Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Miron Szewczyk
Nr albumu: 383504

Separacja języków VASSów za pomocą języków Z-VASSów

Praca licencjacka
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr. hab. Wojeciech Czerwiński
Instytut Informatyki

Warszawa, Wrzesień 2022

Streszczenie

W pracy badam problem separacji języków VASSów przez języki Z-VASSów. Podaję zarówno interesujące przypadki separacji ja i braku separacji. Szczególnie pochylam się nad VASSami deterministycznymi, które są słabszym modelem obliczeniowym niż VASSy. Przyglądam się również problemowi szukania Z-VASSu równoważnego danemu VASSowi, problem ten rozwiązuje w przypadku jednowymiarowych, deterministycznych VASSów. Korzystając z transducerów, podaję konstrukcję języka nieseparowalnego.

Słowa kluczowe

VASS, Sieć Petriego, Z-VASS, separacja, Vector Addition Systems, teoria języków

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

10003752. Theory of computation
10003753. Models of computation
10003761. Concurrency
10003762. Parallel computing models

Tytuł pracy w języku angielskim

Separating VASS languages using Z-VASS languages

Spis treści

Wprowadzenie

Vector Addition System with States (VASS), kojarzone szerzej jako Sieci Petriego, są jednym z lepiej znanym i zrozumianych modeli obliczeniowych, wykorzystywanych do reprezentacji współbieżnych procesów. Kluczowymi problemami w teorii VASSów jest problem pokrywalności, będący w klasie złożoności ExpSpace-complete, oraz problem osiągalności będący w klasie złożoności Ackermann-complete [5]. VASSy są jednym z silniejszych modeli obliczeniowych dla których problem osiągalności, równoważny problemowi stopu, jest rozstrzygalny. Jeszcze prostszym modelem obliczeniowym są Z-VASSy, dla których problem osiągalności jest już w klasie złożoności PSPACE-complete.

Jednym z pierwszych pytań jaki można zadać badając klasę języków $\mathbb{F}$ jest problem charakteryzacji. Mając daną prostszą klasę $\mathbb{G}$ i język $L \in \mathbb{F}$ odpowiedzieć na pytanie czy $L \in \mathbb{G}$. W pewnych przypadkach problem charakteryzacji uogólnia się do problemu separacji. Mając daną klasę języków $\mathbb{F}$, prostrzą klasę $\mathbb{G}$, dwa języki $L_1, L_2 \in \mathbb{F}$ odpowiedzieć na pytanie czy istnieje $S \in \mathbb{G}$ taki że $L_1 \subset S$ oraz $S \cap L_2 = \emptyset$. Problem separacji jest jednym z klasycznych i szeroko badanych problemów teorii języków, najczęściej dla języków regularnych. Znane wyniki obejmują separacje:

1. językami piecewise testable [4, 8]
2. językami definiowalnymi w logice pierwszego rzędu [9]
3. językami jednoznacznymi [8]

Problem separacji języków VASSów jest nowszy i nie zbadany tak dokładnie. W 2017 roku Wojciech Czerwiński i Sławomir Lasota wykazali [3], że każde dwa rozłączne języki VASSów z akceptacją przez stan możemy odseparować od siebie za pomocą języka regularnego. Nie jest to już prawdą dla języków VASSów z akceptacją przez konfigurację. W 2010 roku Leroux Jerome [6] rozwiązał pokrewny problem, wykazał że każde dwa rozłączne zbiory osiągalności VASSów można zawsze odseparować od siebie za pomocą zbioru semiliniowego. Zbiory semiliniowe odpowiadają zbiorom osiągalności Z-VASSów, naturalnym jest wiec pytanie czy każde dwa języki VASSów z akceptacją przez konfigurację możemy od siebie odseparować za pomocą języka Z-VASSu. Te dwie prace były główną motywacją tej pracy.

W pierwszym rozdziale wprowadzam wszystkie istotne definicje oraz podaje istotne i znane właściwości VASSów i Z-VASSów. W drugim opisuje kontekst badań, przytaczam pokrewne wyniki i opisują jak się łączą z tematem tej pracy. Następnie w trzecim rozdziale podaję istotne przypadki języków separowalnych, dowodzę ich separowalności i rozwiązuję problem separowalności dla podklasy języków deterministycznych VASSów. W czwartym rozdziale wykazuję istnienie VASSów których języków nie da się odseparować. Podaję ogólną konstrukcję skonstruowania VASSu nieseparowalnego, rozważam problem istnienia Z-VASSu równoważnego do danego VASSu i rozwiązuje go dla jednowymiarowych deterministycznych VASSów.

Najistotniejszymi wynikami tej pracy są:

1. Redukcja problemu separacji zbiorów osiągalności za pomocą zbiorów semiliniowych do problemu separacji języków VASSów przez języki Z-VASSów. Pokazuje to, że główny temat pracy jest istotnie trudny.

2. Rozwiązanie problemu separacji dla VASSów deterministycznych

3. Konstrukcja VASSu nieseparowalnego od ustalonego VASSu A

4. Algorytm sprawdzający czy dany jednowymiarowy, deterministyczny VASS jest równoważny Z-VASSowi

Rozdział 1

Podstawowe pojęcia i właściwości

1.1. Definicje

Głównym tematem pracy są języki rozpoznawane przez maszyny nazywane Vector Addition System with States (VASS). O VASSach można intuicyjnie myśleć jak o automacie wyposażonym w stałą ilość liczników, każda tranzycja je modyfikuje lecz nigdy ich wartość nie może spaść poniżej zera. d–wymiarowym VASSem nazywamy krotkę $(Q, T, \Sigma, q_s, v_s, q_f, v_f)$ gdzie

- Q jest skończonym zbiorem stanów,
- $q_s \in Q$ oraz $v_s \in N^d$ stanem i wektorem początkowym,
- $q_f \in Q$ oraz $v_f \in N^d$ stanem i konfiguracją końcową
- $T \in Q \times Z^d \times \Sigma \cup \{\epsilon\} \times Q$ zbiorem tranzycji.

Konfiguracją VASSu nazywamy parę: stan i wektor $q(v)$ gdzie $q \in Q$ oraz $v \in N^d$.

Tranzycje (q, v, a, p) oznaczamy jako $q \xrightarrow[a]{v} p$. Wektor v nazywamy efektem tranzycji, a literą tranzycji, q stanem początkowym a p stanem końcowym.

Biegiem VASSu długości n z konfiguracji $q(v)$ do konfiguracji $p(u)$ nazywamy ciąg tranzycji i kofniguracji :

$$q_1(v_1) \xrightarrow[a_1]{u_1} q_2(v_2) \dots \xrightarrow[a_{n-1}]{u_{n_1}} q_n(v_n)$$

gdzie sąsiednie stany początkowe i końcowe się zgadzają. Tranzycje po literze ϵ traktujemy tak w przypadku automatów.

Badając języki VASSów rozważamy dwa, istotnie różne warunki akceptacji słowa przez VASS. Pierwszym jest akceptacja przez osiągalność, słowo $a_1 \dots a_n$ jest rozpoznawane przez VASS jeśli istnieje poprawny bieg postaci

$$q_s(v_s) \xrightarrow[a_1]{u_1} q_2(v_2) \dots \xrightarrow[a_n]{u_n} q_f(v_f)$$

Drugim jest akceptacja przez stan, słowo $a_1 \dots a_n$ jest rozpoznawane przez VASS jeśli istnieje wektor v oraz poprawny bieg postaci

$$q_s(v_s) \xrightarrow[a_1]{u_1} q_2(v_2) \dots \xrightarrow[a_n]{u_n} q_f(v)$$

Akceptacja przez stan jest prostszym przypadkiem, w tej wersji problem stopu jest PSPACE-zupełny natomiast w przypadku akceptacji przez stan jest on już aż Ackermann-zupełny. W pracy, o ile nie zaznaczę tego wcześniej, będę rozważał wyłącznie akceptacje przez osiągalność.

Przez język VASSu rozumiemy zbiór wszystkich słów rozpoznawanych przez VASS i oznaczamy $L(A)$ gdzie A to VASS.

Z-VASSy są maszynami bardzo zbliżonymi do VASSów. Jedyną różnicą jest to, że w biegach Z-VASSu wszystkie konfiguracje mogą przyjmować wartości z $\mathbb{Z}^d$ zamiast tylko z $\mathbb{N}^d$. Słowa rozpoznawane przez $\mathbb{Z}$-VASSy i języki $\mathbb{Z}$-VASSów rozumiemy analogicznie jak w przypadku VASSów. Mechanizm rozróżniający VASSy od Z-VASSów będziemy nazywali słabym zero testem. Warto zauważyć, że w przypadku akceptacji przez stan Z-VASSy niczym się nie różnią od automatów skończonych, licznik nie jest w żaden sposób wykorzystywany.

Mówimy, że dwa VASSy/Z-VASSy są równoważne jeśli ich języki są sobie równe,

Jeśli mamy dwa rozłączne języki L_1 i L_2 to powiemy, że język L_3 je separuje lewostronnie jeśli $L_1 \subseteq L_3$ oraz $L_3 \cap L_2 = \emptyset$. Piszemy $L_1 \mid_{L_3} L_2$. Piszemy $L_1 \nmid L_2$ jeśli nie istnieje język L_3 który by je separował.

Czasami zamiast języków rozpoznawanych przez VASS rozważamy zbiór konfiguracji osiągalnych. W takim przypadku zapominamy o alfabecie i literach przy tranzycjach. Przez zbiór konfiguracji osiągalnych rozumiemy wszystkie wartości liczników które możemy osiągnąć w stanie końcowym zaczynając z konfiguracji początkowej. Zbiór osiągalności VASSu A o konfiguracji początkowej $q_s(v_s)$ i stanie akceptującym q_f oznaczamy

$$Reach(V) = \{v : q_s(v_s) \rightarrow^* q_f(v)\}$$

1.2. Deterministyczne VASSy

Wiele istotnych przykładów i wyników tej pracy opiera się na deterministycznych VASSach, które są szczególnym przypadkiem VASSów. Determinizm VASSu można rozumieć na dwa sposoby. VASS jest deterministyczny z punktu widzenia syntaktycznego, jeśli dla każdego stanu i litery istnieje co najwyżej jedna tranzycja wychodząca z tego stanu po tej literze. VASS jest deterministyczny z punktu widzenia semantycznego, jeśli dla każdej litery i każdej osiągalnej konfiguracji istnieje co najwyżej jedna tranzycja dozwolona z tej konfiguracji. Zbliżonym pojęciem do deterministycznych VASSów są VASSy jednoznaczne, w których dla każdego słowa istnieje co najwyżej jeden bieg akceptujący. W tej pracy, pod pojęciem VASSu deterministycznego rozumiemy VASS deterministyczny w sensie syntaktycznym. Dzięki temu możemy traktować VASS jako funkcję z konfiguracji i słowa w konfigurację lub $\perp$. Jeśli deterministyczny VASS A znajdujący się w konfiguracji $p(v)$ po przeczytaniu słowa w znajdzie się w konfiguracji $q(u)$ to piszemy $A(p(v), w) = q(u)$. W VASSach deterministycznych nie używamy tranzycji po literze ϵ.

VASSy deterministyczne mają istotnie słabszą moc wyrazu niż ogólne VASSy, istnieją języki rozpoznawane tylko przez niedeterministyczne VASSy.

Twierdzenie 1.2.1 *Istnieją VASSy, dla których nie istnieje równoważny deterministyczny VASS.*

Dowód 1.2.2 *Weźmy VASS A nad alfabetem $\Sigma = \{a, b\}$ który rozpoznaje język*

$$L = \{a^n b \Sigma^* a^n \mid n \in \mathbb{N}\}.$$

Załóżmy, że istnieje deterministyczny VASS $B = (Q, T, \Sigma, q_s, v_s, q_f, v_f)$ który rozpoznaje L. Weźmy $n = |A|$ i słowa

$$w_1 = a^n b a^n,\ w_2 = a^n b a^n b a^n$$

Ponieważ $w_1, w_2 \in L$ to

$$A(q_s(v_s), w_1) = q_f(v_f),\ A(q_s(v_s), w_2) = q_f(v_f)$$

Dodatkowo, ponieważ $w_2 = w_1 ba^n$ to

$$A(q_s(v_s), w_2) = A(q_s(v_s), w_1 ba^n) = A(q_f(v_f), ba^n) = q_f(v_f)$$

Weźmy teraz słowa $w_3 = a^{2n}ba^{2n}$ oraz $w_4 = w_3 ba^n$ $w_3 \in L$ a więc $A(q_s(v_s), w_3) = q_f(v_f)$, natomiast $w_4 \notin L$ Jednak, ponieważ $w_4 = w_3 ba^n$ to $A(q_s(v_s), w_4) = A(q_f(v_f), ba^n) = q_f(v_f)$ czyli $w_4 \in L$ co daje sprzeczność z definicją języka L. Czyli taki B nie może istnieć.

Jedną z zalet rozważania deterministycznych VASSów jest łatwość konstruowania ich dopełnień. Problem konstruowania dopełnień, nierozstrzygalny w ogólnym przypadku, w przypadku deterministycznym ma proste rozwiązanie.

Lemat 1.2.3 *Mając dany deterministyczny VASS A można skonstruować VASS B taki, że*

$$w \in L(A) \Leftrightarrow w \notin L(B)$$

Dowód 1.2.4 *Ustalmy $A = (Q, T, \Sigma, q_s, v_s, q_f, v_f)$. Dla każdego $w \in \Sigma^*$ chcemy wykryć czy w nie ma poprawnego biegu w VASSie A. Dzięki determinizmowi A, po przeczytaniu każdej kolejnej litery A może znajdować się w co najwyżej jednej konfiguracji. Nie istnieje poprawny bieg w po A jeśli zachodzi jeden z trzech warunków:*

1. *Po przeczytaniu całego słowa A dojdzie do konfiguracji innej niż $q_f(v_f)$*
2. *Po przeczytaniu pewnego prefiksu słowa w, A dojdzie do stanu z którego nie ma tranzycji wychodzącej po kolejnej literze w*
3. *Po przeczytaniu pewnego prefiksu słowa w, A dojdzie do konfiguracji z której istnieje tranzycja wychodząca po kolejnej literze, lecz skorzystanie z niej spowodowałoby spadnięcie licznika poniżej zera.*

Skonstruujemy VASS B który na początku niedeterministycznie zgaduje, który z trzech warunków sprawia że słowo wejściowe w nie posiada biegu akceptującego w A. Następnie symuluje bieg w po VASSie A do momentu aż niedeterministycznie zgadnie, że ów warunek zaszedł. Sprawdzenie każdego warunku jest kwestią prostego VASSowego gadżetu. Ponieważ to pierwsze miejsce gdzie wspominam o podobnej kontrukcji, opiszę ją w szczegółach.

1. *Najpierw dodajmy do VASSu nowy stan z, stan q'_f, tranzycje $z \xrightarrow[\epsilon]{-1_i} z$ dla każdego i, oraz tranzycje $z \xrightarrow[\epsilon]{0} q'_f$. Stan z odpowiada za zerowanie wszystkich współrzędnych, konfiguracja $q'_f(0)$ jest nową konfiguracją końcową.*
2. *By wykryć, że na koniec biegu jesteśmy w stanie innym niż końcowy q_f, dodajemy do B nową tranzycje, po literze ϵ i nie zmieniającą liczników, z każdego stanu poza q_f do z.*

   Wykrycie innej wartości licznika na koniec biegu jest ciut bardziej subtelne. Dla każdej współrzędnej licznika i, dodajemy do B dwa nowe stany i^+ oraz i^-. Dla stanu końcowego VASSu A q_f dodajemy do nowego VASSu B tranzycje $q \xrightarrow[\epsilon]{+1_i} i^+$ oraz $q \xrightarrow[\epsilon]{-1_i} i^-$. Dla każdego stanu i^+, oraz współrzędnej licznika $j \neq i$, dodajemy do nowego VASSu B tranzycję $i^+ \xrightarrow[\epsilon]{-1_j} i^+$ oraz $i^+ \xrightarrow[\epsilon]{+1_j} i^+$. Dodatkowo dla każdego stanu i^+ dodajemy tranzycję $i^+ \xrightarrow[\epsilon]{+1_i} i^+$. Analogicznie postępujemy z wszystkimi stanami i^+. Na koniec, dla każdego stanu i^+, i^- dodajemy tranzycję $i^+ \xrightarrow[\epsilon]{-v_f} q'_f$

Po tych wszystkich modyfikacjach, jeżeli na koniec biegu VASSu A jesteśmy w konfiguracji $q_f(v'_f)$, gdzie $v'_f \neq v_f$ to możemy wybrać jedną współrzędną na której v_f i v'_f się różnią o conajmniej 1, następnie zinkrementować lub zdekrementować tę współrzędną o conajmniej 1, a każdą pozostałą o dowolną liczbę. W efekcie, możemy uzyskać wartość dokładnie v_f, dzięki czemu po przejściu tranzycją $\xrightarrow[\epsilon]{-v_f} q'_f$ wylądujemy dokładnie w stanie $q'_f(0)$.

3. *By wykryć, że znajdujemy się w stanie z którego nie ma tranzycji po obecnej literze dla kazdego stanu q z którego nie wychodzi tranzycja po literze a dodajemy do VASSu tranzycję $q \xrightarrow[\epsilon]{0} z$*

4. *By wykryć, że nie możemy użyć tranzycji t $q \xrightarrow[a]{v} p$, najpierw zgadujemy, że wartość licznika i jest mniejsza niż $-v[i]$ (nie pozwala skorzystać z tranzycji). Realizujemy to dodając nowy stan z_i, odpowiedzialny za zerowanie każdej współrzędnej poza i, tranzycje $z_i \xrightarrow[\epsilon]{-1_j} z_i$ dla każdego $j \neg i$ oraz $z_i \xrightarrow[\epsilon]{0} q'_f$. Następnie dodajemy tranzycje $q \xrightarrow[a]{-1_i} z_i, q \xrightarrow[a]{-2_i} z_i, \ldots q \xrightarrow[a]{-(v[i]-1)_i} z_i$. Te tranzycje odpowiadają za wyzerowanie współrzędnej i, mogą ją zdekrementować o co najwyżej $v[i]-1$.*

Dowód ten istotnie korzysta z determinizmu VASSu A. Gdyby A był niedeterministyczny, to B musiałby symulować jednocześnie każdy możliwy bieg słowa w po A. Taka konstrukcja jest wykonalna dla automatów, gdzie musimy śledzić tylko wszystkie możliwe stany, lecz jest niemożliwa dla VASSów, ponieważ rozmiar zbioru konfiguracji w których możemy się znaleźć rośnie wykładniczo ze względu na długość słowa.

Z lematu 1.2.1 wiemy, że istnieją VASSy niedeterminizowalne. Okazuje się, że nie tylko istnieją ale też nie da się rozpoznać, czy dany VASS jest determinizowalny.

Twierdzenie 1.2.5 *Następujący problem jest nierostrzygalny:*

DETERMINIZACJA VASSU

Dane: *VASS A*

Problem: *Czy istnieje deterministyczny VASS B, taki że $L(A) = L(B)$*

Dowód 1.2.6 *Dowód wykorzystuje techniki podobne jak dowód lematu 1.2.3 o braniu dopełnień. Ustalmy T, maszynę z licznikami i zerotestami, która ma co najwyżej jeden poprawny bieg. Jak dobrze wiadomo, problem stopu jest nierostrzygalny dla maszyn Turinga, które są równoważne maszynom z licznikami i zerotestami. Dowolny bieg maszyny z zerotestami można zakodować jako ciąg krotek (stan maszyny, zawartość liczników) i tranzycji.*

VASSy są zbyt słabym modelem obliczeniowym by móc zweryfikować czy dany zapis jest poprawnym biegiem. Ponieważ problem osiągalności konfiguracji jest rozstrzygalny dla VASSów to, to gdyby były dość silne by weryfikować poprawność zapisu to przy ich pomocy moglibyśmy rozwiązać problem stopu dla maszyn z zerotestami i maszyn Turinga.

Są natomiast wystarczające silne by rozpoznać niepoprawne biegi. To znaczy, dla danej maszyny z zerotestami można łatwo skonstruować VASS który rozpoznaje niepoprawne zapisy biegów tej maszyny. Zapis może być niepoprawny z powodów czysto składniowych, liczniki w sąsiednich konfiguracjach mogą się nie zgadzać lub zerotest może być niepoprawnie wykonany. Każdy z tych powodów może być wykryty przez VASS.

Niech A będzie VASSem, takim że $L(A) = \{$ niepoprawne zapisy biegów maszyny $T\}$. Załóżmy, że potrafimy zdeterminizować A. Niech B będzie deterministycznym VASSem, takim że $L(B) = L(A)$. Ponieważ B jest deterministyczne to zgodnie z 1.2.3 potrafimy skonstruować VASS C taki, że $L(C) = \overline{L(B)}$. Czyli C akceptuje dokładnie poprawne biegi maszyny T. Sprawdzając czy C posiada jakikolwiek akceptujący bieg (co można zrobić w Ackermannowym czasie [5, 7]) możemy rozwiązać problem stopu dla maszyny T, co jest oczywistą sprzecznością.

Czyli nie może istnieć algorytm który znajduje determinizację VASSów. Warto zauważyć, że to twierdzenie nie implikuje twierdzenia 1.2.1, teoretycznie mogą istnieć VASSy dla których równoważny deterministyczny VASS istnieje ale jest nie obliczalny.

Warto zauważyć, że ani twierdzenie **??** nie implikuje lematu 1.2.1 ani lemat 1.2.1 nie implikuje twierdzenia **??**.

Rozdział 2

Kontekst badań

Separacja języków jest jednym z klasycznych, szeroko badanych problemów informatyki teoretycznej. W ogólnej formie można go sformułować następująco:

PROBLEM SEPARACJI

Dane: Języki L_1, L_2 z klasy $\mathbb{F}$ i klasa języków $\mathbb{G}$.
Problem: Czy istnieje język $L_3 \in \mathbb{F}$ taki, że $L_1 \mid_{L_3} L_2$.

Problem ten jest rozważany dla różnych klas $\mathbb{F}$ i różnych klas $\mathbb{G}$. W tym rozdziale przybliżę kilka interesujących wyników i podam intuicję dlaczego metody zastosowane w nich nie poskutkowały w przypadku separacji języków VASSów przez języki $\mathbb{Z}$-VASSów

2.1. Separacja języków 1-VASSów za pomocą języków regularnych

W pracy [?] Wojciech Czerwiński i Sławomir Lasota zbadali między innymi problem separacji języków 1-VASSów przez języki regularne. Pokazali, że problem ten jest rozstrzygalny w pamięci wielomianowej. Wykorzystali w tym celu technikę nadaproksymacji.

Definicja 2.1.1 *Niech A będzie 1-VASSem, Q zbiorem stanów, $q_s(v_s)$ konfiguracją początkową a $q_f(v_f)$ konfiguracją akceptującą.*

n-aproksymantem A nazwiemy automat A_n który symuluje A dokładnie dla małych wartości licznika i z pewną dokładnością dla dużych. W stanach będziemy trzymali dodatkowe informacje o wartości licznika modulo n oraz o przybliżenie, czy obecny licznik jest duży czy mały. Dokładniej mówiąc, $Q_n = Q \times \{0, 1 \dots n-1\} \times \{LOW, HIGH\}$ gdzie

- *Q jest kopią stanów VASSu A*
- *0,1, ... , n-1 zapisuje wartość licznika modulo n*
- *LOW,HIGH, przybliża czy wartość licznika obecnie jest duża. Zmienna ta może przejść z LOW na HIGH tylko podczas tranzycji która sprawi, że wartość symulowanego licznika przekroczy n (przekręcenie licznika od góry), z HIGH na LOW może przejść tylko podczas tranyzcji która sprawi, że symulowana wartość licznika spadnie poniżej 0 (przekręcenie licznika od dołu).*

Stan początkowy automatu A to $(q_s, 0, LOW)$ *a stan akceptujący to* (q_f, v_f, LOW).

Jeśli $w \in L(A)$ to dla każdego n $w \in L(A_n)$. Dodatkowo, dla każdego $n, k \in \mathbb{N}$ jeśli $w \in L(A_{k*n})$ to $w \in L(A_n)$

Kluczowym wnioskiem, potrzebnym do rozwiązania separacji 1-VASSów przez języki regularne jest następujący lemat.

Lemat 2.1.2 *Nie A będzie VASSem a R językiem regularnym. Wtedy następujące stwierdzenia są równoważne:*

1. *L(A) i R są rozłączne*

2. *Istnieje n, takie że* $L(A_n)$ *i R są rozłączne.*

Korzystając z tego lematu, wnioskujemy że jeśli separacja VASSów A i B zachodzi, to zachodzi również przez pewnego nadaproksymanta A_n.

Na koniec, autorzy pracy przy pomocy standardowych technik pompowania, pokazują górne wielomianowe ograniczenie na minimalne takie n.

Niestety nie widać jak można przenieść powyższe techniki na grunt separacji języków VASSów przez języki Z-VASSów. By wykazać lemat 2.1.2, kluczowy dla algorytmu separacji, autorzy pracy istotnie korzystają z łatwości pompowania biegów automatów. Biegi Z-VASSów są jednak dużo bardziej złożone.

2.2. Separacja VASSów z akceptacją przez stan za pomocą języków regularnych

W pracy [3] autorzy przyjrzeli się problemowi separacji języków VASSów akceptujących stanem przez języki regularne. Wykazali, że w przypadku akceptacji przez stan, separacja przy pomocy języka regularnego jest równoważna rozłączności separowanych języków.

VASSy z akceptacją przez stan są domknięte w górę, to znaczy jeśli z pewnej konfiguracji q(v) możemy osiągnąć p(u), to dla każdego $v' > v$ z konfiguracji $q(v')$ możemy po tym samym biegu osiągnąć konfigurację p(u') gdzie $u' > u$. Domknięcie w górę jest bardzo silną właściwością, której w oczywisty sposób nie mają VASSy z akceptacją przez konfiguracje, wystarczy spojrzeć na język słów postaci $a^n b^n$.

W pracy autorzy korzystają między innymi z właściwości Well Quasi Order (WQO) i domknięcia w górę. Właściwości, których nie można przenieść na grunt VASSów z akceptacją przez stan.

Tak więc techniki użyte w tej pracy nie dają się przenieść na grunt języków VASSów z akceptacją przez konfigurację, pomimo pozornej bliskości problemów.

2.3. Separacja zbiorów osiągalności VASSów za pomocą zbiorów semilinowych

Problemem zbliżonym do separacji języków VASSów jest separacja zbiorów osiągalności VASSów. Pod pojęciem zbioru osiągalności rozumiemy zbiór wszystkich konfiguracji osiągalnych z konfiguracji początkowej, rozważając je zapominamy o literach tranzycji VASSu lub Z-VASSu. Zbiorami semiliniowymi nazywamy zbiory będące sumą kilku zbiorów liniowych o tym samych

okresach. Dla ustalonego skończonego zbioru baz $R \subset N^d$ i ustalonych okresów $p_1, \dots p_k \in \mathbb{N}^d$ można je przedstawić w następującej postaci:

$$\{r + x_1 * p_1 + \dots + x_k * p_k | r \in R, x_1, \dots, x_k \in \mathbb{N}\}$$

Względnie łatwo wykazać, że zbiory osiągalności Z-VASSów są sumą zbiorów semiliniowych, konfiguracja osiągalne w ustalonym stanie stanowią dokładnie zbiór semiliniowy. Natomiast każdy zbiór semiliniowy możemy przedstawić jako zbiór osiągalności Z-VASSu.

W pracy [6] (uwaga 8.4) Leroux Jerome pokazał, że jeśli weźmiemy dwa VASSy A oraz B, to o ile ich zbiory osiągalności są rozłączne to da się je odseparować od siebie za pomocą zbioru semiliniowego, a więc i za pomocą zbioru osiągalności Z-VASSu.

Zdawałoby się, że jest to wynik bliski separacji języków VASSów przez języki Z-VASSów. Istotnie, zachodzą poniższe lematy:

Lemat 2.3.1 *Niech A będzie n-wymiarowym VASSem. Istnieje VASS A' nad alfabetem* $\Sigma = \{a_1, \dots, a_n\}$ *rozpoznający język*

$$L = \{a_1^{k_1} \dots a_n^{k_n} : (k_1, \dots k_n) \in Reach(A)\}$$

Dowód 2.3.2 *Ustalmy VASS A, z stanem akceptującym* q_f. *VASS A' najpierw symuluje bieg VASSu A, wszystkie tranzycje są po literze* ϵ. *Z stanu* q_f *możemy niedeterministycznie przejść do stanu* p_1. *Z stanu* p_1 *wychodzą dwie tranzycje,* $p_1 \xrightarrow[a_1]{(-1,\dots,0)} p_1$ *oraz* $p_1 \xrightarrow[\epsilon]{(0,\dots,0)p_2}$. *Analogicznie tworzymy stany* $p_2, \dots, p_n$.

Z stanu p_n *możemy więc przejść do stanu* p_{n+1}, *konfiguracja* $p_{n+1}((0, \dots, 0))$ *jest nową konfiguracją akceptującą. Tak zbudowany VASS A' rozpoznaje dokładnie kodowania osiągalnych konfiguracji VASSu A.*

Lemat 2.3.3 *Jeśli każde dwa rozłączne języki VASSów możemy odseparować od siebie za pomocą języka Z-VASSa to każde dwa rozłączne zbiory osiągalności VASSów możemy odseparować za pomocą zbioru semiliniowego*

$$\forall_{A,B} \exists_{Z\text{-}VASS\ Z} L(A)|_{L(Z)} L(B) \Rightarrow \forall_{A,B} \exists_{S\ \text{-}\ zbiór\ semiliniowy} Reach(A)|_S Reach(B)$$

Dowód 2.3.4 *Ustalmy n-wymiarowe VASSy A i B które posiadają rozłączne zbiory osiągalności w stanie akceptującym. Chcemy znaleźć semiliniowy zbiór S które je od siebie odseparuje.*

Zgodnie z lematem2.3.1 stwórzmy VASSy A' i B' które rozpoznają kodowania Reach(A) i Reach(B).

Załóżmy, że istnieje m-wymiarowy Z-VASS Z', taki że $L(A')|_{L(Z')} L(B')$. *Niech konfiguracja początkowa Z' to* $q'_s(v'_s)$.

Ponieważ przecięcia jezyka regularnego i języka Z-VASSu jest rozpoznawane przez Z-VASS to bez straty ogólności, możemy założyć że Z' rozpoznaje język postaci $a_1^* \dots a_n^*$.

Stworzymy teraz Z-VASS Z, taki że L(Z') rozpoznaje kodowania Reach(Z).

Z będzie miał $2 * m$ *wymiary. Posiada prawie te same tranzycje co Z', jeśli Z' posiada tranzycje* $p \xrightarrow[a_i]{v'} q$ *to Z posiada tranzycje* $p \xrightarrow[\epsilon]{v} q$, *v na pierwszych m pozycjach jest równe v', na i-tej pozycji ma 1 a na wszystkich pozostałych ma zero.*

Dodatkowo Z posiada nowy stan akceptujący q_f *do którego możemy przejść niedeterministycznie po tranzycji* $q'_f \xrightarrow[\epsilon]{-v_s} q_f$, *gdzie* v_s *jest równe* v'_s *na pierwszych m pozycjach i równe zero na pozostałych.*

Zdefiniujmy zbiór

$$G = \{v \in \mathbb{N}^{2*m} : v \text{ ma zera na pierwszych } m \text{ pozycjach}\}$$

Przyjrzyjmy się zbiorowi Reach(Z), który jako zbiór osiągalności Z-VASSu jest zbiorem semiliniowym [1]. Do stanu q_f możemy dojść samymi zerami na pierwszych m pozycjach. W takim przypadku, konfiguracja końcowa odpowiada słowu rozpoznanemu przez VASS Z'. Niech $H = Reach(Z) \cap G$. Przecięcie zbiorów semiliniowych jest semiliniowe, więc H też jest zbiorem semiliniowym. Jeśli zrzutujemy H na ostatnie m współrzędnych, zapominając o pierwszych m to uzyskamy szukany separator S.

Czyli gdybyśmy dowiedli, że każda para języków VASSów jest separowalna, to dowiedlibyśmy jednego z głównych wyników pracy [6]. Niestety, jak dowiodę w rozdziale 3, lewa strona implikacji nie zachodzi. Istnieją pary języków VASSów które nie są separowalne.

Sugeruje to natomiast, że problem separacji języków VASSów jest istotnie trudny.

Rozdział 3

Przypadki separacji

Rozważania na problemem, zacznę od pokazania kilku istotnych przypadków łatwej separacji. Dają one istotną intuicję jak patrzeć na problem oraz na możliwości i ograniczenia VASSów/Z-VASSów.

3.1. Separacja deterministycznych VASSów wykorzystując dopełnienia języków

Lemat 3.1.1 *Niech B będzie dowolnym deterministycznym VASSem. Wtedy istnieje Z-VASS Z, taki że:*

$$\forall_A L(A) \cap L(B) = \emptyset \Rightarrow L(A) \mid_{L(Z)} L(B)$$

Takim Z-VASSem jest Z-VASS rozpoznający dopełnienie B, tak jak w lemacie 1.2.3.

Przypadek ten jest ciekaw z kilku względów. Po pierwsze, problem separacji nie zależy w żaden sposób od lewego VASSu. Po drugie, pokazuje że problem jest trywialny dla względnie szerokiej grupy przypadków. By znaleźć przypadek braku separacji, prawy VASS musi istotnie korzystać z niedeterminizmu.

3.2. Niedeterministyczny przypadek

Poprzedni przypadek separacji polegał na determiniźmie prawego VASSu. Co więcej, separujący Z-VASS zależał tylko i wyłącznie od prawego VASSu, był wspólny dla wszystkich możliwych lewych VASSów. Dlatego warto wskazać istotnie niedeterministyczny VASS, który można odseparować lewostronnie od każdego rozłącznego VASSu.

Ustalmy alfabet $\Sigma = \{a, b\}$. Słowa nad tym alfabetem możemy zapisać jako $a^{n_1}ba^{n_2}....ba^{n_k}b$. Niech B będzie VASSem który rozpoznaje słowa w których conajmniej dwa bloki a mają równą długość, czyli $\exists_{i,j} i \neq j \wedge n^i = n^j$. B po przeczytaniu litery b może niedeterministycznie zacząć zliczać długość obecnego bloku inkrementując licznik aż przeczyta następną literę b, niejako zgadując, że to jest jeden z bloków o równej długości. Następnie w identyczny sposób VASS niedeterministycznie zaczyna dekrementować ten sam licznik, zliczając długość kolejnego bloku. Jeśli długości bloków są sobie równe, to licznik po przeczytaniu słowa będzie równy zero.

Języki rozłączne z $L(A)$ składają się z słów zawierających tylko bloki różnej długości.

Język VASSa ma pewne ciekawe właściwości, które pozwalają go odseparować od każdego innego języka 1-VASSa.

Lemat 3.2.1 *Niech A będzie 1-VASSem. Jeśli $L(A) \cap L(B) = \emptyset$ to istnieje liczba $n \in \mathbb{N}$ taka że :*

$$\forall_{w \in L(A)} \#_a(w) < n \vee \#_b(w) < n$$

Innymi słowy, języki rozłączne z $L(B)$ muszą mieć ograniczenie na ilość liter a lub liter b.

Dowód 3.2.2 *Ustalmy A i załóżmy, że teza nie zachodzi. Niech n będzie równe $3*|A|$. Weźmy słowo $w \in L(A)$, takie że $\#_a(w) > n \wedge \#_b(w) > n$. Pokażemy, że możemy tak napompować słowo w by było ciągle akceptowane przez A i zawierało dwa bloki liter a tej samej długości.*

Ustalmy teraz bieg VASSu A po słowie w i przyjrzyjmy się licznikowi tego VASSu podczas biegu. W szczególności przyjrzyjmy się konfiguracjom VASSu po przeczytaniu każdej litery b, nazwijmy je $q_1(v_1), \ldots q_k(v_k)$. Istnieje podciąg tych konfiguracji długości co najmniej XX taki, że wszystkie stany w nim są sobie równe, nazwijmy go $q(u_1), \ldots, q(u_m)$, będziemy na nie mówić konfiguracje wyróżnione. Podzielmy teraz słowo w na trzy podsłowa w_1, w_2, w_3 tak by $w = w_1 w_2 w_3$ oraz by każde z tych podsłów kończyło się na literę b i zawierało co najmniej $\lfloor \frac{m}{3} \rfloor$ liter odpowiadających konfigruacjom wyróżnionym. Czyli w pewnym sensie dzielimy słowo w na trzy "równe"podsłowa.

Reszta dowodu dzieli się na dwa przypadki:

1. *Podczas biegu po słowie w licznik powtórzy swoją wartość w dwóch konfiguracjach wyróżnionych. Niech $w = w_4 w_5 w_6$ gdzie powtarzające się wyróżnione konfiguracje występują na początkach słów w_5 i w_6. W takim wypadku, możemy dowolnie wiele razy napompować podsłowo w_6, czyli słowa $w_4 w_5^* w_6$ też są akceptowane przez VASS A, chociaż zawierają dowolnie dużo bloków liter a o tej samej długości.*

2. *Sytuacja z pierwszego przypadku nie ma miejsca, każda wyróżniona konfiguracja ma inną wartość licznika. Nie możemy zastosować wtedy tak prostego pompowania jak w pierwszym przypadku.*

   Przyjrzyjmy się wartościom licznika w wyróżnionych konfiguracjach, występujących podczas czytania w_2, środkowego podsłowa. Ponieważ wartości licznika w wyróżnionych konfiguracjach się nie powtarzają, musimy znaleźć dwie konfiguracje $q(u_i), q(u_j)$ gdzie $i < j$ oraz albo $u_i < u_j$ albo $u_i > u_j$. To jest pół szukanej pompy. Jeżeli zachodzi $u_i < u_j$ to z podsłowa w_3 bierzemy dowolną pompę w dół po podsłowie w^-, musi istnieć ponieważ wartości licznika w w_3 są dowolnie duże a na koniec licznik osiąga wartość v_f. Pompa w dół nie musi zawierać litery b, oznaczmy jej efekt na licznik jako -k. Niech podsłowo po którym przechodzimy z $q(u_i)$ do $q(u_j)$ nazywa się w^+, oznaczmy $l{=}u_j - u_i$.

   Możemy teraz napompować jednocześnie słowo w^+ i w^-, jeżeli powtórzymy słowo w^+ k razy a słowo w^- powtórzymy l razy to słowo uzyskane w efekcie wciąż będzie rozpoznawane przez VASS A.

   Przypadek gdzie $u_i > u_j$ rozwiązujemy analogicznie, drugie podsłowo bierzemy z podsłowa w_1 zamiast w_3.

Rozdział 4

Przypadki braku separacji

Pierwszym pytaniem które należy zadać badając problem separacji, jest pytanie o przypadek braku separacji. W niektórych przypadkach każdą parę rozłącznych języków można odseparować. Przykładowo, nie istnieje para języków VASSów z warunkiem akceptacji przez stan [2], która by była nieseparowalna przez język regularny.

4.1. Przypadek deterministyczny

W lemacie 1.2.3 pokazaliśmy, że jeżeli prawy VASS jest deterministyczny to da się go odseparować od każdego innego VASSu. Odwracając sytuację i używając niemal identycznych argumentów jak przy dowodzie 1.2.3 możemy skonstruować pierwszy przypadek pary nieseparowalnych VASSów.

Lemat 4.1.1 *Niech A będzie deterministycznym VASSem który nie jest równoważny żadnemu Z-VASSowi. Wtedy, istnieje VASS B, taki że* $A \nmid B$

Dowód 4.1.2 *Niech B będzie VASSem rozpoznającym dokładnie dopełnienie A, dzięki lematowi 1.2.3 wiemy jak go skonstruować. Wtedy, nie istnieje Z-VASS Z taki że* $A \mid_Z B$*. Gdyby istniał, to ponieważ musi zawierać A i mieć puste przecięcie z B, musiałby by być równy dokładnie A. Co jest sprzeczne z założeniami o A. Jako przykładowy A można wziąć język poprawnych nawiasowań.*

4.2. Przeciwobraz języka nawiasowań

Wygodnymi narzędziami do badania problemu separacji są języki poprawnych nawiasowań oraz transducery. Narzędzia te pozwalają sprowadzić problem do pojedynczego, kanonicznego przypadku który łatwo rozwiązać a rozwiązanie przenieść na przypadek ogólny. Dodatkowo, dają ciekawy wgląd w problem istnienia Z-VASSu równoważnego danemu VASSowi.

Definicja 4.2.1 *Językiem n-nawiasowań* $N_n \subset \{a_1, b_1, \ldots, a_n, b_n\}$ *nazwiemy język w którym*

$$\forall_{w \in N_n} \forall_i \forall_{s\, prefiks\, w} \#_{a_i}(s) \geq \#_{b_i}(s)$$
$$\forall_{w \in N_n} \forall_i \#_{a_i}(s) = \#_{b_i}(s)$$

Innymi słowy, jak ustalimy słowo $w \in N_n$ *oraz liczbę i to podsłowo w składające się wyłącznie z liter* a_i, b_i *musi definiować poprawne nawiasowanie.*

Definicja 4.2.2 *Transducerem nazwiemy automat, gdzie każda tranzycja jest dodatkowo etykietowana literą/literami wyjściowego alfabetu. Jeśli wejściowy alfabet to* Σ *a wyjściowy to* Γ *to transducer T definiuje funkcje :*

$$T : \Sigma^* \to \Gamma^*$$

Następnych kilka lematów [1] pokazuje, że języki VASSów/Z-VASSów są zamknięte ze względu na branie obrazów transducerów, oraz że język nawiasowań jest w pewnym sensie "podstawowym" językiem VASSów.

Lemat 4.2.3 *Jeśli A to VASS, T to transducer to język* $T(L(A))$ *jest rozpoznawany przez VASS. Jeśli A to Z-VASS, T to transducer to język* $T(L(A))$ *jest rozpoznawany przez Z-VASS.*

Lemat 4.2.4 *Języki VASSów są zamknięte na branie przeciwobrazów transducreów. Jeżeli B jest VASSem a T transducerem to istnieje A, taki że*

$$L(B) = T^{-1}(L(A))$$

Lemat 4.2.5 *Jeśli A jest n wymiarowym VASSem to istnieje transducer T i język n-nawiasowań* N_n*, taki że*

$$L(A) = T(N_n)$$

Lemat 4.2.6 *Dany jest VASS A. Jeśli A jest Z-VASSem to nie istnieje transducer T, taki że* $T(L(A)) = N_n$*. Ujmując to inaczej, jeśli istnieje T, które przenosi* $L(A)$ *na* N_n *to A nie ma równoważnego Z-VASSu.*

Jest to ciekawa własność transducerów i języków nawiasowań. Korzystając z niej możemy zdefiniować istotną rodzinę przypadków braku separacji.

Twierdzenie 4.2.7 *Ustalmy n-wymiarowy VASS A który nie jest równoważny żadnemu Z-VASSowi. Jeśli istnieje transducer T, taki że* $T(L(A)) = N_n$ *to istnieje VASS B którego języka nie da się odseparować od języka A.*

Dowód 4.2.8 *Ustalmy n-wymiarowy VASS A i transducer T,* $T(L(A)) = N_n$*. Weźmy dopełnienie języka* N_n*, nazwijmy je* $\overline{N_n}$*,* $w \in \overline{N_n} \equiv w \notin N_n$*. Łatwo skonstruować taki VASS C, że* $L(C) = N_n$*. Zgodnie z lematem4.2.4, można skonstruować VASS B który rozpoznaje dokładnie przeciw obraz języka* $\overline{N_n}$ *w T,* $L(B) = T^{-1}(\overline{N_n})$*.* $L(A) \cap L(B) = \emptyset$ *ponieważ nic z* $L(A)$ *nie może być w przeciwobrazie* $\overline{N_n}$ *względem F skoro obrazem* $L(A)$ *jest* N_n*.*

Załóżmy, że istnieje Z-VASS Z, taki żę $L(A) \mid_{L(Z)} L(B)$ *i przyjrzyjmy się* $F(L(Z))$*. Ponieważ* $L(A) \subset L(Z)$ *to* $F(L(A)) = N_n \subset F(Z)$*, ale skoro* $L(Z) \cap L(B) = \emptyset$ *to* $L(Z)$ *musi być dokładnie równe* N_n*. Czyli mamy Z-VASS Z i transducer T, takie że* $T(L(Z)) = N_n$*, czyli potrafimy przedstawić język nawiasowań jako obraz języka Z-VASSu względem transducera co jest sprzecznością.*

Czyli Z-VASS Z nie może istnieć.

Intuicyjnie, twierdzenie 4.2.7 mówi, że o ile VASS korzysta choć trochę z właściwości słabego zero testu, to istnieje VASS którego od niego nie odseparujemy. Pozwala też znaleźć świadka braku separacji, niestety brak separacji nie implikuje jego istnienia.

Osobnym problemem pozostaje jak wykryć czy dany język można przekształcić na język nawiasowań. W kolejnej sekcji wrócę do tego problemu i pokaże jego rozwiązanie w pewnym szczególnym przypadku

4.3. Rozpoznawanie czy VASS jest Z-VASSem

Istotnym problemem w kontekście tej pracy jest następujący problem:

Istnienie równoważnego Z-VASSu

Dane: VASS A

Problem: Czy istnieje Z-VASS B równoważny VASSowi A, czyli $L(A) = L(B)$

Problem ten jest zaskakująco trudny. Nie udało mi się znaleźć żadnych badań na ten temat, pomimo łatwego sformułowania problemu i tego jak podstawowym zagadnieniem jest. Wiadomo, że problem regularności VASSów jest nierostrzygalny w ogólnym przypadku [10]. Dopiero po ograniczeniu się do VASSów deterministycznych, problem staje się rozstrzygalny. Przedstawię teraz prosty algorytm który rozwiązuje problem istnienia równoważnego Z-VASSu, przy założeniu że VASS A jest deterministyczny i posiada tylko jeden wymiar.

Lemat 4.3.1 *Problem istnienia równoważnego Z-VASSu jest rostrzygalny dla deterministycznych 1-VASSów.*

Ustalmy deteterministczny 1-VASS A. Patrząc na tranzycje miedzy stanami znajdźmy wszystkie silne spójne składowe. Cały VASS możemy przedstawić jako skierowany, acykliczny graf silnych spójnych składowych (DAG - directed acyclic graph). Każdy DAG możemy przedstawić jako skończoną sumę skończonych ścieżek. Czyli VASS A jest równoważny sumie VASSów $A_1, \dots A_n$ gdzie każdy VASS A_i odpowiada za pojedynczą ścieżkę pokrywającą DAG spójnych składowych. Dzięki determinizmowi A, każdemu słowu w odpowiada co najwyżej jeden A_i taki że $w \in L(A_i)$ Jeśli każdy A_i jest równoważny pewnemu B_i to istnieje B równoważne A. Jeśli któryś A_i nie jest równoważny Z-VASSowi to znajdziemy świadczące o tym przykłady (korzystające z mechanizmów pompowania) które zaświadczą też o tym, że A nie jest równoważne Z-VASSowi. Czyli A jest równoważne Z-VASSowi wtedy i tylko wtedy gdy każdy A_i jest równoważny Z-VASSowi. Przedstawię i udowodnię teraz algorytm sprawdzenia czy jeden A_i jest równoważny pewnemu Z-VASSowi oraz jego konstrukcję.

Twierdzenie 4.3.2 *Niech A będzie deterministycznym 1-VASSem, którego silne spójne składowe tworzą ścieżkę. Istnieje Z-VASS B, równoważny A wtedy i tylko wtedy gdy w żadnej silnej spójnej składowej nie istnieje para cykli o wspólnym stanie, takich że jeden cykl ma efekt ściśle dodatni a drugi ściśle ujemny.*

Intuicyjnie, VASS jest istotnie VASSem (czyli nie istnieje równoważny Z-VASS) jeśli znajdziemy fragment który korzysta z słabego zerotestu.

Dowód 4.3.3 *Niech A będzie deterministycznym 1-VASSem, którego silne spójne składowe tworzą ścieżkę i żadna spójna składowa nie zawiera dwóch cykli takich jak w sformułowaniu twierdzenia.*

Czyli A ma następującą postać

$$S_1 \to S_2 \to \cdots \to S_n$$

gdzie $S_1, \dots, S_n$ są silnie spójnymi składowymi. Dzięki właściwościom A z założeń, w każdej silnie spójnej wszystkie cykle mają niedodatni albo nieujemny efekt. Nazwijmy silnie spójne składowe dodatnimi i ujemnymi.

Weźmy teraz nieujemną silnie spójną składową S_i. Dzięki jej nieujemności, istnieje $k \in \mathbb{N}$ które jest w pewnym sensie ograniczeniem na dekrementację w ramach tej silnej spójnej składowej. Ściślej mówiąc

$$\forall_{q \in Q_{S_i}} \forall_{n \in \mathbb{N}} \forall_w S_i(q(n), w) - n \leq k$$

Czyli możemy zastąpić silnie spójnie składową S_i przez nowy "gadżet" który jest kopią S_i która dodatkowo na początek "konsumuje" zawartość licznika i zapisują ją w stanie, przy czym nie konsumuje więcej niż k. Następnie symuluje działanie S_i operując na wartości zapisanej w stanie zamiast na liczniku, przy czym nie pozwala by wartość licznika zapisana w stanie nie spadła poniżej zera. Jeśli wartość licznika zapisana w stanie przekroczy wartość k to jest gwarancja, że wewnątrz tej spójnej składowej wartość licznika nie może już spaść poniżej zera. Tak więc można "skonsumować" wartość licznika zapisaną w stanie, zinkrementować "normalny" licznik i dalej operować już tylko na nim nie przejmując się słabymi zerotestami. Podobnie wychodząc z silnie spójnej składowej należy 'skonsumować" wartość licznika zapisaną w stanie i zinkrementować "normalny" licznik.

Przypadek z niedodatnimi silnymi spójnymi składowymi jest trochę trudniejszy. Weźmy niedodatnią silnie spójną składową S_i, skonstruujemy dla niej Z-VASSowy odpowiednik Z_i Dzięki jej niedodatności, istnieje $k \in \mathbb{N}$ które jest w pewnym sensie ograniczeniem na inkrementację w ramach tej silnej spójnej składowej. Ściślej mówiąc

$$\forall_{q \in Q_{S_i}} \forall_{n \in \mathbb{N}} \forall_w S_i(q(n), w) - n \geq k$$

W stanie będziemy trzymali informację "o ile obecna wartość licznika jest większa od najmniejszej wartości licznika wewnątrz obecnej silnej spójnej składowej". Dzięki ograniczeniu na k, wartość ta może być tylko z przedziału $[0, k]$. Korzystając z tej informacji oraz z wartości na "normalnym" liczniku możemy przy wyjściu z silnie spójnej składowej sprawdzić, czy najmniejsza wartość na biegu wewnątrz obecnej silnie spójnej składowej była większa od zera. W tym celu, najpierw dekrementujemy wartość na liczniku o wartość zapisaną w stanie, a następnie sprawdzamy czy możemy osiągnąć wartość zero używając dekrementujących, niedeterministycznych tranzycji po ϵ. Ponieważ ta operacja "konsumuje" wartość na liczniku, konstruując Z-VASSowy odpowiednik S_i musimy dodać nową kopię licznika. Wszystkie silnie spójne składowe przed S_i muszą dodatkowo używać tej kopi by trzymać na niej aktualna wartość, nie używają jednak tej kopii w "gadżetach" wprowadzonych w tej konstrukcjii. Wszystkie silnie spójne składowe po S_i nie tykają tego licznika. Nowy stan akceptujący na pozycji odpowiadającej kopii licznika ma zero. W ten sposób, upewniamy się, że skonsumowaliśmy cały licznik podczas sprawdzania minimalnej wartości przy wyjściu z silnej spójnej składowej.

W ten sposób możemy zastąpić każdą VASSową spójną składową za pomocą Z-VASSowej spójnej składowej. W efekcie uzyskujemy Z-VASS B równoważny oryginalnemu VASSowi A

Wrócmy na chwilę do problemu szukania deterministycznego transducera przenoszącego język danego Z-VASSa na język nawiasowań N_1. Korzystając z spostrzeżeń z dowodu poprzedniego twierdzenia potrafimy udowodnić następujące kryterium:

Lemat 4.3.4 *Niech A będzie deterministycznym 1-VASSem, w którym istnieją dwa cykle o wspólnym stanie. Jeden cykl ma efekt $k*n$, drugi ma efekt $-k*m$ gdzie liczby n,k,m są ściśle dodatnie a liczby n i m względnie pierwsze. Istnieje wtedy transducer T który przenosi język L(A) na dokładnie język nawiasowań N_1, $T(L(A)) = N_1$*

Dowód 4.3.5 *Istnieją liczby naturalne p_1, q_1, takie że $n * p_1 - m * q_1 = 1$ oraz p_{-1}, q_{-1}, takie że $n * p_{-1} - m * q_{-1} = -1$. Ustalmy słowo $w = w_1 w_2 \in L(A)$, takie by VASS a po*

przeczytaniu słowa w_1 znajdował się w stanie s, wspólnym dla obu cykli z sformułowania. Niech w_3, w_4 odpowiadają słowom, jakie VASS A musi przeczytać by znajdując się w stanie s skorzystać z cyklu dodatniego/ujemnego. T będzie transducerem, który tłumaczy tylko słowa postaci $w_1((w_3{}^{p_1}w_4{}^{q_1})|(w_3{}^{p-1}w_4{}^{q-1}))^*w_2$, tak że , następnie za każdym razem jak przeczyta podsłowo

1. *Najpierw czyta prefiks w_1 nie produkując żadnych liter wyjściowych*
2. *Następnie za każdym razem jak przeczyta całe podsłowo $w_3{}^{p_1}w_4{}^{q_1}$ lub $w_3{}^{p-1}w_4{}^{q-1}$ to produkuje literę a lub b*
3. *Na koniec czyta sufiks w_2 i akceptuje*

Jeśli któregoś z kroków nie da się wykonać, doczytać któregoś z podsłów $w_1, w_3{}^{p_1}w_4{}^{q_1}, w_3{}^{p-1}w_4{}^{q-1}, w_2$ do końca, T przerywa bieg (nie ma poprawnej tranzycji).

Rozdział 5

Podsumowanie

W pracy, przyjrzałem się problemowi separacji języków VASSów przez Z-VASSy. Pokazałem, że problem ten trywializuje się w przypadku języków VASSów deterministycznych. Wykazałem redukcję problemu separacji zbiorów osiągalności VASSów [6] do problemu separacji języków VASSów. Przedstawiłem liczne przypadki braku separacji, na podstawie twierdzenia 4.2.7 sformułowałem ciekawą metodę wykazywania braku separcji.

W trakcie badań pojawił naturalny problem "Czy dany VASS jest równoważny pewnemu Z-VASSowi", dla którego nie są znane żadne wyniki. Rozwiązałem ten problem w dla podklasy 1-wymiarowych, deterministycznych VASSów 4.3.1 oraz postawiłem hipotezę, że albo język VASSu można przenieść na język nawiasowań przy pomocy transducera albo język jest językiem Z-VASSu. Przyjrzenie się tej hipotezie wydaje się naturalnym dalszym tokiem badań.

Podobnie istotnym kierunkiem badań jest znalezienie (lub wykazanie jego nie istnienia) algorytmu, który przyjmuje język Z-VASSu i sprawdza czy jest on świadkiem separacji. Zbadanie pustości przecięcia języka VASSu i Z-VASSu jest klasycznym problemem, wystarczy stworzyć VASS będący ich ilorazem i zbadać jego pustość. Problem, czy język danego VASSu zawiera się w języku danego Z-VASSu może być już trudniejszy.

Nie rozwiązałem problemu separacji w ogólnym przypadku, ciągle pozostaje otwartym pytaniem czy problem jest rozstrzygalny.

Bibliografia

[1] Mikołaj Bojańczyk. *An Automata Toolbox.* https://www.mimuw.edu.pl/ bojan/papers/toolbox.pdf, 2017.

[2] Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Separability of reachability sets of vector addition systems. *CoRR*, abs/1609.00214, 2016.

[3] Wojciech Czerwinski and Slawomir Lasota. Regular separability of well structured transition systems. *CoRR*, abs/1702.05334, 2017.

[4] Wojciech Czerwinski, Wim Martens, and Tomás Masopust. Efficient separability of regular languages by subsequences and suffixes. *CoRR*, abs/1303.0966, 2013.

[5] Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is ackermann-complete. In *FOCS*, pages 1229–1240. IEEE, 2021.

[6] Jérôme Leroux. The general vector addition system reachability problem by presburger inductive invariants. *Log. Methods Comput. Sci.*, 6(3), 2010.

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

[8] Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. *CoRR*, abs/1304.6734, 2013.

[9] Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Log. Methods Comput. Sci.*, 12(1), 2016.

[10] Rudiger Valk and Guy Vidal-Naquet. Petri nets and regular languages. *JOURNAL OF COMPUTER AND SYSTEM SCIENCES*, 23, 1981.