement of $L$, are separated by a language from $\mathcal{F}$.

### 4.3.2 Separabililty for regular languages

**Membership problem** The $\mathcal{F}$ membership problem for regular languages is a prominent problem, which gained a lot of attention for various subclasses $\mathcal{F}$ of regular languages. The seminal work by Schutzenberger [Sch65] characterizes regular languages definable in first-order logic as those, for which their syntactic monoid contains no nontrivial group. This provides an easy decidability procedure for the membership problem. Another famous result by Simon [Sim75] characterizes piecewise testable languages. Regular language is a *piecewise testable* language if it is a finite boolean combination of languages of a form $\Sigma^* a_1 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$, where $a_1, \ldots, a_n \in \Sigma$ are letters. Simon's theorem states, that regular language is piecewise testable if and only if its syntactic monoid is $\mathcal{J}$-trivial, which means that all its $\mathcal{J}$-classes are singletons. The latter condition is easily decidable. Membership problem was also considered for other subclasses of regular languages and many cases were known. The used techniques usually involve algebraic techniques applied to analysis of syntactical monoid of a given regular language, which can be observed on two mentioned above prominent examples. Despite of the popularity of membership problem till recently the separability problem was not so often considered by the automata theory community.

**Piecewise-testable-separability** In 2013 together with Wim Martens and Thomas Masopust we were considering a problem in database theory, which led us to a result on separability by piecewise testable languages (PTL) [CMM13]. This paper is a part of my main achievement. Main result of this article is a characterization of pairs of languages $K$ and $L$, which cannot be separated by a piecewise testable language. Second main result, which builds on that characterization, is a polynomial time algorithm solving PTL-separability problem for regular languages.

Before stating this result we need to introduce a few notions. We say that an order $(X, \leq)$ is a well quasi order (wqo) if there is neither an infinite decreasing sequence wrt. $\leq$ nor an infinite antichain wrt. $\leq$. Consider a subsequence order on finite words, which is important for our considerations. A word $u = a_1 \cdots a_n$ is a *subsequence* of $v$, denoted $u \preceq v$, if $v \in \Sigma^* a_1 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$. Well known Higman's lemma states that the subsequence order is wqo [Hig52]. We say that a set $S$ is $\leq$*-closed* if $x \in S$ and $x \leq y$ implies that $y \in S$. The key notion introduced in our work is a *zigzag*. Infinite sequence of words $w_1, w_2, \ldots \in \Sigma^*$ is an *infinite* $\leq$*-zigzag* between languages $K, L \subseteq \Sigma^*$ if $w_1 \in K \cup L$ and for each $i \geq 1$ the following conditions hold

1. $w_i \leq w_{i+1}$;
2. if $w_i \in K$ then $w_{i+1} \in L$;
3. if $w_i \in L$ then $w_{i+1} \in K$.

In other words sequence $w_i$ is increasing wrt. $\leq$ and alternates between languages $K$ and $L$.

The main result of [CMM13] is Theorem 3, which states that

**Theorem 1.** *For languages $K$ and $L$ and a wqo $\leq$ on words, the following are equivalent*

1. *$K$ and $L$ are separable by a finite boolean combination of $\leq$-closed languages*
2. *there does not exist an infinite $\leq$-zigzag between $K$ and $L$.*

In fact Theorem 1 claims a bit more, there is also a third equivalent condition, but it is not as important here as the two mentioned above. As mentioned before the subsequence order is wqo. So Theorem 1 implies that PTL-separability of two languages is equivalent to nonexistence of infinite $\preceq$-zigzag between them.

An interesting part of Theorem 1 is that it speaks about separability by PTL, a subclass of regular languages, but it does not use any algebraic techniques. In particular languages $K$ and $L$ does not have to be regular. To my best knowledge it was the first published result of that kind avoiding consideration of syntactic monoid.

Second main result of [CMM13] is a polynomial time algorithm deciding whether there is an infinite $\preceq$-zigzag between two given regular languages. This in particular gives a polynomial time algorithm deciding PTL-separability of regular languages.

In [CMM13] we also have considered separability problem by other classes than PTL, which were based not on subsequence order, but on prefix or suffix order.

In recent few years the topic of separating regular languages by their subclasses was developed very much, mainly by Thomas Place and Marc Zeitoun. They have shown in particular that given two regular languages it is decidable whether they are separable by first order definable languages [PZ14b], languages on the second level of dot depth hierarchy [PZ17] and many others. It also turned out that considering separability problem helps a lot to decide membership problem for other subclasses [PZ14a]. This indicates that separability problem is a natural one, deeply connected with expressive power of the given class of languages.

### 4.3.3 Separabililty beyond regular languages

The fact that the notion of zigzag can be applied also to nonregular languages results inspired us to think about input languages being not only regular.

**Negative results** However, already in 70ties and 80ties there were some negative results, which seem to be a strong indication that obtaining any reasonable decidability results for nonregular languages as input can be hard or impossible. Already in 1976 Szymanski and Williams [SW76] have shown that regular-separability is undecidable for context-free languages. A few years later Hunt has even strengthened this result [Hun82]. Language $L \subseteq \Sigma^*$ is *definite* if there is a number $k \in \mathbb{N}$ such belonging to $L$ depends only on prefix of length $k$. In other words if $u, v \in \Sigma^*$, have the same prefixes of length $k$ then $u \in L \iff v \in L$. Hunt has proven that for any family of languages $\mathcal{F}$, which contains all the definite languages the problem of $\mathcal{F}$-separability is undecidable for context-free languages. Containing all the definite languages is a very weak condition. Not only regular languages satisfy it, but also for example languages definable in FO (first order logic), languages definable in FO with only two variables and many others. Essentially it is enough for a language to be definable in some logic, which can test which letter is on $k$-th position in the word, for a fixed $k \in \mathbb{N}$. Piecewise testable languages are not definite, they are not even capable to test the first letter of a word, for example language $a\Sigma^*$ is not piecewise testable. Therefore decidability of PTL-separability for context-free languages (and other languages classes) is not excluded.

**Common patterns** In [CMvR$^+$17], which is a second part of my main achievement we consider PTL-separability for inputs, which are not regular languages. Conference version of this work is [CMvRZ15]. The whole approach is based on the Theorem 2.1 from our work, which connects PTL-separability with *patterns*. This notion seem to be more appropriate for our work than zigzags. A *pattern* is a pair $(\vec{u}, \vec{B})$, where $u_0, \ldots, u_p \in \Sigma^*$ are words, $B_1, \ldots, B_p \subseteq \Sigma$ are subalphabets, $\vec{u} = (u_0, \ldots, u_p)$ and $\vec{B} = (B_1, \ldots, B_p)$. An *alphabet* of a word $w \in \Sigma$ is the set of all the letters from $\Sigma$, which occur in $w$. We denote it $\text{Alph}(w)$. For $B \subseteq \Sigma$ by $B^{\otimes}$ we denote the set of all the words with alphabet being exactly $B$, $B^{\otimes} = \{w \mid \text{Alph}(w) = B\}$. Then for a pattern $(\vec{u}, \vec{B})$ we define languages

$$L(\vec{u}, \vec{B}, n) = u_0 \big(B_1^{\otimes}\big)^n u_1 \cdots u_{p-1} \big(B_p^{\otimes}\big)^n u_p.$$

Language $L$ *contains the pattern* $(\vec{u}, \vec{B})$ if there is an infinite sequence of words $w_1, w_2, \ldots \in L$ such that for each $n \in \mathbb{N}$ it holds $w_n \in L(\vec{u}, \vec{B}, n)$. Theorem 2.1 in [CMvR$^+$17] says the following.

**Theorem 2.** *Two word languages $K$ and $L$ are not PTL-separable if and only if they contain a common pattern.*

**Separability of context-free languages** Let us see how one can use Theorem 2 to show that PTL-separability is decidable for context-free languages. Let $K$ and $L$ be two context-free languages. We design two semi-procedures, one (*positive*) looks for a witness that $K$ and $L$ are PTL-separable, while the other (*negative*) looks for a witness that $K$ and $L$ are not PTL-separable. Designing positive semi-procedure is an easy task: simply iterate through all the possible languages $S$ in PTL and for each one check whether it includes $K$ and is disjoint from $L$. Checking inclusion and disjointness are special cases of checking emptiness of intersection of regular and context-free languages. Therefore it is easily decidable. If $K$ and $L$ are PTL-separable we find at some moment language $S$ witnessing it.

For the negative semi-procedure we use Theorem 2. We iterate through all the patterns and for each one check whether it is contained both in $K$ and $L$. If $K$ and $L$ are not separable we find at some moment a pattern witnessing it. So the problem boils down to checking for a given pattern $s$ and a given context-free language $L$ whether $L$ contains $s$. We solve this problem in details in [CMvR$^+$17], even for more general case, which I discuss in a moment. A high level idea behind the solution is as follows. A pattern $s = (\vec{u}, \vec{B})$ with $\vec{u} = (u_0, \ldots, u_p)$ and $\vec{B} = (B_1, \ldots, B_p)$ is contained in $L$ roughly speaking when there are words in $L$, which contain first arbitrary many copies of $B_1$, then arbitrary many copies of $B_2$, etc. and at the end arbitrary many copies of $B_p$. In addition between these copies we need to have words $u_i$. However words $u_i$ and concrete form of $B_i$ are technical details. One can compute for a language $L$ and a pattern $s$ another language $\hat{L} \subseteq a_1^* a_2^* \cdots a_p^*$ such that $L$ contains pattern $s$ if and only if $\hat{L}$ contains for every $n$ words of a form $a_1^{n_1} a_2^{n_2} \cdots a_p^{n_p}$ where all $n_i \geq n$. Intuitively in $\hat{L}$ words $u_i$ are omitted, while letters $a_i$ play roles of alphabets $B_i$. This phenomenon inspired us to define the following *diagonal problem*: given language $L \subseteq \{a_1, \ldots, a_k\}$, decide whether for every $n \in \mathbb{N}$ Parikh image of $L$ contains tuples $(n_1, \ldots, n_k)$ such that all $n_i \geq n$. Recall that *Parikh image* of a language $L$ is the set of tuples $(n_1, \ldots, n_k) \in \mathbb{N}^k$ such that there is a word in $L$, which contains exactly $n_1$ occurrences of letter $a_1$, exactly $n_2$ occurrences of letter $a_2$ etc. and exactly $n_k$ occurrences of letter $a_k$. Diagonal problem is decidable for context-free languages, which, together with above ideas, gives decidability of PTL-separability for context-free languages.

**Generalization** One can easily observe that above proof idea can be generalized to any family of languages $\mathcal{F}$, which satisfy the following conditions:

1. checking whether given regular language and a language from $\mathcal{F}$ intersect is decidable,

2. transformation from $L$ to $\hat{L}$, with properties as above, can be computed,

3. diagonal problem is decidable.

It turns out that it is possible for all language families, which are *full trios*. One of the definitions is that family $\mathcal{F}$ is a *full trio* if it is effectively closed under rational transductions, i.e. images of some kind of transducers. Here we do not go into details, it is enough to know that for full trios all the regular-flavor operations, like intersection with regular language, ignoring letters from some alphabet etc. are computable. In [CMvR$^+$17] we have shown that for all full trios, for which diagonal problem is decidable also the PTL-separability problem is decidable.

**Equivalence** A very interesting fact is that also the opposite implication is true: for every full trio, for which PTL-separability problem is decidable also the diagonal problem is decidable. So for full trios decidability of PTL-separability problem and of diagonal problem is equivalent. In order to fully present our main result from [CMvR$^+$17] we need to introduce a few more notions.

For a language $L$ its *downward closure*, denoted $L\downarrow$, is the set of all the subsequences of words from $L$, $L\downarrow = \{u \mid \exists_{v \in L} u \preceq v\}$. A *simultaneous unbounded problem* (SUP) asks whether given $L \subseteq a_1^* a_2^* \cdots a_n^*$ satisfies $L\downarrow = a_1^* a_2^* \cdots a_n^*$. Notice that SUP is similar to diagonal problem, the only difference is that its input satisfy a special requirement that letters are "sorted", i.e. at first occur $a_1$, then $a_2$, etc. We are ready to formulate the main result, Theorem 2.5.

**Theorem 3.** *For a language family $\mathcal{F}$ being a full trio the following are equivalent:*

1. *PTL-separability is decidable for $\mathcal{F}$,*
2. *diagonal problem is decidable for $\mathcal{F}$,*
3. *SUP is decidable for $\mathcal{F}$,*
4. *downward closures of languages from $\mathcal{F}$ are computable.*

**Decidable classes** As a corollary of Theorem 3 many language families have decidable PTL-separability problem, which is also mentioned in [CMvR$^+$17]. We prove there directly that diagonal problem is decidable for context-free languages, which makes them first class beyond regular languages, which has decidable PTL-separability. However there are known results for other language families, which are sufficient to get decidability for them. In [HKO16] downward closures of higher order pushdown automata were shown to be computable, which gives decidability of PTL-separability for that class. We also provide a proof that diagonal problem is decidable for languages of vector addition systems with states (VASS). The proof is a reduction to place-boundedness problem for VASes with one zero test, which was shown to be decidable in [BFLZ10]. The other way to show decidability of PTL-separability for VASS languages is to use the work [HMW10], where computability of downward closures for VASS languages was shown. Last result of that kind is that if PTL-separability is decidable for language families $\mathcal{F}_1$ and $\mathcal{F}_2$ then it is also decidable for their union $\mathcal{F}_1 \cup \mathcal{F}_2$. In other words we can for example decide whether given VASS language and given context-free language are PTL-separable.

### 4.3.4 Separability by regular languages

Results of [CMvR$^+$17] have shown that a lot of separability results is possible beyond regular languages. In [CMvR$^+$17] we have considered PTL-separability, however it is not the most natural class of separators, which can be considered beyond regular languages. The most natural class of separators beyond regular languages are the regular languages itself. Therefore a lot of my later work was (and still is) about regular-separability for different families of languages, which are often languages of infinite state systems. I believe that regular-separability problem is another very natural problem and by considering it one can get inside into some fundamental infinite-state systems, in particular into vector addition systems.

**Obstacles** As mentioned above already in the 70ties it was shown in [SW76] that regular-separability is undecidable for context-free languages. Recently Kopczyński has shown even a stronger result [Kop16] that regular-separability is undecidable for visibly pushdown languages (which are determinizable). Clear message of these results is that one cannot hope for decidability of regular-separability in presence of a stack. An intuition behind the separability problem is that it somehow resembles emptiness of the intersection problem. These ideas led us to consider automata with counters as inputs.

**Vector addition systems** Let us recall that a $d$-dimensional vector addition system (VAS) is a set of transitions $T \subseteq \mathbb{Z}^d$ together with its labeling $\ell : T \to \Sigma \cup \{\varepsilon\}$ by letters of a finite alphabet $\Sigma$. Transition $v \in T$ can be fired in configuration $u \in \mathbb{N}^d$ under the condition that $u + v \in \mathbb{N}^d$. When given *source* $s$ and *target* $t$ configurations in $\mathbb{N}^d$ one can naturally define a language of a VAS as the set of labelings of all the paths from source to target. Such a language we call a *VAS language*. Very similar models, namely extension of VASes by states (vector addition systems with states - VASSes) and Petri nets were also very widely considered in the literature. In particular it is easy to show that families of VAS languages, VASS languages and Petri net languages coincide.

VASes are considered to be a fundamental model of computation since 70ties. There is a plenty of papers investigating various properties of them. Probably the most fundamental problem is the *reachability problem*, which asks whether there is a path from a given source to a given target. It can be seen as asking whether language of a given VAS is empty. Reachability problem was shown to be decidable by Mayr in 1981 [May81]. We will discuss later more about its complexity, as the current best lower bound, a TOWER-hardness is a part of my main achievement [CLL$^+$18]. A main message of the above is that emptiness of the intersection of two VAS languages is decidable, as it can be reduced to emptiness of the product of two considered VASes, so in fact to reachability problem. This fact and other arguments motivated us to state the following conjecture.

**Conjecture 4.** *Regular-separability problem is decidable for VAS languages.*

This conjecture has directed a research, which has led us to several results, which I am about to describe.

**Reachability sets of VASSes** The first work towards solving Conjecture 4 actually focused not on the languages of VASSes, but rather on its reachability sets [CCLP17b]. Set $S \subseteq \mathbb{N}^d$ is *VASS reachability set* if there is a VASS and its source configuration such that the set of all configurations reachable from the source and with same given state is exactly $S$. This results can be phrased in terms of languages, as we do in Corollary 6, however it is more natural to express it in terms of reachability sets.

For $u \in \mathbb{N}^d$ and $i \in \{1, \ldots, d\}$ let $u[i]$ be the $i$-th coordinate of vector $u$. For $n \in \mathbb{N}$ we say that $u \equiv_n v$ if for all $i \in \{1, \ldots, d\}$ we have $u[i] \equiv v[i] \mod n$. Set $S \subseteq \mathbb{N}^d$ is *modular* if there exist a number $n \in \mathbb{N}$ such that belonging to $S$ depends only on counter values modulo $n$. In other words if $u, v \in \mathbb{N}^d$ such that $u \equiv_n v$ then $u \in S \iff v \in S$. Set $S \subseteq \mathbb{N}^d$ is *unary* if there exist number $n \in \mathbb{N}$ such that above threshold $n$ set $S$ is modular modulo $n$. In other words if $u, v \in \mathbb{N}^d$ such that for every $i \in \{1, \ldots, d\}$ either $u[i] = v[i]$ or $u[i], v[i] \geq n$ and $u[i] \equiv v[i] \mod n$ then $u \in S \iff v \in S$. Recall that for alphabet $\Sigma = \{a_1, \ldots, a_k\}$ and word $w \in \Sigma^*$ its *Parikh image*, denoted $\textsc{pi}(w)$, it a vector $v \in \mathbb{N}^k$ such that $v[i]$ equals the number of occurrences of $a_i$ in $w$. Parikh image extends naturally to languages, $\textsc{pi}(L) = \{\textsc{pi}(w) \mid w \in L\} \subseteq \mathbb{N}^k$. One can easily observe that Parikh image of a regular language is always unary, which is why unary sets are important.

When analyzing VASS reachability sets it is very convenient to look at parts of reachability sets in which can coordinates are fixes. Roughly speaking we call such sets *sections*.

For a vector $v \in \mathbb{N}^d$ and set of coordinates $J \subseteq \{1, \ldots, d\}$ let $v[J] \in \mathbb{N}^{|J|}$ be vector $v$ restricted only to coordinates from $J$. Precisely speaking set $T \subseteq \mathbb{N}^c$ is a *section* of set $S \subseteq \mathbb{N}^{c+d}$ if there is a set of coordinates $J$, $|J| = d$ and numbers $a_1, \ldots, a_d \in \mathbb{N}$ such that

$$T = \{v[\{1, \ldots, d\} \setminus J] \mid v \in S \text{ and } v[J] = (a_1, \ldots, a_d)\}.$$

The main result of [CCLP17b] is the following.

**Theorem 5.** *The modular and unary-separability problems are decidable for sections of VASS reachability sets.*

*Commutative closure* of a language $L \subseteq \Sigma^*$ is the set of all words with the same Parikh image as some word in $L$. It is not hard to show that regular-separability of commutative closures of VASS languages reduces to unary-separability of their Parikh images. Therefore Theorem 5 implies the following.

**Corollary 6.** *The regular-separability problem for commutative closures of VASS languages is decidable.*

In order to give a flavor of a proof of Theorem 5 we recall that set $S \subseteq \mathbb{N}^d$ is *linear* if $S = \{b + n_1 v_1 + \ldots + n_k v_k \mid n_1, \ldots, n_k \in \mathbb{N}\}$ for some $b, v_1, \ldots, v_k \in \mathbb{N}^d$. Below we focus only on modular sets, argument works similarly for unary sets. The proof of Theorem 5 is based on the following lemma.

**Lemma 7.** *Let $U, V \subseteq \mathbb{N}^d$ be two sections of VASS reachability sets. Then the following conditions are equivalent:*

1. *$U$ and $V$ are not modular separable,*
2. *there exist linear sets $L_U \subseteq U$, $L_V \subseteq V$ such that $L_U$ and $L_V$ are not modular separable.*

Proof of Lemma 7 (and similar lemma for unary sets) is a technical core of [CCLP17b]. Based on Lemma 7 one can show decidability of modular-separability of two VASS sections $U$ and $V$ in the following way. We design two semi-procedures. First one, *positive*, tries to find a witness that $U$ and $V$ are modular separable. This one is simpler, it tries bigger and bigger numbers $n \in \mathbb{N}$ and for every single $n$ it is decidable (by reduction to reachability problem for VASSes) whether $U$ and $V$ are modular separable modulo $n$. If $U$ and $V$ are modular separable then positive semi-procedure definitely finds at some moment a witness for that. Second one, *negative*, tries to find a witness that $U$ and $V$ are not modular separable. By Lemma 7 such a witness can be a pair of linear sets $(L_U, L_V)$ fulfilling point 2. Therefore negative semi-procedure iterates through all pairs of linear sets (bigger and bigger) and for each one checks whether it fulfills condition 2. of Lemma 7. If $U$ and $V$ are not separable then it definitely finds at some moment such a witness. At first moment it is not clear that checking condition 2. is simpler than checking whether $U$ and $V$ itself are modular separable. However, it is indeed. There exist an easy algorithm for checking modular-separability of two linear sets. Checking whether given linear set is included in given section of VASS reachability set is decidable due to [Ler13]. In fact, in [CCLP17b], we provide also another proof, which does not require using additional results. We prove a stronger version of Lemma 7 for some *special form* of linear sets, for which it is clear from definition that they are included in $U$ and $V$. Therefore in that case it is enough that negative semi-procedure only checks whether $L_U$ and $L_V$ are modular separable.

**One counter automata** Next work from a sequence of papers is a part of my main achievement [CL17]. Its main contribution is the following one.

**Theorem 8.** *Regular-separability for one counter nets is* PSPACE*-complete.*

Recall that one counter net is the same as one dimensional VASS, so automaton with one counter, but without zero-tests. The central insight, which led to Theorem 8 was the notion of *approximant*. Let $\mathcal{A}$ be a one counter net with set of states $Q$, transitions $T$ of a form $(p, a, q, z)$, source configuration $(q_{\text{init}}, 0)$ and target configuration $(q_{\text{fin}}, 0)$. Then for each number $n \in \mathbb{N}$ we define the *$n$-th approximant* of $\mathcal{A}$, denoted $\mathcal{A}_n$, to be the following finite automaton:

- states of $\mathcal{A}_n$ are $Q \times \{0, \ldots, n-1\} \times \{\text{low}, \text{high}\}$,
- each transition $(p, a, q, z)$ gives rise to a number of transitions in $\mathcal{A}_n$

$$\begin{aligned}
&(q, c, \text{low}) \xrightarrow{a} (q, c{+}z, \text{low}) && \text{if } 0 \leq c{+}z < n \\
&(q, c, \text{low}) \xrightarrow{a} (q, (c{+}z) \mod n, \text{high}) && \text{if } n \leq c{+}z \\
&(q, c, \text{high}) \xrightarrow{a} (q, (c{+}z) \mod n, \text{low}) && \text{if } c{+}z < 0 \\
&(q, c, \text{high}) \xrightarrow{a} (q, (c{+}z) \mod n, \text{high}).
\end{aligned}$$

The idea behind the approximant $\mathcal{A}_n$ is that its counter simulates exactly $\mathcal{A}$ till it does not exceed value $n$. If it exceeds value $n$ then $\mathcal{A}_n$ only keeps counter value modulo $n$ and checks that the end of the computation looks correctly. For every $n$ we have $L(\mathcal{A}) \subseteq L(\mathcal{A}_n)$ and approximants are getting smaller with increasing $n$ in the following sense: for $n$ being multiple of $k$ we have $L(\mathcal{A}_n) \subseteq L(\mathcal{A}_k)$. The following lemma is the key insight (Corollary 10 from [CL17]).

**Lemma 9.** *For two one counter nets $\mathcal{A}$ and $\mathcal{B}$ the following conditions are equivalent:*

1. *$L(\mathcal{A})$ and $L(\mathcal{B})$ are regular separable,*
2. *$L(\mathcal{A}_n)$ and $L(\mathcal{B})$ are disjoint, for some $n > 0$,*
3. *$L(\mathcal{A}_n)$ and $L(\mathcal{B}_n)$ are disjoint, for some $n > 0$.*

Therefore in order to decide regular-separability it is enough to focus on point (2). It is possible to check it in PSPACE. Very roughly speaking one considers a kind of product of $\mathcal{A}_n$ and $\mathcal{B}$, which is similar to two dimensional VASS (2-VASS). Using some characterizations regarding reachability in 2-VASSes from [BFG$^+$15] and other results from the area we design a procedure working in PSPACE, which decides whether (2) holds for given $\mathcal{A}$ and $\mathcal{B}$.

In [CL17] we also show two other results. First one is PSPACE-hardness for the regular-separability problem for one counter nets, which is the second part of Theorem 8. Second one is undecidability of regular-separability for one counter automata (i.e. zero test is present now). We use an interesting, in my opinion, technique, which I have seen only once in other place (in [Hun82]). This technique is to show undecidability not by a reduction from an undecidable problem, but rather by a fixed blowup reduction from every decidable problem. Then use of space (or time) hierarchy theorem leads to the conclusion that our problem is undecidable.

**Integer VASSes** We also have considered another problem, which is a special case of Conjecture 4. The paper solving this problem [CCLP17a] is however not a part of my main achievement, so I will mention it only briefly. I decided not to put it into the main achievement list, as it is not good to have too many papers on the list. However, this paper can be as well on the main achievement list according to any other rule.

In [CCLP17a] we show decidability of the regular-separability problem for integer VASSes ($\mathbb{Z}$-VASSes). Integer VASSes are exactly like VASSes with the only difference that we allow counters to be negative. One can easily show that languages of $\mathbb{Z}$-VASSes are a subclass of languages of VASSes, so this result is really a special case of Conjecture 4.

**Languages of WSTSes** Another work, similarly as [CCLP17a], not included in my main achievement, but being a natural part of that line of research is [CLM$^+$18]. We have shown there that for languages of Well Structured Transition Systems (WSTSes) $K$ and $L$ fulfilling some mild conditions regular-separability of $K$ and $L$ is equivalent of emptiness of intersection $K \cap L$. In particular this holds if $K$ and $L$ are languages of VASSes with acceptance condition defined by state (not by a configuration).

### 4.3.5 Separation helping parity games

*Parity games* are two player games played on a graph $G = (V, E)$. Set of vertices $V$ is partitioned into set $V_0$ vertices owned by player *Even* and set $V_1$ vertices owned by player *Odd*. Every edge is labelled by a *rank* given by a function rank : $E \to \mathbb{N}$. A *play* starts in distinguished vertex, owner of a current vertex chooses an outgoing edge and play moves to vertex pointed by this edge. Play continues this way. Formally play is a sequence of edges $e_0, e_1, \ldots$ chosen as described above. Player Even wins a play if the largest number, which occurs infinitely many times in the sequence $\text{rank}(e_0), \text{rank}(e_1), \ldots$ is even, player Odd wins otherwise.

Parity games are central to many topics in automata theory and logic. In particular deciding a winner in parity games is equivalent to model checking of $\mu$-calculus formula. Settling computation complexity of solving parity games (i.e. deciding the winner) is a big open problem since many years. In particular there is a well known conjecture that parity games can actually be solved in polynomial time. Till recently fastest algorithms were mildly subexponential [JPZ08] of complexity approximately $\mathcal{O}(n^{\sqrt{n}})$. Recently there was a big breakthrough in that area, quasi-polynomial algorithm solving parity games was presented in [CJK$^+$17], with complexity around $\mathcal{O}(n^{\log n})$. Soon after a few other quasi-polynomial algorithms for that problem appeared [JL17, Leh18].

In our work [CDF$^+$19], which is a part of my main achievement we show that all the known quasi-polynomial algorithms can be rephrased in terms of finding small separators of some two languages of infinite words. Moreover, we prove that this approach cannot lead directly to a polynomial time algorithm solving parity games by providing quasi-polynomial lower bound on the size of appropriate separators.

More precisely speaking, every play in a game with ranks on edges belonging to the set $\{1, \ldots, d\}$ can be encoded as an infinite word over alphabet $\Sigma_d = \{1, \ldots, d\}$. Let us denote by $\text{LimsupEven}_d$ set of infinite words over $\Sigma_d$, in which the biggest number, which occurs infinitely often is even and similarly by $\text{LimsupOdd}_d$ set of words, in which the biggest number occurring infinitely often is odd. Let a (parity game) graph be called *even* if highest rank on every its cycle is even. Similarly we call a (parity game) graph *odd* if highest rank on every its cycle is odd. Of course most of the games graphs are neither even nor odd. However, if a game graph is even then it is easy to see that every play in it is won by player Even. Similarly for odd games. For $n \in \mathbb{N}$ we define language $\text{EvenCycles}_{n,d} \subseteq \Sigma_d^\omega$ as those infinite words that encode an infinite path in an even game graph with at most $n$ vertices and ranks up to $d$. Notice that languages $\text{EvenCycles}_{n,d}$ can be treated as under-approximations of $\mathrm{LimsupEven}_d$, we have the following inclusions:

$$\mathrm{EvenCycles}_{1,d} \subsetneq \mathrm{EvenCycles}_{2,d} \subsetneq \cdots \subsetneq \mathrm{LimsupEven}_d.$$

Similarly we define $\mathrm{OddCycles}_{n,d}$. We say that an automaton on infinite words is a *safety automaton* if it has a distinguished set of *rejecting states*. A word is accepted by such an automaton if there is a run over this word such that no rejecting state is visited along this run. Connection between separability and solving parity games was first observed in [BC18], where it was used to present algorithm of [CJK$^+$17] in the separability setting.

**Main results** In [CDF$^+$19] we have reformulated a bit connection between separability and solving parity games and shown the following key theorem.

**Theorem 10.** *If $G$ is a parity game with $n$ vertices and ranks up to $d$ and $\mathcal{A}$ is a deterministic safety automaton such that $L(\mathcal{A})$ separates* $\mathrm{EvenCycles}_{n,d}$ *and* $\mathrm{OddCycles}_{n,d}$ *then one can solve $G$ in time $\mathcal{O}(|G| \cdot |\mathcal{A}|)$.*

In fact in [CDF$^+$19] we formulate it (as Proposition 3.2) a bit differently, but together with an easy observation that safety games can be solved in linear time it gives Theorem 10. Proof of Theorem 10 is actually quite simple, the main contribution is to observe that it actually holds.

First main contribution of our paper is to show that based on papers [JL17, Leh18] in both cases one can construct deterministic safety automata of quasi-polynomial number of states, which not only separate $\mathrm{EvenCycles}_{n,d}$ and $\mathrm{OddCycles}_{n,d}$, but they even separate $\mathrm{EvenCycles}_{n,d}$ and $\mathrm{LimsupOdd}_d$. Moreover it holds also for the approach of [CJK$^+$17]. Therefore it seems like designing faster algorithms for parity games boils down to finding even smaller deterministic safety automata, which separate $\mathrm{EvenCycles}_{n,d}$ and $\mathrm{OddCycles}_{n,d}$ (or $\mathrm{LimsupOdd}_d$, as in all approaches till now). Here comes our second main contribution.

**Theorem 11.** *Every deterministic safety automaton $\mathcal{A}$ such that $L(\mathcal{A})$ separates* $\mathrm{EvenCycles}_{n,d}$ *and* $\mathrm{LimsupOdd}_d$ *has at least $\binom{\lfloor \lg n \rfloor + d/2 - 1}{\lfloor \lg n \rfloor}$ states, which is at least $n^{\lg(d/\lg n) - 2}$.*

Theorem 11 tells us that in order to design algorithms solving parity games, which are faster than quasi-polynomial we need to use some different technique. It is still possible that separability approach is a plausible direction, however in that case we need to construct automata such that $L(\mathcal{A})$ separate $\mathrm{EvenCycles}_{n,d}$ and $\mathrm{OddCycles}_{n,d}$, but do not separate $\mathrm{EvenCycles}_{n,d}$ and $\mathrm{LimsupOdd}_d$. In my opinion we should treat Theorem 11 as a hint where to look for faster algorithms rather that as a barrier that getting faster is very hard.

**Proof techniques** Proof of Theorem 11 relies on a notion of *universal trees*. We say that tree $t$ *contains* tree $t'$ if there is a mapping $h$ from nodes of tree $t'$ into nodes of tree $t$ such that for every node $n$ of $t'$ the parent of $n$ is mapped into parent of $h(n)$ and additionally root of $t'$ is mapped into root of $t$. We say that a tree is $(\ell, h)$*-universal* if it contains every tree of depth $h$ (longest path from root to leaf has $h$ edges) with $\ell$ leaves. For example a full binary tree of depth 2 is $(2, 2)$-universal, but there is also a smaller $(2, 2)$-universal tree constructed as follows. Root has two children: one of them has two children, while another one only one child. This tree has 3 leaves, less than 4 from the previous example. We show the following theorem, which is useful in the proof of Theorem 11.

**Theorem 12.** *For all $\ell, h > 0$ every $(\ell, h)$-universal tree has at least $\binom{\lfloor \lg \ell \rfloor + h - 1}{\lfloor \lg \ell \rfloor}$ leaves, which is at least $\ell^{\lg(h/\lg \ell) - 1}$ provided that $2h \leq \ell$.*

The other part of the proof of Theorem 11 shows that in every deterministic safety automaton $\mathcal{A}$ such that $L(\mathcal{A})$ separates $\mathrm{EvenCycles}_{n,d}$ and $\mathrm{LimsupOdd}_d$ one can introduce certain tree structure on states of $\mathcal{A}$. Moreover we show that this tree has to necessarily be $(n, d/2)$-universal, which finishes the argument.

### 4.3.6 Reachability problem in VASes

Already in Subsection 4.3.4 we described the reachability problem for VASes. As mentioned before reachability problem was shown to be decidable by Mayr in 1981 [May81]. However, algorithm was very complicated and no complexity upper bound was delivered. Later, Kosaraju in 1982 [Kos82] and Lambert in 1992 [Lam92] have shown another algorithms, however still based on the same approach, now called KLM decomposition (from surnames of three authors). The first complexity upper bound was provided recently [LS15], but the bound was extremely big cubic Ackermann function. Current best upper bound for its complexity is Ackermann function, which is shown in a very recent paper [LS19], to be published on LICS 2019. The best lower bound, till recently, was ExpSpace-hardness shown in 1976 by Lipton [Lip76]. Despite many efforts of settling complexity of reachability problem not much progress was reported on this topic till our recent paper [CLL+18]. Its main result can be summarized by the following statement.

**Theorem 13.** *VAS reachability problem is not elementary.*

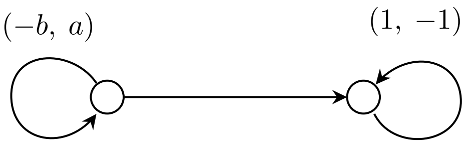
Recall that it means that VASS reachability problem cannot be solved in $n$-ExpTime for any $n \in \mathbb{N}$. Such a property we also call Tower-hardness.

**Publication** I intend to include this result in my main achievement, as it is definitely my best publication. It is accepted to STOC 2019 conference, which will take place in June 2019. However it does not yet (end of April) have a DOI number. Full text is available on Arxiv `https://arxiv.org/abs/1809.07115`. I hope that including this publication is possible. If this is impossible due to formal requirements please treat this part of the description as a text about paper, which is not included in the main achievement.

**Proof techniques** Below I will sketch some basic ideas of our construction, however many details have to be omitted. Let us denote by $!^n$ the $n^{\text{th}}$ iterate of factorial, so $a!^n = a\overbrace{!\cdots!}^{n}$. The proof is, roughly speaking, the reduction from the following problem Tower-hard: given counter automaton of $n$ states with zero tests, decide whether it has an accepting run, where all the counters are all the time bounded by $3!^n$. In [CLL+18] we formulate it a bit differently, using counter programs, however for the purpose of this short description it is easier to talk about counter automata.

The idea of reducing from this particular problem is not an important component of the solution, it is just a way of formulating the key insight. The main idea lies in the construction of VASSes with long runs. Or actually speaking more precisely, the idea lies in the construction of VASS-based implementation of counters with values bounded by some huge bound (namely $3!^n$) such that we still can test them for zero. It is easy to implement such a counter bounded by some small value, say value 3. We initialize counter $x = 0$ and its *friend* counter $\hat{x} = 3$, we all the time keep an invariant $x + \hat{x} = 3$. In that way always $0 \leq x \leq 3$. Moreover if we want to test $x$ for 0 we do the following: increment $x$ by 3 (while decrementing $\hat{x}$ by 3) and then decrement $x$ by 3 (while incrementing $\hat{x}$ by 3). It is easy to observe that this succeeds if and only if $x = 0$. The key insight is how to create a $k!$-bounded counter, which we can test for zero while having only $k$-bounded counter, which we can test for zero.

Before understanding the construction observe that VASS can easily implement multiplication by some fraction $\frac{a}{b}$, however this multiplication is *weak* in the following sense: if counter value before the operation was $x$ that after the operation it is in between $x$ and $\frac{a}{b} \cdot x$. In order to implement the weak multiplication consider the following VASS. Starting in the left state with counter values $(bn, 0)$ by applying the loop we can reach values $(0, an)$. Then we go to the right state and by applying the loop there we can reach values $(an, 0)$. So the total effect of this operation is multiplication of the first counter by $\frac{a}{b}$ (and change of state). However, of course, we also can reach smaller values, in particular if we do not apply any loop we reach values $(bn, 0)$ at the right state. Another property of this gadget is the following: if we start in $(bn + r, 0)$ for some $r < b$ then maximal value, which we can reach in the right state is $(an + r, 0)$. In that case $\frac{an+r}{bn+r} < \frac{a}{b}$, we can multiply exactly by $\frac{a}{b}$ only if counter value at the beginning is divisible by $b$. The main obstacle in creating our construction is the fact that we cannot multiply exactly, but only weakly.



The way we overcome this obstacle is the following. Our construction is based on the following simple equation:

$$\frac{2}{1} \cdot \frac{3}{2} \cdot \ldots \cdot \frac{k}{k-1} = \frac{k}{1}.$$

We construct a gadget in which we start from some counter value $x$, multiply it weakly by $\frac{2}{1}$, then multiply it weakly by $\frac{3}{2}$, etc., and at the end multiply weakly it by $\frac{k}{k-1}$. At the very end we demand that current counter value is exactly $kx$ (we kept $x$ on some additional counter). Notice that this is a maximal counter value, which can be reached and moreover this value is reached if and only if all the multiplications were precise. Our construction is not as simple, this is just a small part of it. During the process of multiplying $x$ by fractions of the form $\frac{i+1}{i}$ we also modify other counters. In particular we multiply some other counter $y$ weakly by $i + 1$, but we are able to show that this multiplication actually needs to be exact in every correct VASS run. Therefore in every correct run of our gadget counter $y$ is multiplied by $2 \cdot 3 \cdot \ldots \cdot k = k!$. This means intuitively that if we can check whether $x$ counter has grown exactly $k$ times than we can enforce counter $y$ to grow exactly $k!$ times. In similar ways we construct a VASS gadget with properties, which can be very informally stated in the following lemma.

**Lemma 14.** *There exists a VASS gadget, which given the ability to zero-test counters bounded by value $k$ delivers an access to zero-testable counters bounded by value $k!$.*

Recall that we can easily construct a VASS, which informally speaking gives an access to zero-testable counters bounded by value 3 (as shown a few paragraphs back). By starting with that small VASS and composing it $n$ times with VASS gadgets delivered by Lemma 14 one obtains $3!^n$-bounded counter values, which are zero testable. In that way one can show that VASS reachability problem is indeed TOWER-hard.

## 5 Other scientific achievements

In this section I briefly describe scientific contributions obtained after writing the PhD thesis and not included in the previous section.

### 5.1 Other works on infinite state systems

1. In [CJKS13] we consider strong bisimulation relation between two quite simple subclasses of infinite state systems: BPA, which is infinite state system generated by grammars in Greibach normal form and BPP, which is infinite state system corresponding to communication free Petri nets. We show that checking bisimulation in that case is in EXPTIME.

2. In [CJ15] we consider a variant of bisimulation taking into account silent transitions: branching bisimulation for BPA. We improve the previous decidability result by Yuxi Fu [Fu13] and show that the problem is in NEXPTIME. Our technique is to show that every branching bisimulation has a bisimulation base and a candidate for a base can be verified to be a correct base in exponential time.

3. In [CCH$^+$16] and its journal version [CCH$^+$19] we consider one counter automata. Previously it was known that if there is a run in one counter automaton from initial configuration of the form $p(0)$ and a final configuration of the form $q(0)$ then there is one of length $\mathcal{O}(n^3)$, where $n$ is number of states of the automaton. We improve this bound to $\mathcal{O}(n^2)$ by use of sophisticated modifications of the existing runs.

4. Reachability problem for VASSes can be seen as the question whether language of a given VASS is empty. However much more subtle questions about VASS languages are often interesting. In [CHZ18] we show that for the class of VASS languages a whole family of problems is decidable, which exhibit certain *unboundedness* properties. In particular we can decide whether given VASS language is included in $w_1^* w_2^* \cdots w_n^*$ for some words $w_1, \ldots, w_n \in \Sigma^*$. We show it by application of KLM decomposition introduced by Kosaraju, Lambert and Mayr [Kos82, Lam92, May81].

### 5.2 Tree patterns

Tree pattern is a model of an XPath query used extensively in XML databases. Informally speaking it is a tree, whose aim is to describe a language of trees. Every node of such a tree is labelled either by a letter from a finite alphabet $\Sigma$ or by a star $*$, whose aim is to describe any letter. Moreover parent-child edges can be either normal or *long*, long ones describe relation between ancestor and descendant. Language $L(p)$ of a tree pattern $p$ is the set of all the trees, which fit to the pattern.

1. In [CMPP15] we analyze computational complexity of the following tree pattern inclusions problem: given two tree patterns $p_1$ and $p_2$, is language $L(p_1)$ of $p_1$ included in the language $L(p_2)$ of $p_2$. We have considered many subcases in which some of the constructions: stars, long edges or normal edges are not allowed in some of the patterns. We have considered also different variants of the problem. In most of about a thousand of cases we gave the full answer. The interesting part of the paper were a few cases, in which the problem was really nontrivial.

2. In [CMNP16] and a bit different journal publications [CMNP17] and [CMNP18] we consider a long standing open problem (open since about a decade) about the minimal tree patterns. Tree pattern $t$ is

*minimal* if there is no other pattern $t'$ with smaller number of nodes for which $L(t) = L(t')$. Pattern $t$ is *redundant* if one can remove one of the leaves of $t$ to get a pattern with the same language. Of course if pattern is minimal it is not redundant. There was an open problem whether the opposite implication holds, i.e. whether every pattern, which is not redundant is also minimal and it seemed that it is indeed so. We provide a counterexample and based in this insight we show that the following *minimization problem* is $\Sigma_2^P$-complete: given tree pattern $p$ and a number $k \in \mathbb{N}$, is there a pattern $q$ of at most $k$ nodes such that $L(p) = L(q)$? This result can be also lifted to graph languages, as we discuss in [CMNP18].

### 5.3 Other

1. In [CGK15] we analyze sequences of vectors in $\mathbb{N}^d$ such that coordinates of vector on position $i + 1$ are obtained from coordinates of vector on position $i$ by one of the two operations: either 1) incrementing them by 1 or 2) reseting to 0. We provide an example of such a sequence of length doubly exponential in $d$ without any *dominating* pair, i.e vector $v_j$ on position $j$ being at least as big as vector $v_i$ on position $i$, for some $i < j$. Previous best example was only of exponential length.

2. In [CDLM13] and its journal version [CDLM17] we show that the problem whether given regular expression has the same language as some deterministic regular expression is PSPACE-complete.

3. In [CDMP16] and its journal version [CDMP18] we consider data trees with a set of restrictions. Every restriction has the following form: if a tree node satisfies some condition (not taking into account data values) then its data has to satisfy some positive boolean combination of equality conditions (data value is equal to another data value) and also some positive boolean combination of inequality conditions (data value is different than another data value). We show that the problem of checking whether in given regular language of trees there is a data tree satisfying all the restrictions is 2-EXPTIME-complete.

### References

[BC18] Mikołaj Bojańczyk and Wojciech Czerwiński. An automata toolbox, February 2018. `https://www.mimuw.edu.pl/~bojan/papers/toolbox-reduced-feb6.pdf`.

[BFG+15] Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is pspace-complete. In *Proceedings of LICS 2015*, pages 32–43, 2015.

[BFLZ10] Rémi Bonnet, Alain Finkel, Jérôme Leroux, and Marc Zeitoun. Place-boundedness for vector addition systems with one zero-test. In *Proceedings of FSTTCS 2010*, pages 192–203, 2010.

[CCH+16] Dmitry Chistikov, Wojciech Czerwinski, Piotr Hofman, Michal Pilipczuk, and Michael Wehar. Shortest paths in one-counter systems. In *Proceedings of FOSSACS 2016*, pages 462–478, 2016.

[CCH+19] Dmitry Chistikov, Wojciech Czerwinski, Piotr Hofman, Michal Pilipczuk, and Michael Wehar. Shortest paths in one-counter systems. *Logical Methods in Computer Science*, 15(1), 2019.

[CCLP17a] Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Regular separability of Parikh automata. In *Proceedings of ICALP '17*, pages 117:1–117:13, 2017.

[CCLP17b] Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Separability of reachability sets of vector addition systems. In *Proceedings of STACS '17*, pages 24:1–24:14, 2017.

[CDF+19] Wojciech Czerwinski, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdzinski, Ranko Lazic, and Pawel Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In *Proceedings of SODA 2019*, pages 2333–2349, 2019.

[CDLM13] Wojciech Czerwinski, Claire David, Katja Losemann, and Wim Martens. Deciding definability by deterministic regular expressions. In *Proceedings of FOSSACS 2013*, pages 289–304, 2013.

[CDLM17] Wojciech Czerwinski, Claire David, Katja Losemann, and Wim Martens. Deciding definability by deterministic regular expressions. *J. Comput. Syst. Sci.*, 88:75–89, 2017.

[CDMP16] Wojciech Czerwinski, Claire David, Filip Murlak, and Pawel Parys. Reasoning about integrity constraints for tree-structured data. In *Proceedings of ICDT 2016*, pages 20:1–20:18, 2016.

[CDMP18] Wojciech Czerwinski, Claire David, Filip Murlak, and Pawel Parys. Reasoning about integrity constraints for tree-structured data. *Theory Comput. Syst.*, 62(4):941–976, 2018.

[CGK15] Wojciech Czerwinski, Tomasz Gogacz, and Eryk Kopczynski. Non-dominating sequences of vectors using only resets and increments. *Fundam. Inform.*, 140(2):123–127, 2015.

[CHZ18] Wojciech Czerwinski, Piotr Hofman, and Georg Zetzsche. Unboundedness problems for languages of vector addition systems. In *Proceedings of ICALP 2018*, pages 119:1–119:15, 2018.

[CJ15] Wojciech Czerwinski and Petr Jancar. Branching bisimilarity of normed BPA processes is in NEXPTIME. In *Proceedings of LICS 2015*, pages 168–179, 2015.

[CJK$^+$17] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of STOC 2017*, pages 252–263, 2017.

[CJKS13] Wojciech Czerwinski, Petr Jancar, Martin Kot, and Zdenek Sawa. Complexity of checking bisimilarity between sequential and parallel processes. In *Proceedings of MFCS 2013*, pages 302–313, 2013.

[CL17] Wojciech Czerwiński and Sławomir Lasota. Regular separability of one counter automata. In *Proceedings of LICS '17*, pages 1–12, 2017.

[CLL$^+$18] Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for petri nets is not elementary (extended abstract). *CoRR*, abs/1809.07115, 2018.

[CLM$^+$18] 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 '18*, pages 35:1–35:18, 2018.

[CMM13] Wojciech Czerwinski, Wim Martens, and Tomás Masopust. Efficient separability of regular languages by subsequences and suffixes. In *Proceedings of ICALP 2013*, pages 150–161, 2013.

[CMNP16] Wojciech Czerwinski, Wim Martens, Matthias Niewerth, and Pawel Parys. Minimization of tree pattern queries. In *Proceedings of PODS 2016*, pages 43–54, 2016.

[CMNP17] Wojciech Czerwinski, Wim Martens, Matthias Niewerth, and Pawel Parys. Optimizing tree patterns for querying graph- and tree-structured data. *SIGMOD Record*, 46(1):15–22, 2017.

[CMNP18] Wojciech Czerwinski, Wim Martens, Matthias Niewerth, and Pawel Parys. Minimization of tree patterns. *J. ACM*, 65(4):26:1–26:46, 2018.

[CMPP15] Wojciech Czerwinski, Wim Martens, Pawel Parys, and Marcin Przybylko. The (almost) complete guide to tree pattern containment. In *Proceedings of PODS 2015*, pages 117–130, 2015.

[CMvR$^+$17] 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.

[CMvRZ15] Wojciech Czerwinski, Wim Martens, Lorijn van Rooijen, and Marc Zeitoun. A note on decidable separability by piecewise testable languages. In *Proceedings of FCT 2015*, pages 173–185, 2015.

[Fu13] Yuxi Fu. Checking equality and regularity for normed BPA with silent moves. In *Proceedings of ICALP 2013*, pages 238–249, 2013.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Mathematical Society*, 3((2)):326–336, 1952.

[HKO16] Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In *Proceedings of POPL 2016*, pages 151–163, 2016.

[HMW10] Peter Habermehl, Roland Meyer, and Harro Wimmel. The downward-closure of petri net languages. In *Proceedings of ICALP 2010*, pages 466–477, 2010.

[Hun82] Harry B. Hunt. On the decidability of grammar problems. *Journal of the ACM*, 29(2):429–447, 1982.

[JL17] Marcin Jurdzinski and Ranko Lazic. Succinct progress measures for solving parity games. In *Proceedings of LICS 2017*, pages 1–9, 2017.

[JPZ08] Marcin Jurdzinski, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. Comput.*, 38(4):1519–1532, 2008.

[Kop16] Eryk Kopczynski. Invisible pushdown languages. In *Proceedings of LICS '16*, pages 867–872, 2016.

[Kos82] S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proceedings of STOC '82*, pages 267–281, 1982.

[Lam92] Jean-Luc Lambert. A structure to decide reachability in Petri nets. *Theor. Comput. Sci.*, 99(1):79–104, 1992.

[Leh18] Karoliina Lehtinen. A modal $\mu$ perspective on solving parity games in quasi-polynomial time. In *Proceedings of LICS 2018*, pages 639–648, 2018.

[Ler13] Jérôme Leroux. Presburger vector addition systems. In *Proceedings of LICS 2013*, pages 23–32, 2013.

[Lip76] Richard J. Lipton. The reachability problem requires exponential space. Technical report, Yale University, 1976.

[LS15] Jérôme Leroux and Sylvain Schmitz. Demystifying reachability in vector addition systems. In *Proceedings of LICS'15*, pages 56–67, 2015.

[LS19] Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. *CoRR*, abs/1903.08575, 2019.

[May81] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of STOC'81*, pages 238–246, 1981.

[PZ14a] Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words.

In *Proceedings of ICALP 2014*, pages 342–353, 2014.

[PZ14b] Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. In *Proceedings of LICS 2014*, pages 75:1–75:10, 2014.

[PZ17] Thomas Place and Marc Zeitoun. Separation for dot-depth two. In *Proceedings of LICS '17*, pages 1–12, 2017.

[Sch65] Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

[Sim75] Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages, 2nd GI Conference, 1975*, pages 214–222, 1975.

[SW76] Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2), 1976.

Wojciech Czerwiński