Autoreferat

1 Imię i nazwisko

Wojciech Karol Czerwiński

2 Posiadane stopnie naukowe

2009–2013 studia doktoranckie na kierunku informatyka, Uniwersytet Warszawski
21.03.2013 stopień doktora nauk matematycznych w zakresie informatyki, Uniwersytet Warszawski
tytuł rozprawy: *Częściowo przemienne grafy bezkontekstowe*
promotor: prof. dr hab. Sławomir Lasota

2002–2008 studia magisterskie na kierunku matematyka, Uniwersytet Warszawski
26.09.2009 stopień magistra matematyki, Uniwersytet Warszawski
tytuł pracy: *Nierówności Chinczyna*
promotor: prof. dr hab. Rafał Latała

2002–2007 studia magisterskie na kierunku informatyka, Uniwersytet Warszawski
27.09.2007 stopień magistra informatyki, Uniwersytet Warszawski
tytuł pracy: *Symulacja pomiędzy grami*
promotor: prof. dr hab. Damian Niwiński

3 Dotychczasowe zatrudnienie w jednostkach naukowych

3.1 Obecne zatrudnienie

od 10.2013 adiunkt w Instytucie Informatyki Uniwersytetu Warszawskiego

3.2 Poprzednie zatrudnienie

07.2012 – 06.2013 staż naukowy na Uniwersytecie w Bayreuth

4 Osiągnięcie naukowe

4.1 Tytuł osiągnięcia

Problemy separowalności i osiągalności w systemach nieskończenie stanowych.

4.2 Publikacje będące częścią osiągnięcia

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 Opis osiągnięcia

Sześć powyżej wspomnianych prac dotyczy tematu separowalności i osiągalności w systemach nieskończenie stanowych.

4.3.1 Wprowadzenie

Podstawowe modele obliczeń Badanie podstawowych modeli obliczeń jest ważną i aktywną dziedziną informatyki teoretycznej od lat 70-tych. Takie modele, zwane systemami nieskończenie-stanowymi, są zwykle nieskończonymi, skierowanymi grafami, których krawędzie są etykietowane literami z pewnego skończonego alfabetu. Systemy nieskończenie stanowe muszą być skończenie opisywalne, jako, że w przeciwnym wypadku nie jest możliwe zaprojektowanie żadnych algorytmów, które przyjmują takie systemy na wejściu. Najbardziej popularne, a jednocześnie najbardziej fundamentalne modele obliczeń to kilka rozszerzeń automatów skończonych: maszyny Turinga, automaty ze stosem i automaty z licznikami (gdzie zwiększenie i zmniejszenie licznika oraz czasami test na zero są dopuszczalne). O wiele więcej modeli było rozważanych przez dekady. Różnią się one posiadaną strukturą oraz operacjami, które mogą wykonywać. Większość z nich używa kilku podstawowych konstrukcji: stosu, licznika, danych z nieskończonej dziedziny oraz zegarów w wielu różnych kombinacjach, modyfikacjach, czy ograniczeniach. To dlatego tak ważne jest, by zrozumieć podstawowe pojęcia, bo one wpływają na wszystko inne.

Można zadań kilka fundamentalnych pytań dla różnych systemów nieskończenie stanowych. Prawdopodobnie najbardziej podstawowym jest problem osiągalności: dane dwa wierzchołki systemu, rozstrzygnij, czy istnieje ścieżka pomiędzy nimi. Ten problem nie bierze pod uwagę etykiet na krawędziach. Używając etykiet krawędzi możemy zdefiniować języki systemów nieskończenie stanowych. Dla danych dwóch wierzchołków u i v język $L(u, v)$ jest zdefiniowany w następujący naturalny sposób: jest to zbiór słów, które są etykietowaniami ścieżek pomiędzy u a v. Problem osiągalności można wtedy przeformułować jako pytanie, czy język $L(u, v)$, dla danych u, v, jest pusty. Można również zadać wiele więcej naturalnych pytań dotyczących języka $L(u, v)$, które były rozważane w literaturze: 1) czy $L(u, v)$ jest skończony?, 2) czy $L(u, v)$ jest uniwersalny?, tzn. czy zawiera wszystkie słowa, 3) czy $L(u, v)$ jest regularny?, 4) czy $L(u, v)$ należy to pewnej ustalonej rodziny języków $\mathcal{F}$, itd. Rozważanie tych pytań prowadzi często do lepszego zrozumienia rozważanych modeli obliczeń.

Problem separowalności W mojej pracy często rozważałem inny problem, który do niedawna był stosunkowo mało popularny, *problem separowalności*. Niech $K, L \subseteq \Sigma^*$ to dwa języki nad alfabetem Σ. Język $S \subseteq \Sigma^*$ *separuje* K *i* L jeśli S zawiera K i jest rozłączny z L. Dla dwóch rodzin $\mathcal{F}$, $\mathcal{G}$ języków nad alfabetem Σ *problem* $\mathcal{F}$*-separowalności dla* $\mathcal{G}$ pyta, czy dla dwóch danych języków $K, L \in \mathcal{G}$ istnieje język w $\mathcal{F}$, który separuje K i L. Mówimy po prostu *problem* $\mathcal{F}$*-separowalności* jeśli $\mathcal{G}$ jest jasne z kontekstu. Dla klas $\mathcal{G}$ zamkniętych na dopełnienie problem $\mathcal{F}$-separowalności jest uogólnieniem problemu $\mathcal{F}$*-należenia*, który pyta, czy dany język $L \in \mathcal{G}$ należy do $\mathcal{F}$. Istotnie, $L \in \mathcal{F}$ wtedy i tylko wtedy gdy L oraz $\bar{L}$, tzn. dopełnienie L, są separowane przez język z rodziny $\mathcal{F}$.

4.3.2 Separowalność dla języków regularnych

Problem należenia Problem $\mathcal{F}$-należenia dla języków regularnych jest ważnym pytaniem, które zyskało wiele uwagi dla różnych podklas $\mathcal{F}$ języków regularnych. Słynna praca Schutzenbergera [Sch65] charakteryzuje języki regularne definiowalne w logice pierwszego rzędu jako te, dla których monoid syntaktyczny nie zawiera nietrywialnej grupy. Ta charakteryzacja prowadzi wprost do rozstrzygalności problemu należenia. Inny słynny wynik [Sim75] charakteryzuje języki piecewise testable. Język regularny jest *piecewise testable* jeśli jest postaci $\Sigma^* a_1 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$, gdzie $a_1, \ldots, a_n \in \Sigma$ są literami. Twierdzenie Simona mówi, że język regularny jest piecewise testable wtedy i tylko wtedy, gdy jego monoid syntaktyczny jest $\mathcal{J}$-trywialny, co oznacza, że jego $\mathcal{J}$-klasy są singletonami. Ten ostatni warunek jest łatwo rozstrzygalny. Problem należenia był również

rozważany dla innych podklas języków regularnych i wiele przypadków jest rozwiązanych. Stosowane metody zazwyczaj używają technik algebraicznych zastosowanych do monoidów syntaktycznych, tak jak w dwóch opisanych wyżej przypadkach. Pomimo popularności problemu należenia do niedawna problem separowalności był rzadko rozważany w społeczności związanej z teorią automatów.

Piecewise-testable-separowalność W 2013 roku wspólnie z Wimem Martensem and Thomasem Masopustem rozważaliśmy problem z teorii baz danych, który doprowadził nas do wyniku na temat języków piecewise testable (PTL) [CMM13]. Ten artykuł jest częścią mojego głównego osiągnięcia. Główny jego wynik to charakteryzacja par języków K, L które nie mogą być separowane przez żaden język piecewise testable. Drugi główny wynik, który korzysta z charakteryzacji, to wielomianowy algorytm rozstrzygający problem PTL-separowalności dla języków regularnych.

Zanim przedstawimy te wyniki wprowadzimy kilka pojęć. Porządek $(X, \leq)$ jest *well quasi order* (wqo) jeśli nie zawiera ani nieskończonego ciągu zstępującego względem $\leq$ ani też nieskończonego antyłańcucha względem $\leq$. Rozważmy porządek podciągu na skończonych słowach, który jest istotny w wielu naszych rozważaniach. Słowo $u = a_1 \cdots a_n$ jest *podciągiem* v, co oznaczamy $u \preceq v$, jeśli $v \in \Sigma^* a_1 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$. Dobrze znamy lemat Higmana mówi, że porządek podciągu jest wqo [Hig52]. Powiemy, że zbiór S jest $\leq$-*domknięty* jeśli $x \in S$ oraz $x \leq y$ implikuje, że $y \in S$. Kluczowym pojęciem wprowadzonym w naszej pracy są *zygzaki*. Nieskończony ciąg słów $w_1, w_2, \ldots \in \Sigma^*$ jest *nieskończonym* $\leq$*-zygzakiem* pomiędzy językami $K, L \subseteq \Sigma^*$ jeśli $w_1 \in K \cup L$ oraz dla każdego $i \geq 1$ następujące warunki są spełnione

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

Innymi słowy ciąg słów w_i jest wstępujący względem $\leq$ oraz alternuje pomiędzy językami K i L.

Główny wynik [CMM13] to Twierdzenie 3, które mówi, że

Twierdzenie 1. *Dla języków K i L oraz wqo $\leq$ na słowach następujące warunki są równoważne*

1. *K i L są separowalne przez skończoną boolowską kombinację $\leq$-domkniętych języków*
2. *nie istnieje nieskończony $\leq$-zygzak pomiędzy K i L.*

W istocie Twierdzenie 1 twierdzi nieco więcej. Jest tam również trzeci równoważny warunek, ale nie jest on tak istotny jak dwa wspomniane powyżej. Tak jak wspomnieliśmy wcześniej porządek podciągu $\preceq$ jest wqo. Zatem Twierdzenie 1 implikuje, że PTL-separowalność dwóch języków jest równoważna nieistnieniu nieskończonego $\preceq$-zygzaka pomiędzy nimi.

Interesującą częścią Twierdzenia 1 jest fakt, że mówi ono o separowalności przez PTL, czyli podklasę języków regularnych, jednak nie używa technik algebraicznych. W szczególności języki K i L nie muszą być regularne. Według mojej wiedzy to był pierwszy opublikowany wynik tego typu unikający rozważania monoidów syntaktycznych.

Drugi główny wynik [CMM13] to algorytm wielomianowy rozstrzygający, czy istnieje nieskończony $\preceq$-zygzak pomiędzy dwoma danymi językami regularnymi. W szczególności daje on natychmiast algorytm wielomianowy rozstrzygający PTL-separowalność dla języków regularnych.

W [CMM13] rozważaliśmy również problem separowalności przez rodziny inne niż PTL, opierające się nie na porządku podciągu, ale na porządkach prefiksu i sufiksu.

W ciągu ostatnich kilku lat temat separowalności języków regularnych przez ich podklasy został bardzo rozwinięty, głównie przez Thomasa Place'a i Marca Zeitouna. Pokazali oni w szczególności, że dla danych dwóch języków regularnych rozstrzygalne jest, czy są one separowalne przez języki definiowalne w logice pierwszego rzędu [PZ14b], przez języki na drugim poziomie hierarchii dot depth [PZ17] i wiele innych. Okazało się również, że rozważanie problemu separowalności istotnie pomaga w rozstrzyganiu problemu należenia dla innych podklas [PZ14a]. Ten fakt wskazuje, że problem separowalności jest naturalnym problemem, głęboko związanym z wyrażalnością danej klasy języków.

4.3.3 Separowalność poza językami regularnymi

Obserwacja, że pojęcie zygzaka może być również stosowane do języków nie będących regularnymi zainspirowała nas to rozważenia takich języków jako wejścia do problemu.

Negatywne wyniki Już w latach 70-tych i 80-tych istniały pewne negatywne wyniki na temat separowalności, które wydawały się sugerować że otrzymanie jakiejkolwiek rozstrzygalności dla naturalnego problemu separacji języków nieregularnych będzie trudne lub niemożliwe. Już w 1976 Szymanski i Williams [SW76] pokazali, że regularna separacja jest nierozstrzygalna dla języków bezkontekstowych. Parę lat później Hunt wzmocnił ten wynik [Hun82]. Język $L \subseteq \Sigma^*$ jest *definite* jeśli istnieje liczba $k \in \mathbb{N}$ taka, że należenie do L zależy jedynie od prefiksu długości k. Innymi słowy jeśli $u, v \in \Sigma^*$, mają te same prefiksy długości k, to wtedy $u \in L \iff v \in L$. Hunt pokazał, że dla dowolnej rodziny języków $\mathcal{F}$, która zawiera wszystkie języki definite problem $\mathcal{F}$-separowalności jest nierozstrzygalny dla języków bezkontekstowych. Zawieranie wszystkich języków definite to bardzo słabe wymaganie. Nie tylko języki regularne spełniają je, ale również na przykład języki definiowalne w FO (tj. logice pierwszego rzędu), języki definiowalne w FO przy użyciu dwóch zmiennych i wiele innych. Ogólnie rzecz biorąc aby język był definite wystarczy, by był definiowalny w pewnej logice, która może przetestować jaka jest k-ta litera słowa, dla dowolnego ustalonego $k \in \mathbb{N}$. Języki piecewise testable nie są definite, nie można w nich nawet sprawdzić pierwszej litery słowa, przykładowo język $a\Sigma^*$ nie jest piecewise testable. Dlatego rozstrzygalność PTL-separowalności dla języków bezkontekstowych (i innych klas języków) nie jest wykluczona.

Wspólne wzorce W pracy [CMvR$^+$17], która jest drugą częścią mojego głównego osiągnięcia rozważamy PTL-separowalność dla języków, które nie są regularne. Wersja konferencyjna tego artykułu to [CMvRZ15]. Całe podejście oparte jest na Twierdzeniu 2.1 z naszej pracy, które łączy PTL-separowalność ze *wzorcami*. Pojęcie wzorca wydaje się być właściwsze dla naszych badań niż zygzaki. *Wzorzec* to para $(\vec{u}, \vec{B})$, gdzie $u_0, \ldots, u_p \in \Sigma^*$ są słowami, $B_1, \ldots, B_p \subseteq \Sigma$ są podzbiorami alfabetu Σ, $\vec{u} = (u_0, \ldots, u_p)$ i $\vec{B} = (B_1, \ldots, B_p)$. *Alfabet* słowa $w \in \Sigma$ to zbiór wszystkich liter z Σ, które występują w w. Oznaczamy go $\mathrm{Alph}(w)$. Dla $B \subseteq \Sigma$ przez $B^{\otimes}$ oznaczamy zbiór wszystkich słów, których alfabet to dokładnie B, $B^{\otimes} = \{w \mid \mathrm{Alph}(w) = B\}$. Dla wzorca $(\vec{u}, \vec{B})$ definiujemy języki

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

Język L *zawiera wzorzec* $(\vec{u}, \vec{B})$ jeśli istnieje nieskończony ciąg słów $w_1, w_2, \ldots \in L$ takie, że dla każdego $n \in \mathbb{N}$ zachodzi $w_n \in L(\vec{u}, \vec{B}, n)$. Twierdzenie 2.1 w [CMvR$^+$17] stwierdza następujący fakt.

Twierdzenie 2. *Dwa języki słów K i L nie są PTL-separowalne wtedy i tylko wtedy, gdy zawierają wspólny wzorzec.*

Separowalność języków bezkontekstowych Przyjrzyjmy się jak możemy użyć Twierdzenia 2 do pokazanie PTL-separowalności języków bezkontekstowych. Niech K i L będą dwoma językami bezkontekstowymi. Rozważmy dwie semi-procedury, jedną (*pozytywną*), która szuka świadectwa na to, że K i L są PTL-separowalne, i drugą (*negatywną*), która szuka świadectwa na to, że K i L nie są PTL-separowalne. Zaprojektowanie pozytywnej semi-procedury jest łatwe: po prostu przeglądamy wszystkie możliwe języki S w PTL i dla każdego z nich sprawdzamy, czy zawiera K oraz czy jest rozłączny z L. Sprawdzanie inkluzji oraz rozłączności to przypadki szczególne sprawdzania pustości przecięcia języka regularnego i bezkontekstowego. A zatem jest to łatwo rozstrzygalne. Jeśli K i L są PTL-separowalne, to znajdziemy w pewnym momencie język S świadczący o tym.

W procedurze negatywnej użyjemy Twierdzenia 2. Przeglądamy wszystkie wzorce, dla każdego z nich sprawdzając, czy jest zawarty zarówno w K jak i w L. Jeśli K i L nie są separowalne, to znajdziemy w pewnym momencie wzorzec, który o tym świadczy. A zatem problem sprowadza się do sprawdzenia, czy dla danego wzorca s i danego języka bezkontekstowego L język L zawiera s. Rozwiązujemy ten problem w szczegółach w [CMvR$^+$17], nawet dla ogólniejszego przypadku, który przedyskutujemy za moment. Wysokopoziomowa idea stojąca za rozwiązaniem jest następująca. Wzorzec $s = (\vec{u}, \vec{B})$ z $\vec{u} = (u_0, \ldots, u_p)$ i $\vec{B} = (B_1, \ldots, B_p)$ jest zawarty w L z grubsza rzecz biorąc jeśli istnieją słowa w L, które zawierają najpierw dowolnie wiele kopii B_1, potem dowolnie wiele kopii B_2, itd., a na końcu dowolnie wiele kopii B_p. Dodatkowo pomiędzy tymi kopiami powinny być słowa u_i. Jednakże słowa u_i oraz konkretna postać B_i to szczegóły techniczne. Dla języka L i wzorca s można skonstruować inny język $\hat{L} \subseteq a_1^* a_2^* \cdots a_p^*$ taki, że L zawiera s wtedy i tylko wtedy, gdy $\hat{L}$ zawiera dla każdego n słowa postaci $a_1^{n_1} a_2^{n_2} \cdots a_p^{n_p}$ gdzie wszystkie $n_i \geq n$. Intuicyjnie w $\hat{L}$ słowa u_i są pominięte, a litery a_i pełnią rolę alfabetów B_i. Ten fenomen zainspirował nas do zdefiniowania następującego *problemu przekątniowego*: dany język $L \subseteq \{a_1, \ldots, a_k\}$, rozstrzygnij, czy dla każdego $n \in \mathbb{N}$

obraz Parikha języka L zawiera tuple $(n_1, \ldots, n_k)$ takie, że wszystkie $n_i \geq n$. Przypomnijmy, że *obraz Parikha* języka L to zbiór tupli $(n_1, \ldots, n_k) \in \mathbb{N}^k$ takich, że istnieje słowo w L, które zawiera dokładnie n_1 wystąpień litery a_1, dokładnie n_2 wystąpień litery a_2 itd. i dokładnie n_k wystąpień litery a_k. Problem przekątniowy jest rozstrzygalny dla języków bezkontekstowych, co wspólnie z pomysłami opisanymi wyżej daje rozstrzygalność PTL-separowalności dla języków bezkontekstowych.

Uogólnienie Można nietrudno zauważyć, że powyższa idea dowodu da się uogólnić do dowolnej rodziny języków $\mathcal{F}$, która spełnia następujące warunki:

1. sprawdzenie, czy dany język regularny i język z $\mathcal{F}$ mają niepuste przecięcie jest rozstrzygalne,
2. język $\hat{L}$, o własnościach jak powyżej, jest obliczalny z L,
3. problem przekątniowy jest rozstrzygalny.

Okazuje się, że wszystkie trzy warunki są spełniane dla rodzin języków, które są *full trio*. Jedna z definicji jest taka, że rodzina $\mathcal{F}$ jest *full trio* jeśli jest efektywnie zamknięta na przekształcenia zwane *rational transduction*, czyli na obrazy pewnego rodzaju transducerów. Nie będę przedstawiał tutaj szczegółów, wystarczy wiedzieć, że dla full trio obliczalne są wszystkie operacje o regularnym posmaku, jak przecięcie z regularnym językiem, usunięcie liter z pewnego podalfabetu itd. W [CMvR$^+$17] pokazaliśmy, że dla wszystkich rodzin full trio, dla których problem przekątniowy jest rozstrzygalny problem PTL-separowalności jest również rozstrzygalny.

Równoważność Bardzo interesującym faktem jest to, że przeciwna implikacja jest również prawdziwa: dla każdego full trio, dla którego problem PTL-separowalności jest rozstrzygalny problem przekątniowy też jest rozstrzygalny. A więc dla rodzin full trio rozstrzygalność PTL-separowalności i rozstrzygalność problemu przekątniowego są równoważne. Aby w pełni zaprezentować główny wynik [CMvR$^+$17] potrzebujemy wprowadzić parę pojęć.

Dla języka L jego *domnięcie w dół*, oznaczane $L{\downarrow}$, to zbiór wszystkich podciągów słów z L, $L{\downarrow} = \{u \mid \exists_{v \in L} u \preceq v\}$. *Problem wspólnego nieograniczenia* (SUP, od ang. simultaneous unbounded problem) pyta, czy dany $L \subseteq a_1^* a_2^* \cdots a_n^*$ spełnia $L{\downarrow} = a_1^* a_2^* \cdots a_n^*$. Zauważmy, że SUP jest podobny do problemu przekątniowego, z tą jedyną różnicą, że wejście do SUP spełnia specjalne założenie, mianowicie litery są „posortowane", tj. na początku występują a_1, potem a_2, itd. Jesteśmy gotowi do sformułowania głównego wyniku (Theorem 2.5).

Twierdzenie 3. *Dla rodziny języków $\mathcal{F}$, która jest full trio, następujące własności są równoważne:*

1. *PTL-separowalność jest rozstrzygalna dla $\mathcal{F}$,*
2. *problem przekątniowy jest rozstrzygalny dla $\mathcal{F}$,*
3. *SUP jest rozstrzygalny dla $\mathcal{F}$,*
4. *domknięcia w dół języków z $\mathcal{F}$ są obliczalne.*

Rozstrzygalne klasy Jako wniosek z Twierdzenia 3 wiele rodzin języków ma rozstrzygalny problem PTL-separowalności, co zostało wspomniane w [CMvR$^+$17]. Dowodzimy tam wprost, że problem przekątniowy jest rozstrzygalny dla języków bezkontekstowych, co czyni języki bezkontekstowe pierwszą klasą poza językami regularnymi, która ma rozstrzygalny problem PTL-separowalności. Jednakże znane są wyniki dla innych rodzin języków, które są wystarczające aby dostać dla nich również rozstrzygalność. W [HKO16] pokazano, że domknięcia w dół języków automatów ze stosem wyższego rzędu są obliczalne, co daje rozstrzygalność PTL-separowalności dla tej klasy. Dowodzimy także w [CMvR$^+$17], że problem przekątniowy jest rozstrzygalny dla języków systemów dodawania wektorów ze stanami (ang. VASS od vector addition systems with states). Dowód jest redukcją z problemu ograniczoności w stanie dla VASSów z jednym testem na zero, który jest rozstrzygalny, jak pokazano w [BFLZ10]. Inny sposób pokazania rozstrzygalności PTL-separowalności języków VASSów to użycie pracy [HMW10], gdzie udowodniona została obliczalność domknięć w dół dla języków VASSów. Ostatnim wynikiem tego typu jest fakt, że jeśli PTL-separowalność jest rozstrzygalna jest rodzin języków $\mathcal{F}_1$ i $\mathcal{F}_2$, to wtedy jest ona rozstrzygalna również dla sumy $\mathcal{F}_1 \cup \mathcal{F}_2$. Innymi słowy możemy na przykład rozstrzygnąć, czy dany język VASSa i dany język bezkontekstowy są separowalne przez język PTL.

4.3.4 Regularna separowalność

Wyniki [CMvR$^+$17] pokazały, że wiele wyników nt. rozstrzygalności separowalności jest możliwych poza językami regularnymi. W [CMvR$^+$17] rozważamy PTL-separowalność, jednak rodzina PTL nie jest najbardziej naturalną klasą separatorów, która może być rozważana dla wejść będących językami nieregularnymi. Najbardziej naturalną klasą separatorów w tej sytuacji są same języki regularne. Z tego powodu wiele moich późniejszych prac było (i wciąż jest) na temat regularnej separowalności. Mam przekonanie, że problem regularnej separowalności jest kolejnym bardzo naturalnym problemem i rozważanie go może dać głębsze zrozumienie pewnych podstawowych systemów nieskończenie stanowych, w szczególności VASSów.

Przeszkody Tak jak wspomnieliśmy powyżej już w latach 70-tych pokazano w [SW76] że regularna separowalność jest nierozstrzygalna dla języków bezkontekstowych. Ostatnio Kopczyński wzmocnił ten rezultat [Kop16] pokazując, że regularna separowalność jest nierozstrzygalna nawet dla języków visibly automatów ze stosem (które są determinizowalne). Jasny przekaz tych wyników jest taki, że nie można mieć nadziei na rozstrzygalność regularnej separowalności w obecności stosu. Intuicja stojąca za problemem separacji jest taka, że przypomina on nieco problem pustości przecięcia dwóch języków. Te powody skierowały nas w stronę rozważania automatów z licznikami.

VASy Przypomnijmy, że d-wymiarowy system dodawania wektorów (VAS - ang. vector addition system) to zbiór tranzycji $T \subseteq \mathbb{Z}^d$ razem z ich etykietowaniem $\ell : T \to \Sigma \cup \{\varepsilon\}$ przez litery z pewnego skończonego alfabetu Σ. Tranzycja $v \in T$ może być użyta w konfiguracji $u \in \mathbb{N}^d$ pod warunkiem, że $u + v \in \mathbb{N}^d$. Dla danych konfiguracji *początkowej* s i *końcowej* t ze zbioru $\mathbb{N}^d$ można naturalnie zdefiniować język VASa jako zbiór wszystkich etykietowań ścieżek prowadzących z początku do końca. Taki język nazywamy *językiem VASa*. Bardzo podobne modele, tj. rozszerzenie VASów o stany (VASSy - ang. vector addition systems with states) oraz sieci Petriego były również bardzo szeroko rozważane w literaturze. W szczególności łatwo jest pokazać, że rodziny języków VASów, VASSów oraz sieci Petriego są równe.

VASy są fundamentalnym modelem obliczeń rozważanym od lat 70-tych. Wiele jest literatury badającej różne ich aspekty. Prawdopodobnie najbardziej podstawowym problemem jest *problem osiągalności*, który pyta, czy jest ścieżka z ustalonego początku do ustalonego końca. Problem ten może być również interpretowany jako pytanie, czy język VASa jest pusty. Problem osiągalności jest rozstrzygalny, jak pokazał Mayr w roku 1981 [May81]. Powiemy później więcej na temat jego złożoności obliczeniowej, jako, że najlepsze aktualnie ograniczenie dolne złożoności, TOWER-trudność, jest częścią mojego głównego osiągnięcia [CLL$^+$18]. Główny wniosek z powyższych rozważań to fakt, że pustość przecięcia dwóch języków VASów jest rozstrzygalna, gdyż może być zredukowana do pustości języka produktu dwóch rozważanych VASów, czyli w istocie do problemu rozstrzygalności. Ten fakt i inne argumenty doprowadziły nas do postawienia następującej hipotezy.

Hipoteza 4. *Problem regularnej separowalności jest rozstrzygalny dla języków VASów.*

Ta hipoteza leżała u podstaw badań prowadzących do kilku wyników, które niebawem omówię.

Zbiory osiągalności VASSów Pierwsza praca w kierunku rozwiązania Hipoteza 4 właściwie skupiła się nie na językach VASSów, tylko bardziej na ich zbiorach osiągalności [CCLP17b]. Zbiór $S \subseteq \mathbb{N}^d$ jest *zbiorem osiągalności VASSa* jeśli istnieje pewien VASS i jego konfiguracja początkowa taka, że zbiór konfiguracji osiągalnych z konfiguracji początkowej o pewnym ustalonym stanie to dokładnie zbiór S. Nasz wynik może być również przedstawiony w terminach języków, tak jak we Wniosku 6, ale bardziej naturalne jest jego przedstawienie w terminach zbiorów osiągalności.

Dla $u \in \mathbb{N}^d$ oraz $i \in \{1, \ldots, d\}$ niech $u[i]$ będzie i-tą współrzędną wektora u. Dla $n \in \mathbb{N}$ powiemy, ze $u \equiv_n v$ jeśli dla każdych $i \in \{1, \ldots, d\}$ zachodzi $u[i] \equiv v[i] \mod n$. Zbiór $S \subseteq \mathbb{N}^d$ jest *modularny* jeśli istnieje liczba $n \in \mathbb{N}$ taka, że należenie do S zależy jedynie od wartości liczników modulo n. Innymi słowy jeśli $u, v \in \mathbb{N}^d$ takie, że $u \equiv_n v$ wtedy $u \in S \iff v \in S$. Zbiór $S \subseteq \mathbb{N}^d$ jest *unarny* jeśli istnieje liczba $n \in \mathbb{N}$ taka, że powyżej progu n zbiór S jest modularny modulo n. Innymi słowy jeśli $u, v \in \mathbb{N}^d$ takie, że dla każdgo $i \in \{1, \ldots, d\}$ albo $u[i] = v[i]$ albo $u[i], v[i] \geq n$ oraz $u[i] \equiv v[i] \mod n$ wtedy $u \in S \iff v \in S$. Przypomnijmy, że dla alfabetu $\Sigma = \{a_1, \ldots, a_k\}$ i słowa $w \in \Sigma^*$ jego *obraz Parikha*, oznaczany $\text{PI}(w)$, to wektor $v \in \mathbb{N}^k$ taki, że $v[i]$ to liczba wystąpień litery a_i in w. Obraz Parikha rozszerza się naturalnie na języki, $\text{PI}(L) = \{\text{PI}(w) \mid w \in L\} \subseteq \mathbb{N}^k$. Można łatwo zaobserwować, że obraz Parikha języka regularnego jest zawsze unarny, co jest powodem, dla którego uznajemy zbiory unarne za ważne.

Podczas analizy zbiorów osiągalności VASSów bardzo wygodne jest rozważanie części zbiorów osiągalności, dla których pewne współrzędne są ustalone. Mówiąc zgrubnie takie zbiory nazywamy *sekcjami*.

Dla wektora $v \in \mathbb{N}^d$ i zbioru współrzędnych $J \subseteq \{1, \ldots, d\}$ niech $v[J] \in \mathbb{N}^{|J|}$ będzie wektorem v ograniczonym do współrzędnych z J. Precyzyjnie mówiąc zbiór $T \subseteq \mathbb{N}^c$ jest *sekcją* zbioru $S \subseteq \mathbb{N}^{c+d}$ jeśli istnieje zbiór współrzędnych J, $|J| = d$ oraz liczby $a_1, \ldots, a_d \in \mathbb{N}$ takie, że

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

Główny wynik [CCLP17b] jest następujący.

Twierdzenie 5. *Problemy modularnej i unarnej separowalności są rozstrzygalne dla sekcji zbiorów osiągalności VASSów.*

Domknięcie przemienne języka $L \subseteq \Sigma^*$ to zbiór wszystkich słów, które mają ten sam obraz Parikha, co pewne słowo z L. Nietrudno jest pokazać, że regularna separowalność domknięć przemiennych języków VASSów redukuje się do unarnej separowalności ich obrazów Parikha. A zatem Twierdzenie 5 implikuje następujący wniosek.

Wniosek 6. *Problem regularnej separowalności dla domknięć przemiennych języków VASSów jest rozstrzygalny.*

Żeby przekazać ideę dowodu Twierdzenia 5 przypomnijmy, że zbiór $S \subseteq \mathbb{N}^d$ jest *liniowy* jeśli $S = \{b + n_1v_1 + \ldots + n_kv_k \mid n_1, \ldots, n_k \in \mathbb{N}\}$ dla pewnych $b, v_1, \ldots, v_k \in \mathbb{N}^d$. Poniżej skupiamy się jedynie na zbiorach modularnych, podobny argument działa dla zbiorów unarnych. Dowód Twierdzenia 5 bazuje na następującym lemacie.

Lemat 7. *Niech $U, V \subseteq \mathbb{N}^d$ będą dwoma sekcjami zbiorów osiągalności VASSów. Wtedy następujące warunki są równoważne:*

1. *U i V nie są modularnie separowalne,*
2. *istnieją zbiory liniowe $L_U \subseteq U$, $L_V \subseteq V$ takie, że L_U oraz L_V nie są modularnie separowalne.*

Dowód Lematu 7 (oraz podobnego lematu dla zbiorów unarnych) jest istotą głównego argumentu w [CCLP17b]. Bazując na Lemacie 7 można pokazać rozstrzygalność modularnej separowalności dwóch sekcji VASSów U i V w następujący sposób. Rozważmy dwie semi-procedury. Pierwsza, *pozytywna*, próbuje znaleźć świadectwo, że U i V są modularnie separowalne. Ta procedura jest prostsza, wystarczy próbować coraz większe liczby $n \in \mathbb{N}$, dla każdej z nich jest rozstrzygalne (przez redukcję do problemu osiągalności w VASSach) czy U i V są modularnie separowalne modulo n. Jeśli U i V są modularnie separowalne, to pozytywna semi-procedura istotnie znajdzie w pewnym momencie świadectwo na to. Druga, *negatywna*, próbuje znaleźć świadectwo, że U i V nie są modularnie separowalne. Dzięki Lematowi 7 takie świadectwo może być parą zbiorów liniowych (L_U, L_V) spełniających punkt 2. A zatem procedura negatywna przeszukuje coraz większe pary zbiorów liniowych i dla każdej z nich sprawdza, czy spełnia on a punkt 2 Lematu 7. Jeśli U i V nie są separowalne wtedy istotnie znajdzie ona w pewnym momencie taką parę. W pierwszym momencie nie jest jasne, że sprawdzanie warunku 2 jest prostsze niż samo sprawdzanie, czy U i V są modularnie separowalne. Jednakże, faktycznie jest ono o wiele prostsze. Istnieje prosty algorytm na sprawdzanie modularnej separowalności dla dwóch zbiorów liniowych. Natomiast sprawdzanie, czy dany zbiór liniowy jest zawarty w danej sekcji zbioru osiągalności VASSa jest rozstrzygalne dzięki [Ler13]. W rzeczywistości w [CCLP17b], przedstawiamy też inny dowód, który nie korzysta z dodatkowych wyników. Dowodzimy tam silniejszej wersji Lematu 7 dla pewnej *specjalnej postaci* zbiorów liniowych, dla których jest oczywiste z definicji, że są one zawarte w U i w V. Wówczas wystarczy, że procedura negatywna sprawdzi jedynie, czy L_U i L_V są modularnie separowalne.

Automaty jednolicznikowe Następna praca jest również częścią mojego głównego osiągnięcia [CL17]. Główny jej wynik jest następujący.

Twierdzenie 8. *Regularna separowalność dla sieci jednolicznikowych jest* PSPACE*-zupełna.*

Przypomnijmy, że sieć jednolicznikowa, to to samo co jednowymiarowy VASS, czyli automat jednolicznikowy bez testów na zero. Kluczowym pojęcie, które pozwoliło pokazać Twierdzenie 8 to *aproksymant*. Niech $\mathcal{A}$ będzie siecią jednolicznikową ze zbiorem stanów Q, zbiorem tranzycji T postaci (p, a, q, z), konfiguracją początkową $(q_{\text{init}}, 0)$ oraz konfiguracją końcową $(q_{\text{fin}}, 0)$. Dla każdej liczby $n \in \mathbb{N}$ definiujemy *n-ty aproksymant* sieci $\mathcal{A}$, oznaczany $\mathcal{A}_n$, jako następujący automat skończony:

- stany $\mathcal{A}_n$ to $Q \times \{0, \ldots, n-1\} \times \{\text{low}, \text{high}\}$,
- dla każdej tranzycji (p, a, q, z) automat $\mathcal{A}_n$ posiada kilka tranzycji

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

Idea stojąca za aproksymantem $\mathcal{A}_n$ jest taka, że licznik $\mathcal{A}$ jest symulowany dokładnie dopóki nie przekroczy wartości n. Jeśli ta wartość zostanie przekroczona, to wtedy $\mathcal{A}_n$ trzyma jedynie (w stanie) wartość licznika modulo n i sprawdza, czy koniec obliczenia wygląda poprawnie. Dla każdego n zachodzi $L(\mathcal{A}) \subseteq L(\mathcal{A}_n)$ oraz aproksymanty maleją przy rosnącym n w następującym sensie: dla n będącego wielokrotnością k zachodzi $L(\mathcal{A}_n) \subseteq L(\mathcal{A}_k)$. Następujący lemat jest kluczową obserwacją (Corollary 10 z [CL17]).

Lemat 9. *Dla dwóch sieci jednolicznikowych $\mathcal{A}$ i $\mathcal{B}$ następujące warunki są równoważne:*

1. *$L(\mathcal{A})$ i $L(\mathcal{B})$ są regularnie separowalne,*
2. *$L(\mathcal{A}_n)$ i $L(\mathcal{B})$ są rozłączne dla pewnego $n > 0$,*
3. *$L(\mathcal{A}_n)$ i $L(\mathcal{B}_n)$ są rozłączne dla pewnego $n > 0$.*

A zatem aby rozstrzygnąć regularną separowalność wystarczy skupić się na punkcie (2). Punkt (2) można rozstrzygnąć w PSPACE. Bardzo z grubsza mówiąc rozważa się rodzaj produktu $\mathcal{A}_n$ i $\mathcal{B}$, który jest podobny do dwu wymiarowego VASSa (2-VASS). Używając pewnych charakteryzacji osiągalności w 2-VASSach z pracy [BFG$^+$15] oraz innych wyników z dziedziny opracowaliśmy algorytm działający w PSPACE, który rozstrzyga, czy (2) zachodzi dla danych $\mathcal{A}$ i $\mathcal{B}$.

W [CL17] pokazujemy również dwa inne wyniki. Pierwszy to PSPACE-trudność problemu regularnej separowalności dla sieci jednolicznikowych, co jest drugą częścią Twierdzenia 8. Drugi wynik to nierozstrzygalność regularnej separowalności dla automatów jednolicznikowych (test na zero jest teraz obecny). Używamy interesującej, w mojej opinii, techniki, którą widziałem tylko raz w innym miejscu (w [Hun82]). Ta technika to pokazywanie nierozstrzygalności nie poprzez redukcję z innego nierozstrzygalnego problemu, ale poprzez redukcję o ograniczonym wzroście z dowolnego problemu rozstrzygalnego. W tej sytuacji użycie twierdzenia o hierarchii czasowej (bądź pamięciowej) prowadzi do konkluzji, że nasz problem jest nierozstrzygalny.

VASSy całkowitoliczbowe Rozważaliśmy również inny problem, który jest szczególnym przypadkiem Hipotezy 4. Artykuł rozwiązujący ten problem to [CCLP17a], jednak nie jest on częścią mojego głównego osiągnięcia, więc wspomnę o nim tylko w skrócie. Zdecydowałem się nie włączać tej pracy do listy moich głównych osiągnięć, jako, że nie jest dobrze, żeby ta lista była zbyt długa. Jednakże ta praca mogłaby pod każdym innym względem być częścią mojego głównego osiągnięcia.

W [CCLP17a] pokazujemy rozstrzygalność problemu regularnej separowalności dla VASSów całkowitoliczbowych ($\mathbb{Z}$-VASSów). VASSy całkowitoliczbowe zachowują się dokładnie tak jak VASSy z tą jedyną różnicą, że liczniki mogą być ujemne. Nietrudno jest pokazać, że języki $\mathbb{Z}$-VASSów są podklasą języków VASSów, więc jest to istotnie przypadek szczególny Hipotezy 4.

Języki WSTSów Następna praca, podobnie jak [CCLP17a], nie jest zawarta w moim głównym osiągnięciu, ale jest naturalną częścią tej linii badań [CLM$^+$18]. Pokazaliśmy, że dla dwóch języków WSTSów (Well Structured Transition Systems) K i L spełniających pewne łagodne kryteria regularna separowalność K i L jest równoważne pustości przecięcia $K \cap L$. W szczególności zachodzi to dla K i L będących językami VASSów z warunkiem akceptacji przez stan (a nie przez konfigurację).

4.3.5 Separacja pomaga grom parzystości

Gry parzystości to gry pomiędzy dwoma graczami na grafie $G = (V, E)$. Zbiór wierzchołków V jest podzielony na zbiór V_0 wierzchołków gracza *Even* oraz zbiór V_1 wierzchołków gracza *Odd*. Każda krawędź grafu jest etykietowana *rankiem* zadanym przez funkcję rank : $E \to \mathbb{N}$. *Rozgrywka* rozpoczyna się w wyróżnionym wierzchołku, właściciel aktualnego wierzchołka wybiera krawędź z niego wychodzącą i rozgrywka przenosi się do wierzchołka, do którego prowadzi ta krawędź. W ten sposób rozgrywka jest kontynuowana. Formalnie rozgrywka to ciąg krawędzi $e_0, e_1, \ldots$ wybranych tak jak opisano powyżej. Gracz Even wygrywa grę jeśli największa liczba, która wystąpiła nieskończenie wiele razy w ciągu $\text{rank}(e_0), \text{rank}(e_1), \ldots$ jest parzysta, gracz Odd wygrywa w przeciwnym wypadku.

Gry parzystości są centralne dla wielu tematów związanych z teorią automatów i logiką. W szczególności znajdowanie gracza posiadającego strategię wygrywającą w grze parzystości jest równoważne sprawdzaniu, czy formuła μ-rachunku jest prawdziwa w danym modelu. Ustalenie złożoności obliczeniowej problemu gier parzystości (tj. znalezienie gracza posiadającego strategię wygrywającą) jest dużym otwartym problemem od wielu lat. W szczególności dobrze znana jest hipoteza mówiąca, że złożoność tego problemu jest w rzeczywistości wielomianowa. Do niedawna najszybsze algorytmy były nieco podwykładnicze [JPZ08] o złożoności mniej więcej $\mathcal{O}(n^{\sqrt{n}})$. Ostatnio nastąpił duży przełom w tej dziedzinie, kwazi-wielomianowy algorytm rozwiązujący gry parzystości został zaprezentowany w [CJK+17], ze złożonością około $\mathcal{O}(n^{\log n})$. Niedługo później pojawiło się kilka innych kwazi-wielomianowych algorytmów dla tego problemu [JL17, Leh18].

W naszej pracy [CDF+19], która jest częścią mojego głównego osiągnięcia pokazujemy że wszystkie znane algorytmy kwazi-wielomianowe mogą być przeformułowane na problem znajdowania małych separatorów pewnych dwóch języków słów nieskończonych. Co więcej pokazujemy, że to podejście nie może prowadzić bezpośrednio do wielomianowego algorytmu rozwiązującego gry parzystości, gdyż odpowiednie separatory muszą mieć co najmniej kwazi-wielomianowy rozmiar.

Bardziej precyzyjnie, każda rozgrywka w grze z rankami należącymi do zbioru $\{1, \ldots, d\}$ może być zakodowana jako nieskończone słowo nad alfabetem $\Sigma_d = \{1, \ldots, d\}$. Oznaczmy przez LimsupEven_d zbiór nieskończonych słów nad Σ_d, w których największa liczba występująca nieskończenie wiele razy jest parzysta, i podobnie oznaczmy LimsupOdd_d zbiór słów, w których największa liczba występująca nieskończenie często jest nieparzysta. Nazwiemy graf (gry parzystości) *parzystym* jeśli najwyższy rank na każdym jego cyklu jest parzysty. Podobnie graf (gry parzystości) jest *nieparzysty* jeśli najwyższy rank na każdym jego cyklu jest nieparzysty. Oczywiście większość grafów nie jest ani parzysta ani nieparzysta. Jeżeli jednak pewien graf jest parzysty, to nietrudno zauważyć, że każda rozgrywka na nim jest wygrana przez gracza Even. Podobnie dla grafów nieparzystych. Dla $n \in \mathbb{N}$ definiujemy język $\text{EvenCycles}_{n,d} \subseteq \Sigma_d^\omega$ jako te nieskończone słowa, które reprezentują pewną nieskończoną ścieżkę w parzystym grafie (gry parzystości) o co najwyżej n wierzchołkach oraz rankach ograniczonych przez d. Zauważmy, że języki $\text{EvenCycles}_{n,d}$ mogą być traktowane jako dolna aproksymacja LimsupEven_d, zachodzą następujące inkluzje:

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

Podobnie definiujemy $\text{OddCycles}_{n,d}$. Powiemy, że automat na nieskończonych słowach to *safety automat* jeśli ma pewien wyróżniony zbiór *stanów odrzucających*. Słowo jest akceptowane przez taki automat jeśli istnieje bieg po tym słowie, który nigdy nie odwiedza stanu odrzucającego. Powiązanie pomiędzy separowalnością a rozwiązywaniem gier parzystości zostało najpierw zaobserwowane w [BC18], gdzie zostało użyte do przedstawienia algorytmu [CJK+17] przy użyciu pojęcia separowalności.

Główne wyniki W [CDF+19] przeformułowaliśmy nieco powiązanie pomiędzy separowalnością i rozwiązywaniem gier parzystości i pokazaliśmy następujące kluczowe twierdzenie.

Twierdzenie 10. *Jeśli G jest grą parzystości o n wierzchołkach i rankach ograniczonych przez d, a $\mathcal{A}$ jest deterministycznym safety automatem takim, że $L(\mathcal{A})$ separuje* $\text{EvenCycles}_{n,d}$ *i* $\text{OddCycles}_{n,d}$*, to wtedy istnieje algorytm rozwiązujący G w czasie $\mathcal{O}(|G| \cdot |\mathcal{A}|)$.*

W rzeczywistości w [CDF+19] sformułowaliśmy ten fakt (jako Proposition 3.2) nieco inaczej, ale łącząc Proposition 3.2 z prostą obserwacją mówiącą, że gry safety dadzą się rozwiązać w czasie liniowym otrzymujemy Twierdzenie 10. Dowód Twierdzenia 10 jest nietrudny, głównym wkładem było zauważenie, że takie twierdzenie zachodzi.

Pierwszym głównym wynikiem naszej pracy jest pokazanie, że bazując na wynikach opisanych w [JL17, Leh18] można (w obu przypadkach) skonstruować deterministyczny automat safety o kwazi-wielomianowej liczbie stanów, który nie tylko separuje $\text{EvenCycles}_{n,d}$ i $\text{OddCycles}_{n,d}$, ale nawet separuje $\text{EvenCycles}_{n,d}$ i LimsupOdd_d. Co więcej jest to prawdą również dla podejścia z pracy [CJK$^+$17]. A zatem wydawać by się mogło, że zaprojektowanie szybszego algorytmu dla gier parzystości sprowadza się do znalezienia jeszcze mniejszych deterministycznych automatów safety, które separują $\text{EvenCycles}_{n,d}$ i $\text{OddCycles}_{n,d}$ (albo LimsupOdd_d, jak we wszystkich podejściach do tej pory). W tym momencie wkracza nasz drugi główny rezultat.

Twierdzenie 11. *Każdy deterministyczny automat safety $\mathcal{A}$ taki, że $L(\mathcal{A})$ separuje* $\text{EvenCycles}_{n,d}$ *i* LimsupOdd_d *ma przynajmniej $\binom{\lfloor \lg n \rfloor + d/2 - 1}{\lfloor \lg n \rfloor}$ stanów, czyli przynajmniej $n^{\lg(d/\lg n)-2}$ stanów.*

Twierdzenie 11 mówi nam, że w celu znalezienia szybszych algorytmów rozwiązujących gry parzystości musimy użyć jakichś technik innych niż używane do tej pory. Co prawda wciąż jest możliwe, że podejście przez separację jest dobrym kierunkiem, ale w tym celu musimy konstruować automaty takie, że $L(\mathcal{A})$ separuje $\text{EvenCycles}_{n,d}$ i $\text{OddCycles}_{n,d}$, ale nie separuje $\text{EvenCycles}_{n,d}$ i LimsupOdd_d. W mojej opinii powinniśmy traktować Twierdzenie 11 raczej jako wskazówkę gdzie szukać szybszych algorytmów niż jako barierę mówiącą, że ulepszenia będą bardzo trudne.

Techniki dowodowe Dowód Twierdzenia 11 opiera się na pojęciu *drzew uniwersalnych*. Powiemy, że drzewo t *zawiera* drzewo t' jeśli istnieje mapowanie h z wierzchołków drzewa t' w wierzchołki drzewa t takie, że dla każdego wierzchołka n drzewa t' rodzic wierzchołka n jest zmapowany na rodzica wierzchołka $h(n)$, i dodatkowo korzeń t' jest zmapowany na korzeń t. Powiemy, że drzewo jest (ℓ, h)*-uniwersalne* jeśli zawiera każde drzewo o głębokości h (najdłuższa ścieżka z korzenia do liścia ma h krawędzi) z ℓ liśćmi. Na przykład pełne drzewo binarne głębokości 2 jest $(2,2)$-uniwersalne, ale istnieje mniejsze drzewo $(2,2)$-uniwersalne skonstruowane następująco. Korzeń drzewa ma dwójkę dzieci: lewe dziecko ma dwójkę dzieci, natomiast prawe dziecko ma tylko jedno dziecko. To drzewa ma 3 liście, mniej niż 4 liście poprzedniego drzewa. Pokazujemy następujące twierdzenie, które jest używane w dowodzie Twierdzenia 11.

Twierdzenie 12. *Dla dowolnych $\ell, h > 0$ każde (ℓ, h)-uniwersalne drzewa ma przynajmniej $\binom{\lfloor \lg \ell \rfloor + h - 1}{\lfloor \lg \ell \rfloor}$ liści, czyli przynajmniej $\ell^{\lg(h/\lg \ell)-1}$ liści, przy założeniu, że $2h \leq \ell$.*

Druga część dowodu Twierdzenia 11 pokazuje, że dla każdego deterministycznego automatu safety $\mathcal{A}$ takiego, że $L(\mathcal{A})$ separuje $\text{EvenCycles}_{n,d}$ i LimsupOdd_d można wprowadzić pewną strukturę drzewa na stanach $\mathcal{A}$. Co więcej pokazujemy, że to drzewo musi być koniecznie $(n, d/2)$-uniwersalne, co kończy argument.

4.3.6 Problem osiągalności w VASSach

Już w Sekcji 4.3.4 opisaliśmy problem osiągalności w VASSach. Tak jak wspomniano wcześniej algorytm rozstrzygający problem osiągalności został znaleziony przez Mayra w 1981 [May81]. Jednakże algorytm ten był bardzo skomplikowany i żadne górne ograniczenie na jego złożoność nie było znane. Później, w 1982 roku Kosaraju [Kos82] oraz w 1992 roku Lambert [Lam92] opracowali inne algorytmy, również bazujące na tym samym podejściu teraz zwanym KLM dekompozycją (od nazwisk trzech autorów). Pierwsze górne ograniczenie złożoności zostało przedstawione niedawno [LS15], przy czym było ono ekstremalnie dużą sześcienną funkcją Ackermana. Obecnie najlepsze górne ograniczenie na złożoność to funkcja Ackermana, jest to wynik z bardzo niedawnej pracy [LS19], która będzie opublikowana na konferencji LICS 2019. Najlepsze dolne ograniczenie, jeszcze do niedawna, to EXPSPACE-trudność pokazana przez Liptona w 1976 roku [Lip76]. Pomimo wielu wysiłków w kierunku ustalenia złożoności problemu osiągalności postęp był niewielki do czasu naszego ostatniego artykułu [CLL$^+$18]. Jego główny wynik może być zawarty w następującym sformułowaniu.

Twierdzenie 13. *Problem osiągalności dla VASSów jest nieelementarny.*

Przypomnijmy, że oznacza to, że problem osiągalności w VASSach nie należy do n-EXPTIME dla żadnego $n \in \mathbb{N}$. Te własność nazywamy też TOWER-trudnością.

Publikacja Zamierzam włączyć ten rezultat do mojego głównego osiągnięcia, jako, że to jest zdecydowanie moja najlepsza publikacja. Jest zaakceptowana na konferencję STOC 2019, które odbędzie się w czerwcu 2019. Jednak publikacja to nie ma jeszcze (koniec kwietnia) numeru DOI. Pełen tekst jest dostępny na Arxivie `https://arxiv.org/abs/1809.07115`. Mam nadzieję, że dołączenie tej publikacji jest możliwe. Jeśli jest to niemożliwe w związku z formalnymi ograniczeniami, to proszę potraktować tę część opisu jako nie będącą opisem głównego osiągnięcia.

Techniki dowodowe Poniżej nakreślę podstawowe idee konstrukcji, jednakże jestem zmuszony do opuszczenie wielu szczegółów. Niech $!^n$ oznacza n-tą iterację silni, czyli $a!^n = a\overbrace{!\cdots!}^{n}$. Dowód jest, z grubsza rzecz biorąc, redukcją z następującego TOWER-trudnego problemu: dany automat o n stanach, z testami na zero, rozstrzygnąć, czy posiada on bieg akceptujący, w którym wszystkie liczniki są ograniczone przez $3!^n$. W [CLL+18] zostało to sformułowane nieco inaczej, przy użyciu programów licznikowych, jednak na potrzeby tego krótkiego opisu łatwiej jest posługiwać się automatami licznikowymi.

Idea redukcji z tego konkretnego problemu nie jest istotnym elementem rozwiązania, jest tylko sposobem wyrażenia kluczowego wglądu. Główny pomysł znajduje się w konstrukcji VASSów z długimi biegami. Albo raczej, mówiąc precyzyjniej, w opartej na VASSach implementacji liczników, których wartości są ograniczone przez pewne olbrzymie liczby (konkretnie przez $3!^n$), tak, że wciąż możemy te liczniki testować na zero. Łatwo jest zaimplementować taki licznik ograniczony przez pewną małą wartość, na przykład 3. Inicjalizujemy licznik $x = 0$ i jego licznik *kolegę* $\hat{x} = 3$, zawsze będziemy zachowywali niezmiennik $x + \hat{x} = 3$. W tej sposób zapewniamy, że zawsze $0 \leq x \leq 3$. Co więcej jeśli chcemy przetestować, czy x równe jest 0, to postępujemy następująco: inkrementujemy x o 3 (i równocześnie dekrementujemy $\hat{x}$ o 3), następnie dekrementujemy x o 3 (i równocześnie inkrementujemy $\hat{x}$ o 3). Łatwo zauważyć, że ta operacja dochodzi do skutku wtedy i tylko wtedy, gdy $x = 0$. Kluczowym wkładem jest sposób, w jaki można skonstruować licznik ograniczony przez $k!$, który możemy testować na 0 mając do dyspozycji licznik ograniczony przez k, który możemy testować na 0.

Zanim przedstawimy elementy konstrukcji zauważmy, że VASS może łatwo zaimplementować pomnożenie wartości licznika przez ułamek $\frac{a}{b}$, przy czym to mnożenie jest *słabe* w następującym sensie: jeśli wartość licznika przed operacją wynosi x, to po operacji jest ona pomiędzy x a $\frac{a}{b} \cdot x$. Aby zaimplementować słabe mnożenie rozważmy następującego VASSa. Zaczynając w lewym stanie z wartościami liczników równymi $(bn, 0)$ i aplikując pętlę możemy osiągnąć wartości $(0, an)$. Jeśli teraz pójdziemy do prawego stanu i będziemy aplikować pętlę, to możemy osiągnąć wartości $(an, 0)$. A więc całkowity efekt tych operacji to pomnożenie wartości pierwszego licznika przez $\frac{a}{b}$ (oraz zmiana stanu). Jednak, oczywiście, możemy osiągnąć także mniejsze wartości. W szczególności jeśli nie będziemy w ogóle aplikować pętli, to osiągniemy wartości $(bn, 0)$ w prawym stanie. Inna własność tego gadżetu jest następująca: jeśli startujemy z wartościami $(bn + r, 0)$ dla pewnego $r < b$ wtedy maksymalna wartość którą możemy osiągnąć to $(an + r, 0)$. W tym przypadku $\frac{an+r}{bn+r} < \frac{a}{b}$, możemy mnożyć dokładnie przez $\frac{a}{b}$ tylko jeśli wartość licznika jest podzielna przez b. Główną przeszkodą w stworzeniu naszej konstrukcji jest fakt, że nie możemy mnożyć dokładnie, a jedynie słabo.

Sposobem na przezwyciężenie tej przeszkody jest oparcie się na następującym prostym równaniu:

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

Konstruujemy gadżet, w którym startujemy z licznika o wartości x i mnożymy go słabo przez $\frac{2}{1}$, potem mnożymy go słabo przez $\frac{3}{2}$ itd., aż na końcu mnożymy go słabo przez $\frac{k}{k-1}$. Na samym końcu wymagamy by wartość licznika wynosiła dokładnie kx (trzymaliśmy wartość x na jakimś dodatkowym liczniku). Zauważmy, że to jest maksymalna wartość licznika, którą mogliśmy osiągnąć i co więcej jeśli ta wartość została osiągnięta, to wszystkie mnożenia były dokładne. Nasza konstrukcja nie jest aż tak prosta, to jest tylko jej część. Podczas procesu mnożenia x przez ułamki postaci $\frac{i+1}{i}$ modyfikujemy również pozostałe liczniki. W szczególności mnożymy pewien inny licznik y słabo przez $i+1$, ale jesteśmy w stanie pokazać, że to mnożenie musi być dokładne w każdym poprawnym biegu VASSa. Wówczas w każdym poprawnym biegu nasz gadżet mnoży licznik y przez $2 \cdot 3 \cdot \ldots \cdot k = k!$. To znaczy intuicyjnie, że jeśli możemy sprawdzać, czy licznik x wzrósł dokładnie k razy, to możemy też wymusić, by licznik y wzrósł dokładnie $k!$ razy. W podobny sposób konstruujemy VASSa, który jest gadżetem o właściwościach, które mogą być bardzo nieformalnie sformułowane w następującym lemacie.

Lemat 14. *Istnieje gadżet VASS, które mając możliwość testowania na zero liczników ograniczonych przez k dostarcza możliwość testowania na zero liczników ograniczonych przez $k!$.*

Przypomnijmy, że możemy łatwo skonstruować VASSa, które nieformalnie mówiąc dostarcza możliwość testowania na zero liczników ograniczonych przez wartość 3 (jak pokazaliśmy parę paragrafów wcześniej). Zaczynając z tego małego VASSa i składając go n razy z VASSami będącymi gadżetami dostarczonymi przez Lemat 14 otrzymujemy $3!^n$-ograniczone liczniki, które możemy testować na zero. W ten sposób można udowodnić, że problem osiągalności w VASSach jest istotnie TOWER-trudny.

5 Inne prace naukowe

W tej sekcji opisuję skrótowo inne prace naukowe uzyskane po doktoracie, które nie zawierają się w poprzedniej sekcji.

5.1 Pozostałe prace w tematyce systemów nieskończenie stanowych

1. W [CJKS13] rozważamy relację silnej bisymulacji pomiędzy dwoma prostymi klasami systemów nieskończenie stanowych: BPA, czyli systemami generowanymi przez gramatyki w postaci normalnej Greibach oraz BPP, czyli systemami odpowiadającymi sieciom Petriego bez komunikacji. Pokazujemy, że sprawdzanie bisymulacji w tym wypadku jest w EXPTIME.

2. W [CJ15] rozważamy wariant relacji bisymulacji, który bierze pod uwagę ciche tranzycje: bisymulację rozgałęzioną (ang. branching) dla systemów BPA. Poprawiamy rozstrzygalność pokazaną przez Yuxi Fu [Fu13] pokazując, że problem jest w NEXPTIME. Nasza technika polega na wykazaniu, że każda rozgałęziona bisymulacja na BPA ma *bazę* bisymulacji, a poprawność kandydata na bazę może być zweryfikowana w czasie wykładniczym.

3. W [CCH$^+$16] i jego wersji czasopismowej [CCH$^+$19] rozważamy automaty jednolicznikowe. Poprzednio wiadomo było, że jeśli istnieje bieg akceptujący w automacie jednolicznikowym z konfiguracji początkowej postaci $p(0)$ do konfiguracji końcowej postaci $q(0)$, to istnieje również bieg długości $\mathcal{O}(n^3)$, gdzie n to liczba stanów automatu. Poprawiamy to ograniczenie do $\mathcal{O}(n^2)$ przy użyciu skomplikowanych modyfikacji istniejących biegów.

4. Problem osiągalności dla VASSów może być widziany jako pytanie, czy język danego VASSa jest pusty. Jednakże istnieje wiele ciekawych i dużo subtelniejszych pytań na temat języków VASSów. W [CHZ18] pokazujemy, że dla klasy języków VASSów rozstrzygalna jest cała rodzina problemów, która przejawia pewne własności *nieograniczoności*. W szczególności umiemy rozstrzygnąć, czy dany język VASSa jest zawarty w $w_1^* w_2^* \cdots w_n^*$ dla pewnych słów $w_1, \ldots, w_n \in \Sigma^*$. Pokazujemy to przez aplikację KLM dekompozycji wprowadzonej przez Kosaraju, Lamberta oraz Mayra [Kos82, Lam92, May81].

5.2 Wzorce drzewiaste

Wzorce drzewiaste są modelem zapytań XPath, które to są szeroko używane w bazach danych XML. Nieformalnie mówiąc wzorce drzewiaste to drzewa, których celem jest opisanie języków drzew. Każdy wierzchołek takiego wzorca jest etykietowany albo literą ze skończonego alfabetu Σ albo gwiazdką $*$, której celem jest opisanie dowolnej litery. Co więcej krawędzie rodzic-dziecko albo są normalne, albo *długie*, które to opisują relację pomiędzy przodkiem a potomkiem. Język $L(p)$ wzorca p to zbiór wszystkich drzew, które pasują do wzorca.

1. W [CMPP15] analizujemy złożoność obliczeniową następującego problemu inkluzji wzorców: dane dwa wzorce drzewiaste p_1 i p_2, czy język $L(p_1)$ wzorca p_1 jest zawarty w języku $L(p_2)$ wzorca p_2. Rozważaliśmy wiele podklas, w których niektóre z konstrukcji: gwiazdki, długie krawędzie lub normalne krawędzie były zabronione w niektórych wzorcach. Rozważaliśmy też różne warianty problemu. W większości z około tysiąca przypadków udzieliliśmy pełnych odpowiedzi. Interesująca część pracy to kilka przypadków, w których problem staje się rzeczywiście nietrywialny.

2. W [CMNP16] i nieco różnych publikacjach czasopismowych [CMNP17] oraz [CMNP18] rozważamy otwarty przez około dekadę problem otwarty dotyczący minimalnych wzorców drzewiastych. Wzorzec drzewiasty t jest *minimalny* jeśli nie ma żadnego innego wzorca t' o mniejszej liczbie wierzchołków takiego, że $L(t) = L(t')$. Wzorzec jest *redundantny* jeśli możemy usunąć jeden z liści wzorca t i otrzymamy wzorzec o tym samym języku. Oczywiście wzorzec, który jest minimalny nie może być

redundantny. Problem otwarty dotyczył przeciwnej implikacji: czy każdy wzorzec, który nie jest redundantny jest również minimalny i wydawało się, że w istocie tak jest. My przedstawiliśmy kontrprzykład na tę hipotezę i bazując na tym zrozumieniu pokazaliśmy, że następujący *problem minimalizacji* jest Σ_2^P-zupełny: dany wzorzec drzewiasty p i liczba $k \in \mathbb{N}$, czy istnieje wzorzec q o co najwyżej k wierzchołkach taki, że $L(p) = L(q)$? Wynik ten może być również podniesiony do języków grafów, jak opisujemy w [CMNP18].

5.3 Inne

1. W [CGK15] analizujemy ciągi wektorów w $\mathbb{N}^d$ takie, że współrzędne wektora na pozycji $i+1$ są otrzymane ze współrzędnych wektora na pozycji i przez jedną z dwóch operacji: albo 1) zwiększenie o 1, albo 2) zresetowanie do 0. Przedstawiamy przykład ciągu o długości podwójnie wykładniczej względem d bez żadnej pary *dominującej*, czyli wektora v_j na pozycji j co najmniej tak dużego jak wektor v_i na pozycji i, dla jakichś $i < j$. Poprzedni najlepszy przykład był długości wykładniczej.

2. W [CDLM13] i jego wersji czasopismowej [CDLM17] pokazujemy, że problem czy dane wyrażenie regularne ma taki sam język jest pewne deterministyczne wyrażenie regularne jest PSPACE-zupełny.

3. W [CDMP16] i jego wersji czasopismowej [CDMP18] rozważamy drzewa z danymi wraz ze zbiorem ograniczeń. Każde ograniczenie ma następującą formę: jeśli wierzchołek drzewa spełnia pewien warunek (nie biorący pod uwagę danych), to wtedy jego dana musi spełniać pewną pozytywną boolowską kombinację warunków równościowych (dana jest równa pewnym innym danym) i musi też spełniać pewną pozytywną boolowską kombinację warunków nierównościowych (dana jest różna od innych danych). Pokazujemy, że problem sprawdzania, czy w danym języku regularnym drzew jest drzewo z danymi spełniające wszystkie takie ograniczenia jest 2-EXPTIME-zupełny.

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 μ 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