Wydział Matematyki, Informatyki i Mechaniki UW

Nieskończone alfabety

skrypt z wykładu

semestr zimowy 2011/2012

Przedmiot prowadził Mikołaj Bojańczyk

Teksty spisali Paweł Borycki
Wojciech Czerwiński
Andrzej Findeisen
Marek Kiszkis
Filip Mazowiecki
Joanna Ochremiak
Michał Skrzypczak
Tomasz Wysocki
Michał Żak

Spis treści

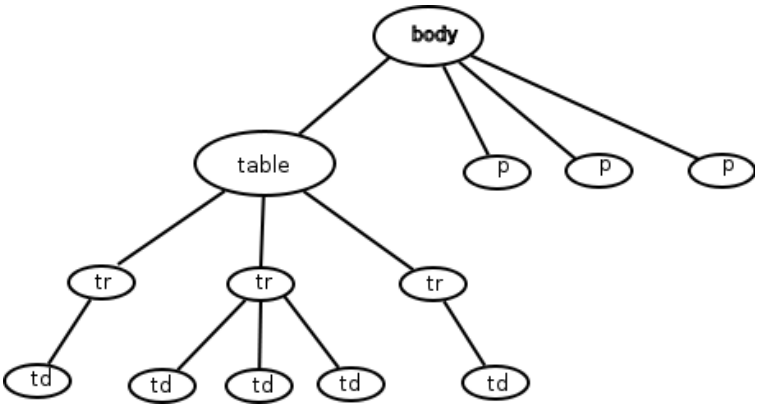
1 Wykład I: wstęp do nieskończonych alfabetów

Spisali: Tomasz Wysocki, Andrzej Findeisen

1.1 Dlaczego warto badać nieskończone alfabety?

W dzisiejszych czasach bardzo popularnym formatem dokumentów stał się XML. Jest to format bardzo wygodny do zapisu drzew. Mając taki zapis, chcielibyśmy móc badać własności takiego dokumentu.

Przykładowo, rozważmy prosty dokument HTML. Jest to drzewo o wierzchołkach etykietowanych pewnym alfabetem $\mathbb{A} = \{body, p, tr, table, td, ...\}$ (Rysunek 1).



Rysunek 1: Schemat dokumentu HTML

Naturalną potrzebą jest zbadanie takich własności jak:

- Każdy element z etykietą tr ma rodzica etykietowanego table
- Korzeniem jest body

Do badania takich własności używa się automatów drzewowych.

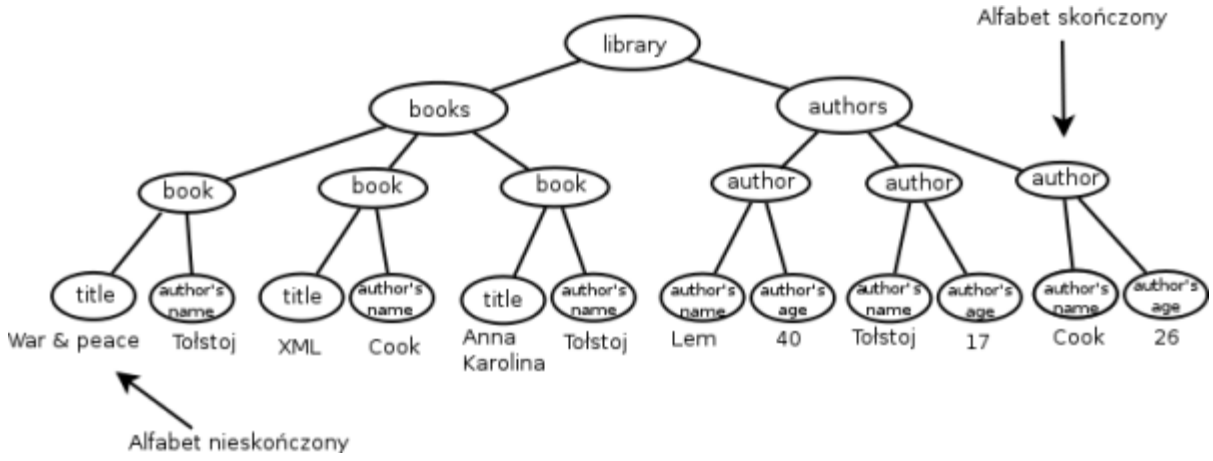
Jednak wiele własności dokumentów znika w użytej abstrakcji dokumentu. W szczególności nie bierzemy pod uwagę żadnych danych pojawiających się w dokumencie.

Rozważmy więc następujący schemat dokumentu opisującego bibliotekę (rys. 2).

Część z jego etykiet pochodzi ze skończonego alfabetu (np. library, book, author), a część z nieskończonego (np. Tołstoj, 17). W takim modelu możemy na przykład zadawać następujące własności:

- Każdy autor napisał jakąś książkę.
- Któryś autor napisał dwie książki.

Zauważmy, że nie są to własności, które można zbadać automatem skończonym. Aby je zbadać trzeba w jakiś sposób zapamiętać etykietę pochodzącą z nieskończonego zbioru, a do tego potrzeba nieskończonego zbioru stanów w tradycyjnym podejściu.



Rysunek 2: Biblioteka

1.2 Znane modele automatów dla słów

Większość wykładu dotycząca nieskończonych alfabetów będzie poświęcona modelom operującym na słowach. Decyzja taka wynika z faktu, że problemy dotyczące operacji na drzewach nie wpisują się w tematykę tego przedmiotu.

Przez większość wykładu prezentowane modele będą rozważały jedynie równość elementów z nieskończonego alfabetu. Alfabet nie musi mieć zatem żadnej narzuconej struktury.

Def. niech A będzie skończonym zbiorem „etykiet", D - nieskończonym zbiorem danych. Wtedy strukturę $w \in (AxD)^*$ nazywamy „słowem z danymi".

Większość automatów używanych w trakcie wykładu, będzie operować na słowach w postaci zdefiniowanej powyżej.

1.2.1 Przykłady

Na potrzeby przykładów przyjmujemy, że $w \in D^*$

$L_1 =$ „pierwsza litera jest taka sama jak ostatnia litera"
$L_1 = \bigcup_{d \in D} dD^*d + d$
$L_2 =$ „pierwsza litera pojawia się ponownie"
$L_2 = \bigcup_{d \in D} dD^*dD^*$
$L_3 =$ „jakaś litera pojawia się dwukrotnie"
$L_3 = \bigcup_{d \in D} D^*dD^*dD^*$

1.3 Definicja niedeterministycznego automatu z rejestrami

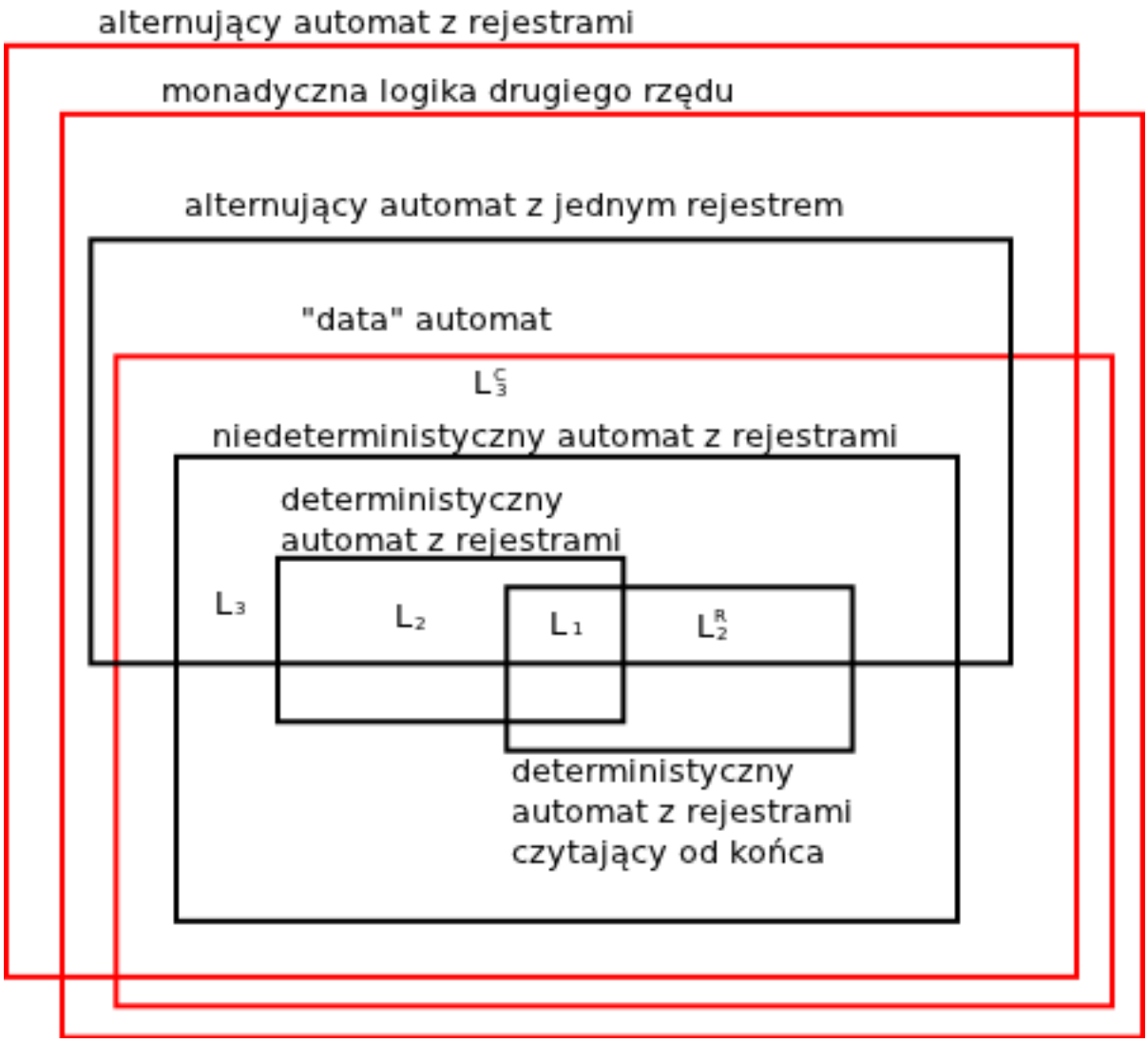
Niedeterministyczny automat z rejestrami jest zadawany przez:

- skończony zbiór etykiet A
- skończony zbiór stanów Q
- stany początkowe i końcowe $I, F \in Q$
- skończony zbiór nazw rejestrów R

- relację zadającą przejścia automatu $\delta \in QxAx2^{R}xQx(R \cup \perp)$

Konfiguracją automatu jest $(q, r) \in Qx(R \rightharpoonup D)$.

Należenie krotki (q, a, r_1, r_3, p, r_4) do relacji przejścia automatu, należy rozumieć jako: automat będąc w stanie q i czytając etykietę a oraz daną, która jest równa z rejestrami r_1, r_3 i nierówna z pozostałymi rejestrami, przechodzi do stanu p i zapisuje czytaną daną do rejestru r_4.



Rysunek 3: Moc wyrazu poszczególnych modeli

2 Wykład II

Spisał: Marek Kiszkis

Podczas dzisiejszego wykładu udowodnione zostanie następujące

Twierdzenie 1:
Niepustość dla niedeterministycznych automatów rejestrowych jest PSPACE-zupełna.

Oznaczenia
Przez $booleq(n)$ będziemy określać zbiór predykatów (nad n zmiennymi), korzystających tylko z " $=$ " i " $\neq$ ". Semantyka jest następujaca: stosujemy to do $n-tek$ danych , tzn $[\![\varphi]\!] \subseteq \mathbb{D}^n$.
Przykład: $\varphi = (x_1 = x_2 \wedge x_2 \neq x_3) \in booleq(3)$ wyznacza zbiór $R = \{(d_1, d_2, d_3) : d_1 = d_2 \wedge d_2 \neq d_3\} \subseteq \mathbb{D}^3$.

Podlemat Niech $\bar{d}, \bar{e} \in \mathbb{D}^n$. Wówczas są równoważne:

- $\forall_\pi \ \bar{d} = \bar{e} \cdot \pi$
- $\forall_{i,j \in \{1,\ldots,n\}} \ \bar{d} \in [\![x_i = x_j]\!] \Leftrightarrow \bar{e} \in [\![x_i = x_j]\!]$

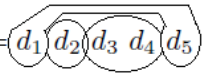
Dowód jest banalny, więc go pomijamy.

Lemat $R \subseteq \mathbb{D}^n$. Są równoważne:

- $R = [\![\varphi]\!]$ dla pewnego $\varphi \in booleq(n)$
- R jest ekwiwariantna

Dowód
($\Downarrow$) oczywiste, samo w sobie
($\Uparrow$) oczywiste, z podlematu

O $\varphi \in booleq(n)$ będziemy mówić, że opisuje *typ równościowy* na n-tkach danych: φ może (ale nie musi) mówić, które współrzędne są sobie równe, a które od siebie różne. Np. $\varphi = (x_1 = x_5 \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3 \wedge x_3 = x_4)$ opisuje taki typ równościowy:

$\bar{d} =$ 

Z kolei np. $\varphi = true$ opisuje taki typ równościowy („nic nie wiadomo"):

$$\bar{d} = d_1 \ d_2 \ d_3 \ d_4 \ d_5$$

Zauważmy, że dla każdego typu równościowego można napisać odpowiadającą mu formułę.

Zdefiniujmy teraz porządnie model obliczeń, o którym mowa w Twierdzeniu 1.
Definicja Niedeterministyczny automat rejestrowy (NRA) to szóstka uporządkowana:

- Q - stany

- $A = A_{fin} \times \mathbb{D}$ - alfabet: skończony zbiór etykiet i nieskończony zbiór danych
- $I, F \subseteq Q$ - stany początkowe i akceptujące
- R - skończony zbiór rejestrów

plus relacja przejscia, ktorą można zdefiniować na dwa sposoby.

Sposób pierwszy: skończony zbiór krotek postaci
$(q_0, a, \varphi, q_1) \in Q \times A_{fin} \times booleq(2|R| + 1) \times Q$

co nalezy rozumieć tak: jeśli jesteśmy w stanie q_0, następna czytana dana jest o etykiecie a, a φ opisuje typ równościowy pomiędzy wartościami w rejestrach przed przeczytaniem następnej danej (pierwsze n współrzędnych), nastepną daną ($(n+1)$-sza współrzędna), i zawartością rejestrów po przeczytaniu tej danej (ostatnie n współrzędnych), to przechodzimy do stanu q_1, z zawartością rejestrów taką, jak nam mówi φ.

(drobna) Uwaga: na specjalne traktowanie zasługuje typ równościowy w przypadku, gdy wartość rejestru nie jest określona. Możemy się jednak umowić tak, że jeśli np. w rejestrze 1 nic nie ma, to dopisujemy nie-równość $x_1 \neq x_1$.

Sposób drugi: oznaczmy dla zwięzłości zbiór konfiguracji $Conf = Q \times (R \to \mathbb{D})$. Wówczas relacja przejścia to pewna relacja $\Delta \subseteq Conf \times A \times Conf$, taka że $\Delta \cdot \pi = \Delta$ dla każdego π.

Aczkolwiek obie te definicje sa równoważne, obie mają swoje zalety i wady: sposób drugi jest o wiele bardziej elegancki :) Jednak gdy potrzebujemy pewnego rodzaju formalizmu (np. mamy zakodować NRA jako wejście dla maszyny Turinga), nie bardzo wiadomo jak zapisać nieskończoną relację w jakikolwiek sposób, także jesteśmy zmuszeni użyć wtedy pierwszego sposobu.

Przykład $L = \bigcup_d (\mathbb{D} - d)^+ d$ (ostatnia litera nie występuje nigdzie wcześniej)
L mozna rozpoznać NRA z 1 rejestrem, i 3 stanami: początkowym q_I, roboczym q i końcowym (akceptującym) q_F. Zapisując przejścia pierwszym sposobem (Uwaga: dla zachowania prostoty pomijamy etykietowanie danych):

- $(q_I, \varphi_1, q), \varphi_1 = (x_2 \neq x_3)$
  Wpisujemy do rejestru *nieokreśloną* - niedeterministycznie zgadniętą wartość, od której wymagamy tylko tyle, zeby była różna od wczytanej danej. Zgadujemy, ze taka będzie ostatnia dana w słowie.
- $(q, \varphi_2, q), \varphi_2 = (x_1 = x_3 \wedge x_2 \neq x_3)$
  Jeśli przeczytana dana jest różna od tej wpisanej w rejestrze, nic nie robimy
- $(q, \varphi_3, q_F), \varphi_3 = (x_1 = x_2)$
  Sprawdzamy, czy faktycznie ostatnia litera jest tą, ktorą przechowujemy w rejestrze

Możemy już teraz przejść do dowodu Twierdzenia 1.

Dowód Niech $\alpha, \beta \in Conf$. Powiemy, że $\alpha \sim \beta$, kiedy $\alpha = \beta \cdot \pi$ dla pewnego π.
Nietrudno zauważyć, że tak zdefiniowana relacja $\sim$ jest relacją równoważności na zbiorze konfiguracji $Conf$.

Zastanówmy się, jakiej mocy jest zbiór ilorazowy $Conf/_\sim$. Na $|Q|$ sposobów możemy wybrać stan dla danej konfiguracji (różne stany $\Rightarrow$ różne klasy abstrakcji, bo permutacje danych oczywiście nie zmieniają stanu). A na ile sposobów możemy wybrać typ rownościowy zbioru R ? Ano na tyle, ile jest relacji równoważności na zbiorze $|R|$-elementowym z wyróżnioną co najwyżej jedną klasą abstrakcji. Wystarczy chwila refleksji, żeby przekonać się, że ta liczba jest co najwyżej wykładnicza względem $|R|$. Nie będziemy się nad tym dłużej zastanawiać.

Definicja Zdefiniujmy teraz relację " $\rightarrow$ " $\subseteq Conf/_\sim \times Conf/_\sim$: powiemy, że $\alpha/_\sim \rightarrow \beta/_\sim$, jeśli istnieją $\alpha' \sim \alpha, \beta' \sim \beta, (a,d) \in A$ takie, że $\alpha' \xrightarrow{(a,d)} \beta'$.

Tradycyjnie, definiujemy " $\rightarrow^*$ " jako domknięcie przechodnio-zwrotne relacji " $\rightarrow$ ".

Lemat (kluczowy)
$\alpha/_\sim \rightarrow^* \beta/_\sim$, wtedy i tylko wtedy, gdy dla pewnych $\alpha' \sim \alpha, \beta' \sim \beta$ zachodzi $\alpha' \rightsquigarrow^* \beta'$. (Tutaj uwaga: druga strzałka dotyczy przejść automatu! Przejście automatu oznaczamy tutaj falowaną strzałka)

Zarys dowodu lematu: oczywiście trudna jest tylko implikacja ($\Rightarrow$). Dowodzimy stosując indukcję po ilości użytych strzałek. Ale jeśli $\alpha/_\sim \rightarrow \beta/_\sim$, i $\beta/_\sim \rightarrow \gamma/_\sim$, to znaczy że istnieją $\alpha' \sim \alpha, \beta' \sim \beta'' \sim \beta, \gamma' \sim \gamma$, takie że $\alpha' \rightsquigarrow \beta', \beta'' \rightsquigarrow \gamma'$. Dalej, skoro $\beta' \sim \beta''$, to istnieje σ takie, że $\beta' \cdot \sigma = \beta''$.

A stąd już wynika, że $\alpha/_\sim \rightarrow \gamma/_\sim$, bo $\alpha \cdot \sigma \rightsquigarrow \beta' \cdot \sigma = \beta''$, a przecież $\beta'' \rightsquigarrow \gamma'$.

Wniosek Język rozpoznawany przez dany automat jest niepusty $\Leftrightarrow$ istnieją $\alpha/_\sim \rightarrow^* \beta/_\sim$, i ponadto:

- w α jest stan początkowy
- w β jest stan akceptujący
- w α są niezdefiniowane wartości rejestrów

Z którego to wniosku mamy od razu

Niedeterministyczny **algorytm** rozstrzygający niepustość
Zgadnij $\alpha/_\sim \in Conf/_\sim$
while w α nie ma stanu akceptującego **do**
niedeterministycznie zgadnij β, takie że $\alpha/_\sim \rightarrow \beta/_\sim$
$\alpha := \beta$

Uwagi

- Bardzo cieszy nas twierdzenie Savage'a, które mówi ze NPSPACE = PSPACE. Zatem podany algorytm istotnie należy do klasy PSPACE.
- Mając kodowanie $\alpha/_\sim$ i $\beta/_\sim$, możemy sprawdzić czy $\alpha/_\sim \rightarrow \beta/_\sim$ w NPTIME (przegladając wszystkie dostępne tranzycje).

3 Wykład III

Spisali: Filip Mazowiecki, Joanna Ochremiak

Twierdzenie 1:
Pustość jest PSPACE-trudna dla niedeterministycznych automatów rejestrowych.

Dowód:
Zredukujemy do naszego problemu następujący problem PSPACE-trudny:
(Wejście): Maszyna Turinga M i dwie konfiguracje tej maszyny c i d o tej samej długości – n.
(Pytanie): Czy maszyna M ma przebieg z c do d, który pozostaje na tej porcji taśmy, którą zajmują te konfiguracje? Ozn. $c \to_M d$.

Idea redukcji jest taka, że kombinacja zdefiniowanych i niezdefiniowanych rejestrów będzie kodować daną konfigurację maszyny M. Formalnie mamy daną maszynę M i dwie konfiguracje c, d długości n. Oznaczmy przez Σ alfabet roboczy maszyny M, a przez Q – jej zbiór stanów.

Definiujemy zbiór rejestrów R następująco:

$$R = \{1, 2, ..., n\} \times \Sigma \times \{Q \cup \{\bot\}\}.$$

Chcemy, aby zdefiniowany rejestr (i, σ, q) oznaczał, że litera σ występuje na i-tym miejscu oraz głowica znajduje się na i-tym miejscu w stanie q. Natomiast zdefiniowany rejestr $(i, \sigma, \bot)$ oznacza, że litera σ występuje na i-tym miejscu, a głowica znajduje się w innym miejscu.

Powiemy, że częściowe wartościowanie $v\colon R \dot{\to} D$ jest poprawne, czyli koduje pewną konfigurację maszyny M, jeśli:

- Dla każdego $i \in \{1, ..., n\}$ istnieje dokładnie jedna para $\sigma \in \Sigma$ oraz $p \in Q \cup \{\bot\}$ taka, że rejestr (i, σ, p) jest zdefiniowany;
- Dla każdego $\sigma \in \Sigma$ oraz $q \in Q$ istnieje dokładnie jedno $j \in \{1, ..., n\}$ takie, że rejestr (j, σ, q) jest zdefiniowany.

Jeśli wartościowanie v jest poprawne, to zadaje konfigurację maszyny M. Oznaczmy ją przez $\underline{v}$.

Lemat 1
Dla dowolnej konfiguracji c rozmiaru n maszyny M istnieje formuła $\varphi_c \in Boolex(|R|)$ taka, że $v \in [\![\varphi_c]\!]$ wtedy i tylko wtedy, gdy

- v jest poprawne;
- $\underline{v} = c$ (wartościowanie v koduje konfigurację c).

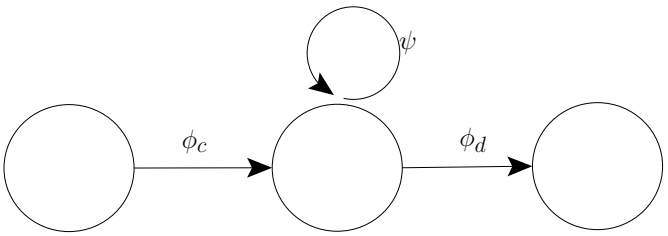
Ponadto rozmiar φ_c jest wielomianowy od (M, n).

Lemat 2
Istnieje formuła $\psi \in Boolex(2 \cdot |R|)$ taka, że $vw \in [\![\psi]\!]$ wtedy i tylko wtedy, gdy

- wartościowania v i w są poprawne;
- maszyna M przechodzi z konfiguracji $\underline{v}$ do konfiguracji $\underline{w}$ w jednym kroku.

Ponadto rozmiar ψ jest wielomianowy od (M, n).

Mając dane M, c, d oraz n konstruujemy taki automat $A(M)$ o trzech stanach, że $A(M)$ akceptuje jakieś słowo wtedy i tylko wtedy, gdy $c \to_M d$, czyli wtedy i tylko wtedy, gdy na maszynie M istnieje bieg ze stanu c do stanu d, który pozostaje na tej porcji taśmy, którą zajmują te konfiguracje.



Rysunek 4: Automat

Na powyższym rysunku przejście ϕ_c oznacza wpisanie do rejestrów konfiguracji c. Przejście ψ oznacza ustawianie rejestrów tak, żeby odpowiadało to przejściom w maszynie M. Ostatnie przejście oznacza, że jeśli w rejestrach mamy konfigurację d to przechodzimy do ostatniego stanu, który jest akceptujący. Można się pozbyć pierwszego stanu, ale wtedy rysunek jest brzydszy :).

Twierdzenie 2 Pytanie, czy niedeterministyczny automat rejestrowy akceptuje każde słowo (uniwersalność) jest nierozstrzygalne.

Dowód:
Zredukujemy do naszego problemu następujący problem nierozstrzygalny:
(Wejście): Maszyna Turinga M z alfabetem Σ i zbiorem stanów Q.
(Pytanie): Czy maszyna M ma bieg akceptujący?

Idea redukcji jest taka, że kodujemy skończone biegi maszyny M jako słowa. Skonstruujemy automat, który sprawdza czy dane słowo nie jest przebiegiem maszyny M. Zbiór danych $\mathbb{D}$ będzie zbiorem liczb naturalnych $\mathbb{D} = \mathbb{N}$.

Aby słowo $w \in (A \times \mathbb{D})^*$ zadawało bieg maszyny M muszą być spełnione między innymi następujące warunki:

- $w = \#w_1\#w_2...\#w_k\#$, gdzie wszystkie w_i są tej samej długości – n;
- dla każdego $i \in \{1, ..., k\}$ i każdego $j \in \{1, ..., n\}$ słowa w_i mają tą samą daną na pozycji j;
- dla każdego $i \in \{1, ..., k\}$ dane na pozycjach $1, .., n$ w słowie w_i są parami różne.

Każde słowo w_i ma odpowiadać kolejnej konfiguracji w pewnym skończonym przejściu maszyny M. Słowo w_i na każdej pozycji ma inną daną, bo ich numer oznacza zawartość odpowiedniej komórki w taśmie. Z tego samego powodu te numery mają się powtarzać w tej samej kolejności w innych słowach w_j. Część A jest skończonym alfabetem, który koduje jaka litera się znajduje w danej komórce, czy jest tam głowica i w jakim stanie się znajduje.

Lemat
Istnieje automat A, dla którego dla każdego słowa $w \in (A \times \mathbb{D})^*$ następujące warunki są równoważne:

- słowo w zadaje przebieg maszyny M;
- A odrzuca słowo w.

Dowód:
Idea jest taka, że akceptujemy wszystkie słowa w, które nie kodują pewnego skończonego przebiegu maszyny M, a zakodowane przebiegi akceptujemy tylko jeśli odpowiadają biegom, które zostają zaakceptowane przez maszynę M. Poniżej są wymienione przypadki, które nie kodują przebiegów i są łatwe do sprawdzenia przez automat:

- w słowie w występuje następujące konfiguracja danych: $...de...df...$ dla $e \neq f$;

- w słowie w występuje następująca konfiguracja danych: $...dd...$;
- słowo nie kończy się znakiem $\#$;
- w słowie w występują pary $(\#, d)$ oraz $(\#, e)$ dla $d \neq e$;
- w słowie w występują pary $(\#, d)$ oraz $(a, \#)$ dla $a \neq \#$;
- fragment słowa w kodujący pojedynczą konfigurację maszyny M nie koduje jej poprawnie;
- dwa kolejne fragmenty słowa w kodujące kolejne konfiguracje maszyny M nie są możliwe do uzyskania po sobie w jednym kroku maszyny M.

Z powyższego lematu wynika, że istnieje automat A, który odrzuca pewne słowo w wtedy i tylko wtedy, gdy M ma bieg akceptujący.

4 Wykład IV

Spisał: Andrzej Findeisen

Przez $\mathbb{D}$ oznaczamy zbiór wartości danych. Zauważmy, że $G = \{\pi : \mathbb{D} \to \mathbb{D} \mid \pi \text{ jest bijekcją } \}$ jest grupą.

Określamy notację działań w grupie: $\pi \cdot \sigma(d) = \sigma(\pi(d))$. Aby było to bardziej czytelne będziemy zapisywać $d \cdot \pi \cdot \sigma$.

Wkrótce będziemy utożsamiać grupę permutacji ze strukturą w zbiorze. W grupie będą permutacje zachowujące strukturę. Zauważmy, że można równoważnie zdefiniować strukturę przez podanie odpowiedniej grupy.

Struktura	Grupa
Brak struktury	Wszystkie permutacje
Porządek	Permutacje zachowujące porządek
Porządek i pewien wyróżniony podzbiór	Permutacje zachowujące porządek i zachowujące należenie do podzbioru

Definicja 1 G-zbiór to para $(X, \cdot)$

- X jest zbiorem
- $\cdot$ jest funkcją $X \times G \to X$ zapisywaną infiksowo jako $x \cdot \pi \in X$

taka, że $x \cdot 1 = x$, $(x \cdot \pi) \cdot \sigma = x \cdot (\pi \cdot \sigma)$

Obserwacja 1 Funkcje

- $x \mapsto x \cdot \pi$
- $x \mapsto x \cdot (\pi^{-1})$

są wzajemnymi odwrotnościami. $(x \cdot \pi) \cdot \sigma = x \cdot (\pi \cdot \sigma)$

Przykład G-zbioru: $(\mathbb{D}, \cdot)$ dla grupy permutacji. Dziedziną jest $\mathbb{D}$. Akcją jest $d \cdot \pi = \pi(d)$

Rozważmy pewne $\sigma \in G$ i G-zbiór $(\mathbb{D}, \cdot_\sigma)$. Dziedziną jest $\mathbb{D}$, akcję definiujemy jako $d \cdot_\sigma \pi := \sigma(\pi(\sigma^{-1}(d)))$. Łatwo sprawdzić, że jest to G-zbiór:

$$x \cdot_\sigma 1 = \sigma(1(\sigma^{-1}(d))) = d$$

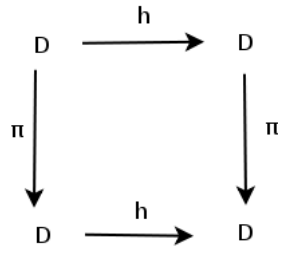
$$(x \cdot_\sigma \pi_1) \cdot_\sigma \pi_2 = \sigma(\pi_2(\sigma^{-1}(\sigma(\pi_1(\sigma^{-1}(d)))))) = \sigma(\pi_2(\pi_1(\sigma^{-1}(d)))) = d \cdot_\sigma \pi_1 \cdot \pi_2$$

G-zbiór $(X, \cdot)$ może być traktowany jako algebra:

- Dziedziną jest X
- Mamy unarną operację $X \to X$ dla każdego π

Wtedy możemy rozważać homomorfizm, izomorfizm, kongruencje.

Obserwacja 2 $(\mathbb{D}, \cdot)$ i $(\mathbb{D}, \cdot_\sigma)$ są izomorficzne (rys. 5).



Rysunek 5: Izomorfizm

Łatwo sprawdzić, że nie jest G-zbiorem:

$$(\mathbb{D}, \circ)$$

$$d \circ \pi = \pi^{-1}(d)$$

W G-zbiorze $(X, \cdot)$ definiujemy orbitę elementu $x \in X$ jako $\{x \cdot \pi | \pi \in G\}$. Orbitę oznaczamy $x \cdot G$.

Lemat 1 Orbity tworzą podział zbioru X. Dowód: proste sprawdzenie.

Teraz należy zadać sobie pytanie: „Co to jest skończoność w G-zbiorach?". Możemy np. rozważać zbiory o skończonej liczbie orbit. Niestety:

Fakt 1 Jest niepoliczalnie wiele nieizomorficznych G-zbiorów z jedną orbitą.

Przykłady G-zbiorów: $\mathbb{D}$, $\mathbb{D}^2$, $\mathbb{D}^*$, $P(\mathbb{D})$, $P_{fin}(\mathbb{D})$, $\mathbb{D}^\omega$, $\mathbb{D}^{(n)} \subset \mathbb{D}^n$ - krotki, w których żadne dwie współrzędne się nie powtarzają

Definicja 2 Niech $(X, \cdot)$ jest G-zbiorem. Wtedy $C \subset \mathbb{D}$ nazywamy wsparciem elementu $x \in X$, jeśli dla każdego $\pi \in G$ zachodzi $\pi_{|C} = id_{|C} \Rightarrow x = x \cdot \pi$

Przykładowo wsparciem krotki $(1, 1, 2, 7, 2, 9)$ jest $\{1, 2, 7, 9\}$. Wsparciem zbioru $\{1, 9, 17\}$ jest $\{1, 9, 17\}$.

Definicja 3 Zbiór nominalny to G-zbiór, gdzie każdy element ma pewne skończone wsparcie.

Rozważmy wsparcia w zbiorze $P(\mathbb{D})$:

$X \subset \mathbb{D}$	Pewne wsparcia
$\{1, 2, 3\}$	$\{1, 2, 3\}$
$\emptyset$	$\emptyset$
$\mathbb{D}$	$\emptyset$, $\mathbb{D}$, $\mathbb{D} - \{1\}$
X	te same, co dla $\mathbb{D} - X$

Wsparcie jest skończone dla skończonych lub ko-skończonych zbiorów.

Przykłady zbiorów nominalnych: $\mathbb{D}$, $\mathbb{D}^2$, $\mathbb{D}^*$, $P_{fin}(\mathbb{D})$

Przykłady zbiorów nienominalnych: $P(\mathbb{D})$, $\mathbb{D}^\omega$

Twierdzenie 1 Każdy jednoorbitowy zbiór nominalny jest izomofriczny z $\mathbb{D}^n$ *modulo* S (zdefiniowane później).

Zauważmy, że nie istnieje homomorfizm (funkcja ekwiwariantna) pomiędzy zbiorami $\mathbb{D}^3$ (5 orbit), a $\binom{\mathbb{D}}{3}$ (1 orbita).

Niech $n \in \mathbb{N}$. Niech dalej $\tau : \{1, \ldots, n\} \to \{1, \ldots, n\}$. Możemy zastosować τ do krotki $\overline{d} \in \mathbb{D}^{(n)}$.

$$\overline{d} = (d_1, \ldots, d_n)$$

$$rearrange_\tau(\overline{d}) = (d_{\tau(1)}, \ldots, d_{\tau(n)})$$

Weźmy $S \leqslant S_n$. Wtedy $\mathbb{D}^n$ *modulo* S jest zbiorem orbit $\mathbb{D}^{(n)}$ względem działania akcji $\{rearrange_\tau\}_{\tau \in S}$.

Fakt 2 $\mathbb{D}^{(n)}$ *modulo* S jest jednoorbitowym zbiorem nominalnym.

5 Wykład V

Spisał: Paweł Borycki

Twierdzenie 5.1 (**o najmniejszym wsparciu**). *Niech X będzie G-zbiorem oraz niech $x \in X$. Załóżmy, że skończone zbiory C oraz E są wsparciem elementu x. Wtedy $C \cap E$ również jest wsparciem elementu x. Zatem każdy zbiór nominalny posiada wsparcie najmniejsze ze względu na relację zawierania.*

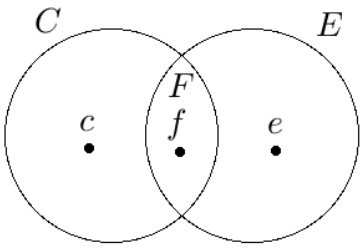
Przecięcie skończonych wsparć pewnego elementu G-zbioru jest więc też wsparciem tego elementu. Najmniejsze wsparcie elementu x będzie oznaczane jako $leastsupport(x)$.

Wniosek. Wsparciem każdego elementu $x \in X$ w zbiorze nominalnym jest co najmniej zbiór C.

Przykład. $1 \in \mathbb{D}$. Najmniejszym wsparciem jest zbiór $\{1\}$. Innym wsparciem jest $C = \mathbb{D} \setminus \{1\}$. Ponieważ zbiór C nie jest skończony, ich przecięcie nie jest wsparciem elementu $1 \in \mathbb{D}$.

Dowód Twierdzenia 5.1 o najmniejszym wsparciu. Dowód ma charakter indukcyjny na $|C \cup E|$. Rozpoczynamy dowodzenie od dowodu dla singletonów.

Niech $F = C \cap E$. Pokażę, że zbiór F jest wsparciem.



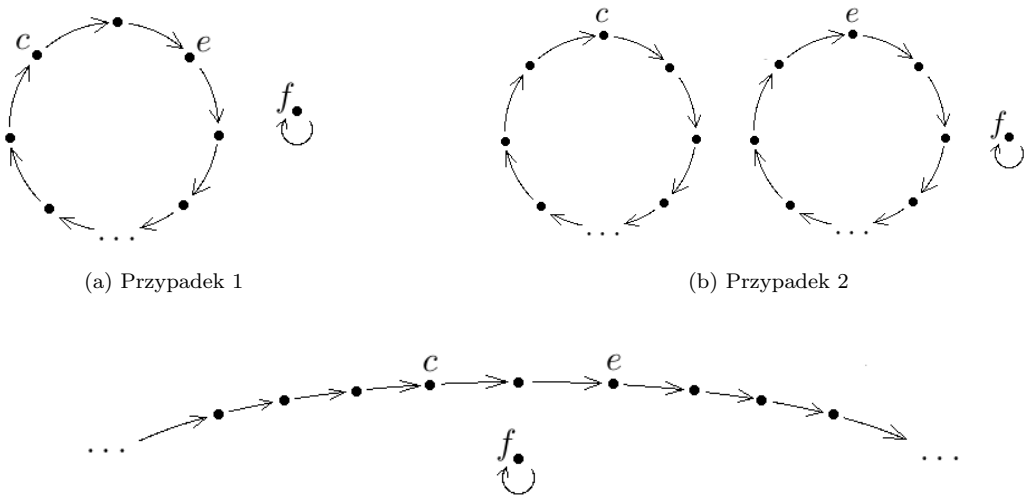
Rysunek 6

Zdefiniujmy następujące grupy:
$G_C = \{\pi : \pi_{|C} = id_{|C}\}$,
$G_E = \{\pi : \pi_{|E} = id_{|E}\}$,
$G_F = \{\pi : \pi_{|F} = id_{|F}\}$.

Lemat 5.1. *Dla każdej akcji $\pi \in G_F$ istnieje taki rozkład $\pi = \pi_1 \cdot \ldots \cdot \pi_n$, że dla każdego $i \in \{1, \ldots, n\}$ zachodzi $\pi_i \in G_C \cup G_E$.*

Niech $\pi \in G_F$. Na podstawie Lematu 5.1 możemy stwierdzić, że:
$(\pi_{|F} = id_{|F}) \Rightarrow \pi = \pi_1 \cdot \ldots \cdot \pi_n$,
$x \cdot \pi = x \cdot \pi_1 \cdot \pi_2 \cdot \ldots \cdot \pi_n$.

Niech $\pi \in G_F$. Możemy rozważyć następujące trzy przypadki:



(a) Przypadek 1

(b) Przypadek 2

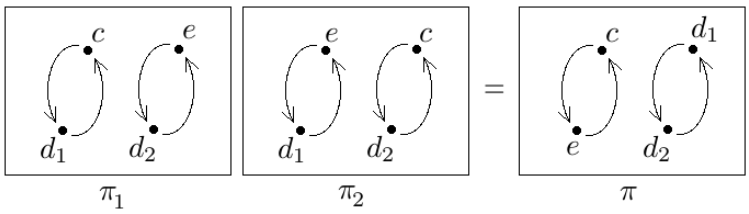
(c) Przypadek 3

Rysunek 7

Element $f \in F$ znajduje się zawsze na trywialnym, jednoelementowym cyklu.

Rozważam **przypadek 1**. Każdy cykl może zostać podzielony na transpozycje zamieniające miejscami dwa elementy: $\pi = \pi_1 \cdot \ldots \cdot \pi_k$, gdzie dla $i \in \{1, \ldots, k\} \pi_i$ to transpozycja.

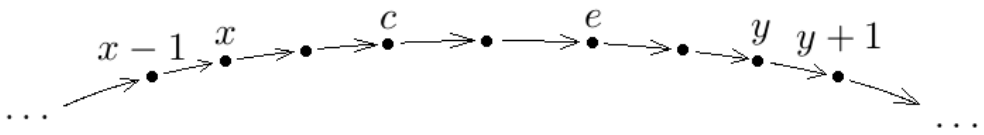
Niech $c, e \notin F$.



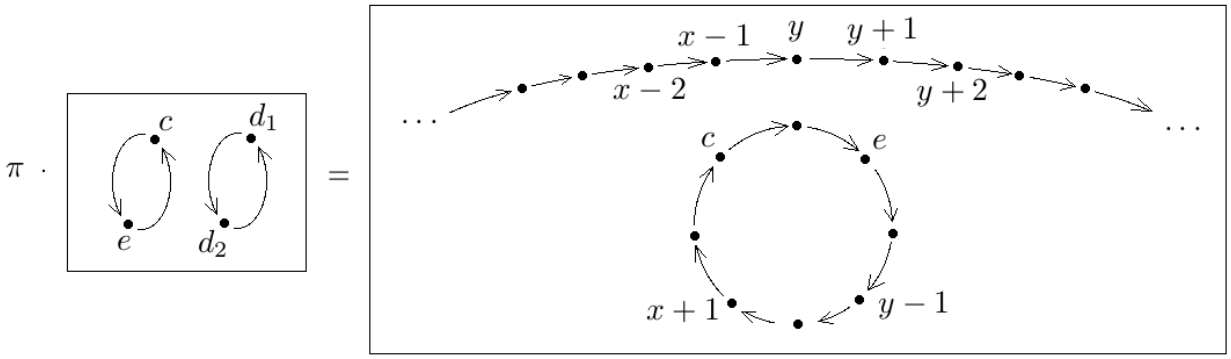
Rysunek 8

Pokazano, że Lemat 5.1 zachodzi, gdy elementy c, e są na tym samym cyklu.

Rozważam **przypadek 3**. Wybierzmy pewne elementy x oraz y.



Rysunek 9



Rysunek 10

Sytuację można zatem sprowadzić do przypadku 1, gdy elementy c i e znajdują się na jednym cyklu. Twierdzenie zachodzi dla zbiorów będących singletonami.
Niech:

- $\{c_1, c_2, c_3\} \in C \setminus E$,
- $\{e_1, e_2, e_3, e_4\} \in E \setminus C$,
- $C' = C \cup E \setminus \{e_4\}$
- $E' = C \cup E \setminus \{c_3\}$

Zbiory C' oraz E' są wsparciem. Wsparciem jest też zbiór $C' \cap E'$. W wyniku dalszego usuwania elementów z C' i E' możemy stwierdzić, że F też jest wsparciem.

Twierdzenie 5.2. *Dla każdego jednoorbitowego zbioru nominalnego X istnieją takie liczba $n \in \mathbb{N}$ oraz grupa S permutacji zbioru $\{1, \ldots, n\}$, że zbiór ten jest izomorficzny z $\mathbb{D}^{(n)}$ mod S.*

Przed udowodnieniem Twierdzenia 5.2 zwróćmy uwagę na następujące lematy.

Lemat 5.2. *Niech X oraz Y będą jednoorbitowymi zbiorami nominalnymi. Dla każdego $x \in X$ oraz $y \in Y$ istnieje co najwyżej jedna funkcja ekwiwariantna $f : X \to Y$ taka, że $f(x) = y$. Funkcja ta istnieje wtedy i tylko wtedy, gdy $leastsupport(y) \subseteq leastsupport(x)$.*

Ponadto, jeśli funkcja f istnieje, jest suriekcją na zbiór Y. Dla każdej permutacji π zachodzi $y \cdot \pi \in range(f)$, gdzie $range(f)$ jest zbiorem wartości funkcji f. Nie musi być to jednak funkcja różnowartościowa.

Przykład. Niech $X = Y = \mathbb{D}$. Na mocy Lematu 5.2 nie istnieje taka funkcja ekwiwariantna f, że $f(1) = 5$. Zbiór $leastsupport(5) = \{5\}$ nie jest zawarty w zbiorze $leastsupport(1) = \{1\}$. Jedyną funkcją ekwiwariantną $f : \mathbb{D} \to \mathbb{D}$ jest funkcja identycznościowa $f = id$.

Niech $f : X \to Y$ będzie ekwiwariantną suriekcją na zbiór Y. Definiujemy relację równoważności $\sim_f$ na zbiorze X:

$$x_1 \sim_f x_2 \text{ jeśli } f(x_1) = f(x_2)$$

Ponieważ f jest funkcją ekwiwariantną zachodzi implikacja:

$$\text{Jeśli } x_1 \sim_f x_2 \text{ to } x_1 \cdot \pi \sim_f x_2 \cdot \pi \text{ dla każdego } \pi \in G.$$

Klasy abstrakcji relacji $\sim_f$ mają strukturę G-zbioru. Możemy więc zdefiniować akcję $\pi \in G$ na klasach abstrakcji tej relacji równoważności:

$$[x]_{\sim_f} \cdot \pi = [x \cdot \pi]_{\sim_f}$$

Rodzinę relacji równoważności $\sim_f$ z tak zdefiniowaną grupą oznaczamy $X/_{ker(f)}$.

Lemat 5.3. *Niech $f : X \to Y$ będzie ekwiwariantną suriekcją na zbiór Y. Wtedy zachodzi izomorfizm*

$$X/_{ker(f)} \equiv Y.$$

Dla ustalonej permutacji $\tau \in S_n$ możemy zdefiniować ekwiwariantną funkcję

$$rearrange_\tau : \mathbb{D}^{(n)} \to \mathbb{D}^{(n)},$$

której wartość jest wyrażona wzorem:

$$rearrange_\tau(d_1, \ldots, d_n) = (d_{\tau(1)}, \ldots, d_{\tau(n)})$$

Lemat 5.4. *Niech $f : \mathbb{D}^{(n)} \to \mathbb{D}^{(n)}$ będzie ekwiwariantną suriekcją na zbiór $\mathbb{D}^{(n)}$. Dla dowolnej permutacji $\tau \in S_n$ oraz dowolnych elementów $\overline{d}, \overline{e} \in \mathbb{D}^{(n)}$ zachodzi równoważność:*

$$f(\overline{d}) = f(rearrange_\tau(\overline{d})) \text{ wtw, gdy } f(\overline{e}) = f(rearrange_\tau(\overline{e})).$$

Dowód Lematu 5.4. Udowodnimy implikację "$\Leftarrow$".

Wybierzmy $\pi \in G$, dla której $\overline{d} = \overline{e} \cdot \pi$. Wtedy zachodzi:

$$f(\overline{d}) = f(\overline{e} \cdot \pi) = f(\overline{e}) \cdot \pi = f(rearrange_\tau(\overline{e})) \cdot \pi = f(rearrange_\tau(\overline{e}) \cdot \pi) =$$

$$= f(rearrange_\tau(\overline{e} \cdot \pi)) = f(rearrange_\tau(\overline{d})).$$

Dowód implikacji "$\Rightarrow$" przebiega analogicznie.

Lemat 5.5. *Niech $f : X \to Y$ będzie funkcją ekwiwariantną. Jeśli zbiór C jest wsparciem elementu $x \in X$, to jest też wsparciem elementu $f(x) \in Y$.*

Lemat 5.6. *Jeśli zbiór $C \subseteq \mathbb{D}$ jest wsparciem elementu $x \in X$, to zbiór $C \cdot \pi$ jest wsparciem elementu $x \cdot \pi$ dla każdej permutacji $\pi \in G$.*

Dowód Twierdzenia 5.2. Niech X będzie jednoorbitowym zbiorem nominalnym i niech $x \in X$. Niech

$$leastsupport(x) = C = \{c_1, \ldots, c_n\}.$$

Najmniejsze wsparcie elementu x istnieje na mocy Twierdzenia 5.1. Elementom zbioru C przypisujemy numery w dowolnej kolejności. Na mocy Lematu 5.2 istnieje funkcja ekwiwariantna $f : \mathbb{D}^{(n)} \to X$ będąca suriekcją na zbiór X, dla której:

$$f(c_1, \ldots, c_n) = x.$$

Na mocy Lematu 5.3 zbiór X jest izomorficzny ze zbiorem $\mathbb{D}^{(n)}/_{ker(f)}$. Znajdziemy podgrupę grupy n-elementowych permutacji $S \leqslant S_n$, dla której zachodzi izomorfizm:

$$\mathbb{D}^{(n)}/_{ker(f)} \equiv \mathbb{D}^{(n)} \bmod S.$$

Niech $S \leqslant S_n$ będzie podgrupą:

$$S = \{\tau \in S_n : f(\overline{d}) = f(rearrange_\tau(\overline{d})) \text{ dla pewnego } \overline{d}\}.$$

Dla każdych $\overline{d}, \overline{e} \in \mathbb{D}^{(n)}$ zachodzi równoważność:

$$f(\overline{d}) = f(\overline{e}) \text{ wtw, gdy } \overline{d} = rearrange_\tau(\overline{e}) \text{ dla pewnego } \tau \in S.$$

Z powyższej równoważności wynika, że

$$\mathbb{D}^{(n)}/_{ker(f)} \equiv \mathbb{D}^{(n)} \bmod S,$$

a więc S jest poszukiwaną w dowodzie podgrupą. Aby zakończyć dowód Twierdzenia 5.2 wystarczy zatem udowodnić powyższą równoważność. Dowód implikacji "$\Leftarrow$" wynika wprost z Lematu 5.4 oraz z definicji podgrupy S. Udowodnimy implikację "$\Rightarrow$":

Niech $\overline{d} = (d_1, \ldots, d_n)$ oraz $\overline{e} = (e_1, \ldots, e_n)$, a także $f(\overline{d}) = f(\overline{e}) = y$. Pokażemy, że $\overline{e}$ oraz $\overline{f}$ składają się z tych samych danych, ale w innej kolejności. Zauważamy, że:

- zbiór $\{d_1, \ldots, d_n\}$ jest wsparciem elementu $\overline{d}$ w $\mathbb{D}^{(n)}$,
- zbiór $\{e_1, \ldots, e_n\}$ jest wsparciem elementu $\overline{e}$ w $\mathbb{D}^{(n)}$.

Na mocy Lematu 5.5 oba zbiory, $\{d_1, \ldots, d_n\}$ oraz $\{e_1, \ldots, e_n\}$, są wsparciem elementu y. Na mocy Twierdzenia 5.1 o najmniejszym wsparciu przecięcie tych dwóch zbiorów, $\{d_1, \ldots, d_n\} \cap \{e_1, \ldots, e_n\}$, również jest wsparciem elementu y. Jeśli zbiory $\{d_1, \ldots, d_n\}$ oraz $\{e_1, \ldots, e_n\}$ nie są równe, element y posiada wsparcie o mocy mniejszej niż n. Ponieważ na jednej orbicie najmniejsze wsparcia mają tę samą moc, element x również posiada wsparcie o mocy mniejszej niż n. Wniosek ten jest sprzeczny z założeniem, że zbiór $\{c_1, \ldots, c_n\}$ jest najmniejszym wsparciem elementu x.

Możemy więc stwierdzić, że $\{d_1, \ldots, d_n\} = \{e_1, \ldots, e_n\}$, a zatem

$$\overline{d} = rearrange_\tau(\overline{e}) \text{ dla pewnego } \tau \in S_n.$$

Na mocy Lematu 5.4 można stwierdzić, że $\tau \in S$.

Dowód równoważności kończy dowód Twierdzenia 5.2.

6 Wykład VI

Spisał: Michał Skrzypczak

Naszym celem w tym rozdziale jest udowodnienie odpowiednika klasycznego twierdzenia Myhilla-Nerodego w kontekście słów z danymi. Najpierw przypomnijmy sformułowanie tego twierdzenia dla skończonych alfabetów (powtórka z JAiO).

Twierdzenie 6.1. *Niech $L \subseteq A^*$ będzie dowolnym językiem. Określmy relację $\sim_L$ na słowach skończonych w następujący sposób*

$$w \sim_L w' \quad wtw. \quad \forall_{v \in A^*} \ (w \cdot v \in L \Leftrightarrow w' \cdot v \in L)\,.$$

Wówczas następujące warunki są równoważne:

1. *zbiór klas abstrakcji $A^*/\sim_L$ jest skończony,*
2. *język L jest regularny.*

Dodatkowo, jeśli zachodzi któryś z powyższych warunków to zbiór $A^*/\sim_L$ można interpretować jako zbiór stanów minimalnego automatu deterministycznego rozpoznającego L. Każdy automat rozpoznający L można uprościć *by otrzymać $A^*/\sim_L$.*

Standardowy przykład to język $L = (aa)^*$ dla którego $\sim_L$ to relacja *słowa mają tę samą parzystość długości*. Wówczas zbiór $A^*/\sim_L$ zawiera dwie klasy abstrakcji: słowa parzystej długości i słowa nieparzystej długości.

Rozważmy teraz jak wygląda zbiór klas abstrakcji powyższej relacji gdy zamiast skończonego alfabetu A rozważymy zbiór danych $\mathbb{D}$. W tym celu wprowadźmy dwa przykładowe języki z danymi:

$$L_1 = \bigcup_{d \in \mathbb{D}} d\mathbb{D}^* d\mathbb{D}^* \text{ — } \textit{pierwsza dana się powtarza},$$

$$L_2 = \left\{abc \in \mathbb{D}^3 : a \neq b \wedge b \neq c \wedge c \neq a\right\} \text{ — } \textit{trzy różne dane}.$$

Będziemy wykorzystywać te języki do ilustracji wprowadzanych pojęć.

Relacja $\sim_{L_1}$ ma następujące klasy abstrakcji (nieskończenie wiele):

1. $\{\epsilon\}$ — klasa zawierająca jedynie słowo puste,
2. $\{1, 12, 17, 158, \ldots\}, \{2, 23, 257, \ldots\}, \ldots$ — nieskończenie wiele klas abstrakcji, każda postaci $d\,(\mathbb{D} \setminus \{d\})^*$,
3. $L_1 = \{11, 121, 2526, \ldots\}$ — jedna klasa zawierająca wszystkie słowa z języka.

Zauważmy, że jakkolwiek klas abstrakcji jest nieskończenie wiele, to jednak w naturalny sposób układają się one w trzy *kategorie* wymienione powyżej. Jak łatwo zgadnąć, sformalizowanie czym są owe *kategorie* wykorzystuje permutacje zbioru danych.

Fakt 6.1. *Dla każdej permutacji danych $\pi \in G$ i dowolnej klasy abstrakcji $K = [w]_{\sim_{L_1}}$ obraz K przy π również jest klasą abstrakcji.*

Wobec powyższego, zbiór $A^*/\sim_L$ jest w istocie G-zbiorem — grupa permutacji danych działa na nim, mapując klasy abstrakcji na klasy abstrakcji. Dodatkowo zauważmy, że przy tym działaniu $\{\epsilon\}$ i L_1 są niezmiennicze, zaś pozostałe klasy wymienione w punkcie 2 stanowią jedną orbitę tego działania — każda z nich może być przekształcona w każdą inną. Stąd intuicja, że poprawne sformułowanie twierdzenia Myhilla-Nerodego dla języków z danymi powinno dopuszczać nieskończenie wiele klas abstrakcji w $A^*/\sim_L$ a za to wymagać, by zbiór ten miał skończenie wiele orbit.

Nim jednak sformułujemy twierdzenie wprowadźmy kilka definicji pozwalających patrzeć bardziej abstrakcyjnie na języki z danymi.

6.1 Automaty nominalne

Zacznijmy od uogólnienia pojęcia alfabetu z danymi. Do tej pory był to zbiór postaci $A \times \mathbb{D}$ dla skończonego alfabetu A. Jednak można sobie wyobrazić kilka równie dobrych alfabetów które jednak nie są tej postaci. Na przykład:

- $\mathbb{D}^2 \cup \mathbb{D}$ — pary danych lub pojedyncze dane (suma rozłączna),
- $\mathbb{D} \cup A$ — dane lub litery bez danych,
- $\binom{\mathbb{D}}{2}$ — dwuelementowe zbiory danych.

Oczywiście dla każdego konkretnego przypadku można tak rozszerzyć definicję automatu z rejestrami, by mógł wczytywać litery danego alfabetu. Jednak brak jest wspólnej metody działającej dla wszystkich alfabetów jednocześnie. Dlatego też od teraz przyjmiemy następującą definicję.

Definicja 6.1. *Alfabet nominalny (oznaczany po prostu A) to dowolny skończenie orbitowy zbiór nominalny.*

Oczywiście $A \times \mathbb{D}$ jest w szczególności alfabetem nominalnym, ma tyle orbit ile jest elementów zbioru A.

By móc zdefiniować automat pracujący nad tak abstrakcyjnie określonym alfabetem potrzebujemy jeszcze jednego technicznego pojęcia.

Twierdzenie 6.2. *Dla dowolnego zbioru nominalnego Y i jego podzbioru $X \subseteq Y$ następujące warunki są równoważne:*

1. *X jest sumą orbit w Y,*
2. *$X = X\pi$ dla każdej permutacji $\pi \in G$,*
3. *indykator X jest funkcją ekwiwariantną $Y \to \{0, 1\}$,*
4. *zanurzenie $i_X \colon X \to Y$ jest funkcją ekwiwariantną.*

Dowód tego twierdzenia pozostawimy jako proste ćwiczenie.

Definicja 6.2. *Dla zbioru nominalnego Y, powiemy, że $X \subseteq Y$ jest jego podzbiorem ekwiwariantnym jeśli spełnia którykolwiek z warunków w powyższym twierdzeniu.*

Korzystając z powyższej definicji możemy zdefiniować automat nominalny.

Definicja 6.3. *Nominalny niedeterministyczny automata to krotka $\langle A, Q, \delta, q_I, F\rangle$ gdzie*

- *A to alfabet nominalny,*
- *Q to zbiór nominalny* — zbiór konfiguracji automatu,
- *δ to ekwiwariantny podzbiór $Q \times A \times Q$* — relacja przejścia,
- *q_I to element Q taki że $\{q_I\} \subseteq Q$ jest podzbiorem ekwiwariantnym* — konfiguracja początkowa,
- *$F \subseteq Q$ to podzbiór ekwiwariantny zbióru konfiguracji* — konfiguracje końcowe.

Jeżeli Q jest skończenie orbitowy, to automat nazywamy skończonym. Jeżeli δ jest funkcją z $Q \times A \to Q$, to automat nazwiemy deterministycznym.

Zauważmy, że przypadku automatu deterministycznego, ekwiwariantność funkcji przejścia sprowadza się do wymagania by następujący diagram komutował.

$$\begin{array}{ccc} Q \times A & \xrightarrow{\delta} & Q \\ {\scriptstyle\pi}\downarrow & & \downarrow{\scriptstyle\pi} \\ Q \times A & \xrightarrow{\delta} & Q \end{array}$$

Definicja 6.4. *Dla danego automatu nominalnego $\mathcal{A}$ język rozpoznawany przez ten automat definiujemy jako*

$$L(\mathcal{A}) \;=\; \{w \in A^* : \delta^*(q_I, w) \in F\}\,.$$

6.2 Twierdzenie Myhilla-Nerodego dla alfabetów nominalnych

Mamy już teraz komplet definicji potrzebnych do sformułowania twierdzenia Myhilla-Nerodego w kontekście słów z danymi.

Twierdzenie 6.3. *Dla dowolnego alfabetu nominalnego A i języka z danymi $L \subseteq A^*$, jeśli L jest ekwiwariantnym podzbiorem A^* to następujące warunki są równoważne:*

a) *$A^*/\sim_L$ ma skończenie wiele orbit (jako G-zbiór),*

b) *L jest rozpoznawany przez deterministyczny skończony automat nominalny.*

Dodatkowo jeśli alfabet A jest postaci $A_{fin} \times \mathbb{D}$ to równoważny powyższym jest warunek

c) *L jest rozpoznawany deterministycznym automatem rejestrowym.*

Dowód powyższego twierdzenia jest podany w ramach kolejnego rozdziału. Wcześniej podamy kilka komentarzy i przykładów ilustrujących jego sformułowanie.

Fakt 6.2. *Założenie że L jest ekwiwariantnym podzbiorem A^* jest wymagane, bez niego zbiór $A^*/\sim_L$ nie jest G-zbiorem — permutacja może mapować klasę abstrakcji na zbiór który nie jest klasą abstrakcji.*

6.3 Przykłady

Rozważmy teraz strukturę konfiguracji najbardziej naturalnego automatu rozpoznającego L_1. Automat taki ma stany $P = \{p_I, p, t\}$ i jeden rejestr $R = \{r\}$. W stanach p_I, t rejestr ma wartość nieokreśloną natomiast w p jego wartość jest określona. W związku z tym zbiór konfiguracji automatu jest sumą następujących trzech zbiorów:

1. $(p_I, \bot)$ — konfiguracja początkowa,
2. (p, d) dla $d \in \mathbb{D}$ — możliwe konfiguracje po wczytaniu słowa postaci $d\,(\mathbb{D} \setminus \{d\})^*$,
3. $(t, \bot)$ — konfiguracja końcowa po wczytaniu słowa z języka L_1.

Jak łatwo stwierdzić, ma miejsce następujący fakt.

Fakt 6.3. *Powyższy zbiór konfiguracji jest izomorficzny (jako G-zbiór) zbiorowi klas abstrakcji $A^*/\sim_L$.*

Czyli w tym przypadku udało nam się znaleźć automat rejestrowy którego zbiór konfiguracji odpowiada bezpośrednio klasom abstrakcji $A^*/\sim_L$. Spróbujmy teraz spojrzeć na ten automat jako automat nominalny.

Definicja 6.5. *Niech $\mathcal{A}_1$ będzie automatem nominalnym pracującym nad alfabetem $\mathbb{D}$. Niech $Q = \{p_I, t\} \cup \mathbb{D}$ będzie skończenie orbitowym zbiorem jego konfiguracji. Rozważmy konfigurację początkową $p_I \in Q$ i końcową $F = \{t\} \subseteq Q$. Oczywiście podany wybór konfiguracji jest ekwiwariantny. Zdefiniujmy teraz funkcję przejścia δ. Niech:*

- $\delta(p_I, d) = d$,
- $\delta(a, d) = t \;\; dla \; a = d$,
- $\delta(a, d) = a \;\; dla \; a \neq d$,
- $\delta(t, d) = t$.

W powyższych równaniach $a \in \mathbb{D} \subseteq Q$ oznacza konfigurację automatu zaś $d \in \mathbb{D}$ to kolejna dana w słowie wejściowym. Jak łatwo sprawdzić tak określona funkcja δ jest ekwiwariantna.

Powyższy automat nominalny jest w istocie podanym wcześniej automatem rejestrowym. Różnica jest jedynie taka, że zamiast mówić odrębnie o stanach automatu i wartości rejestrów, mówimy o konfiguracjach. Oczywiście biegi powyższego automatu odpowiadają biegom automatu rejestrowego i z definicji $L(\mathcal{A}_1) = L_1$.

Przejdźmy teraz do języka L_2. Zacznijmy od stworzenia *jak najprostszego* automatu dla tego języka. Automat taki ma stany $P = \{p_I, p_1, p_2, t, f\}$, stan początkowy to p_I zaś końcowy t. Automat dysponuje dwoma rejestrami $R = \{r_1, r_2\}$, przy czym:

- w stanach p_I, t, f automat nie używa rejestrów,
- w stanie p_1 automat używa jedynie rejestru r_1,
- w stanie p_2 automat używa rejestrów r_1, r_2.

W związku z tym możliwe konfiguracje tego automatu są następujące:

1. $(p_I, \bot, \bot), (t, \bot, \bot), (f, \bot, \bot)$ — trzy konfiguracje bez rejestrów, odpowiednio początkowa, akceptująca, odrzucająca,
2. $(p_1, d, \bot)$ dla $d \in \mathbb{D}$ — konfiguracje po wczytaniu jednej litery słowa,
3. (p_2, d, e) dla $d \neq e \in \mathbb{D}$ — konfiguracje po wczytaniu dwóch (różnych) liter słowa.

W szczególności automat może znaleźć się w konfiguracji $(p_2, 1, 2)$ po wczytaniu słowa 12 zaś wczytawszy słowo 21 osiąga konfigurację $(p_2, 2, 1)$ — są to dwie różne konfiguracje! A przecież te dwa słowa są równoważne: $12 \sim_{L_2} 21$. Czyli pomimo starań by uzyskać automat w jakimś sensie minimalny nie udało nam się: istnieją dwa równoważne słowa prowadzące do różnych konfiguracji automatu. Intuicyjnie nie da się tego uniknąć, gdyż rejestry automatu są zawsze uporządkowane zaś orbita klasy abstrakcji $[12]_{\sim_{L_2}}$ jest izomorficzna ze zbiorem nieuporządkowanych par $\binom{\mathbb{D}}{2}$.

Rozwiązaniem powyższego problemu są właśnie automaty nominalne. Ich zbiór konfiguracji może być dowolnym skończenie orbitowym zbiorem nominalnym czyli na przykład może zawierać $\binom{\mathbb{D}}{2}$ jako podzbiór. W szczególności minimalny automat nominalny dla L_2 ma następujący zbiór konfiguracji:

$$Q \;=\; \{p_I, t, f\} \cup \mathbb{D} \cup \binom{\mathbb{D}}{2}.$$

Konfiguracja początkowa to p_I a końcowa $F = \{t\} \subseteq Q$. Funkcja przejścia działa w naturalny sposób, między innymi zawiera ona następującą regułę:

$$\delta(a, d) = \{a, d\} \in Q \text{ gdy } a \neq d.$$

7 Wykład VII

In this lecture, we prove Theorem 6.3. As usual, Myhill-Nerode equivalence $\sim_L$ is a congruence with respect to a appending letters, so one can define a transition function on equivalence classes

$$\delta_L : \quad A^*/\sim_L \quad \times \quad A \quad \to \quad A^*/\sim_L$$

such that the following property is satisfied

$$\delta_L([w]_{\sim_L}, a) = [w \cdot a]_{\sim_L} \tag{1}$$

We use the name *syntactic automaton of* L for the automaton whose states are equivalence classes of $\sim_L$, whose initial state is the equivalence class of the empty word, whose transition function is δ_L defined above, and where accepting states are those that are included in L.

Implication a) to b) The following lemma establishes implication a) to b) in Theorem 6.3.

Lemat 7.1. *Suppose that $L \subseteq A^*$ is equivariant. Then the syntactic automaton is a nominal deterministic automaton.*

Dowód
Observe that we do not claim that the syntactic automaton is necessarily a nominal deterministic finite automaton.

We begin with the following observation, which relies on equivariance of L:

$$w \sim_L w' \quad \text{implies} \quad w \cdot \pi \sim_L w' \cdot \pi. \tag{2}$$

Indeed, to prove the above observation, suppose that $w \sim_L w'$. By unraveling the definition of $\sim_L$, we need to show that, for all $v \in A^*$, the following equivalence holds.

$$(w \cdot \pi) \cdot v \in L \iff (w' \cdot \pi) \cdot v \in L$$

By multiplying both sides by π^{-1}, the above is equivalent to

$$((w \cdot \pi) \cdot v) \cdot \pi^{-1} \in L \cdot \pi^{-1} \iff ((w' \cdot \pi) \cdot v) \cdot \pi^{-1} \in L \cdot \pi^{-1}$$

By equivariance of L, the above is equivalent to

$$((w \cdot \pi) \cdot v) \cdot \pi^{-1} \in L \iff ((w' \cdot \pi) \cdot v) \cdot \pi^{-1} \in L$$

By equivariance of concatenation in A^*, the above is equivalent to

$$w \cdot (v \cdot \pi^{-1}) \in L \iff w' \cdot (v \cdot \pi^{-1}) \in L$$

The above is implied by $w \sim_L w'$, which completes the proof of (2). By (2), one can define an action of G on equivalence classes of $\sim_L$ by

$$[w]_{\sim_L} \cdot \pi = [w \cdot \pi]_{\sim_L}. \tag{3}$$

So far, we have defined the structure of a G-set on the state space of the syntactic automaton. Why is this G-set nominal? By (3), the function $w \mapsto [w]_{\sim_L}$ is an equivariant function from the nominal set A^* to the state space. The image of a nominal set under an equivariant function is also a nominal set.

To complete the proof of the lemma, we need to show that the various components of the syntactic automaton are equivariant. It is easy to see that the initial state is a singleton orbit

$$[\epsilon]_{\sim_L} \cdot \pi \overset{(3)}{=} [\epsilon \cdot \pi]_{\sim_L} = [\epsilon]_{\sim_L}.$$

By equivariance of L, the set of final states is also equivariant:

$$[w]_{\sim_L} \in F \iff w \in L \iff w \cdot \pi \in L \iff [w \cdot \pi]_{\sim_L} \in F \iff [w]_{\sim_L} \cdot \pi \in F.$$

Finally, the transition function in the syntactic automaton is equivariant:

$$\delta_L([w]_{\sim_L}, a) \cdot \pi \overset{(1)}{=} [w \cdot a]_{\sim_L} \cdot \pi \overset{(3)}{=} [(w \cdot \pi) \cdot (a \cdot \pi)]_{\sim_L} \overset{(1)}{=} \delta_L([w]_{\sim_L} \cdot \pi, a \cdot \pi)$$

□

Implication b) to a) Suppose that we have two nominal DFAs

$$\mathcal{A} = (Q, A, q_I, F, \delta) \qquad \mathcal{A}' = (Q', A, q'_I, F', \delta')$$

over the same input alphabet A. An equivariant function

$$f : Q \to Q'$$

is called an automaton homomorphism if it maps q_I to q'_I, it maps F to F', and it maps δ to δ' in the following sense:

$$f(\delta(q, a)) = \delta'(f(q), a) \qquad \text{for every } q \in Q \text{ and } a \in A.$$

It is easy to see that if recognized languages are preserved under homomorphisms of automata.

The following lemma establishes implication a) to b) in Theorem 6.3.

Lemat 7.2. *The syntactic automaton is a homormorphic image of any automaton recognizing L*

Dowód
The implication from 1 to 2 follows from Lemma 7.1. The implication from 2 to 1 is a corollary of the "Furthermore" condition, so we only prove that the syntactic automaton of L is a homomorphic image of every nominal deterministic automaton recognizing L.

Consider any deterministic automaton recognizing L, of the form

$$\mathcal{A} = (Q, A, q_I, F, \delta).$$

Define the extended transition function

$$\delta^* : Q \times A^* \to Q$$

in the usual way. It is easy to see that the set of reachable states, namely

$$\delta^*(q_I, A^*) \subseteq Q$$

is an equivariant set. Therefore we can assume without loss of generality that all states in $\mathcal{A}$ are reachable. We will show that there is a homomorphism from $\mathcal{A}$ to the syntactic automaton of L. This homomorphism is defined by

$$f(\delta^*(q_I, w)) = [w]_{\sim_L}.$$

The definition is well formed, because

$$\delta^*(q_I, w) = \delta^*(q_I, w') \qquad \text{implies} \qquad w \sim_L w'.$$

It is also not difficult to see that the function f is equivariant, maps the initial state to the initial state, and maps accepting states to accepting states. □

7.1 Relationship with finite memory automata

Partial data tuples. Consider a set N of names. A partial data tuple over N is a partial function from N to $\mathbb{D}$. We write $(\mathbb{D} \cup \bot)^N$ for the set of partial data tuples. It is easy to see that this is an orbit-finite nominal set, as long as N is finite. An equality constraint over N is an element

$$(r, \tau, r') \in N \times \{=, \neq\} \times N$$

We say a partial tuple t satisfies the constraint if $t(r)$ is defined, and $t(r')$ is defined, and their data values are related by τ. For instance, the completely undefined tuple is the unique partial tuple that satisfies no constraints.

Lemat 7.3. *Every equivariant subset of $(\mathbb{D} \cup \bot)^R$ is equivalent to a boolean combination of equality constraints.*

Definicja 7.1. *A* nondeterministic finite memory automaton *consists of*

- *A finite set A_{fin} of input labels.*
- *A finite set C of control states.*
- *A finite set N of register names.*
- *Sets of initial $I \subseteq C$ and final $F \subseteq C$ control states.*
- *A transition relation, which is a subset of*

  $$\delta \subseteq C \times A_{\text{fin}} \times \mathit{bool}(\Phi) \times C$$

  where Φ is the set of equality constraints over the following set of names:

  $$N' = \quad \{\text{before}\} \times C \quad \cup \quad \{\text{input}\} \quad \cup \quad \{\text{after}\} \times C$$

Such an automaton is used to accept or reject words over the alphabet $A_{\text{fin}} \times \mathbb{D}$. The automaton works as follows. After reading a prefix of the input word, the configuration of the automaton consists of a control state from C together with a partial valuation from registers to data values. In other words, a configuration is an element of the set

$$C \times (\mathbb{D} \cup \bot)^R,$$

which we will call $Q_{\mathcal{A}}$. Suppose that the automaton is in a configuration

$$(c, d_1, \ldots, d_n) \in Q_{\mathcal{A}}$$

and that it reads an input letter $(a, d) \in A$. The automaton can nondeterministically choose any new configuration

$$(c', d'_1, \ldots, d'_n) \in Q_{\mathcal{A}}$$

provided that there is a transition

$$(c, a, \phi, c') \in \delta$$

such that the partial tuple

$$(d_1, \ldots, d_n, d, d'_1, \ldots, d'_n),$$

interpreted as a partial tuple over N', satisfies the boolean combination of equality constraints given by ϕ.

Lemat 7.4. *Consider an alphabet $A = A_{\text{fin}} \times \mathbb{D}$, where A_{fin} is a finite set. Then the following conditions are equivalent for every language $L \subseteq A^*$:*

1. *L is recognized by finite memory automaton.*
2. *L is recognized by a nominal nondeterministic finite automaton, where*
   - *The state space is $C \times (\mathbb{D} \cup \bot)^n$ for some finite set C and $n \in \mathbb{N}$.*
   - *There is only one initial state.*

Dowód
The implication from 1 to 2 follows immediately from the definition. For the converse implication, we use Lemma 7.3. The assumption on one initial state guarantees that the initial state is $(c, \bot^n)$ for some $c \in C$. □

Equivalence for nondeterministic automata In this section, we prove a stronger version of Lemma 7.4, namely:

Twierdzenie 7.1. *Consider an alphabet $A = A_{\text{fin}} \times \mathbb{D}$, where A_{fin} is a finite set. Then the following conditions are equivalent for every language $L \subseteq A^*$:*

1. *L is recognized by finite memory automaton.*
2. *L is recognized by a nominal nondeterministic finite automaton.*

The implication from 1 to 2 has already been shown in Lemma 7.4. We now show the implication from 2 to 1.

Define $\mathbb{D}^{(n)}$ to be the set of non-repeating n-tuples of data values.

Lemat 7.5. *Every nominal set X is an image, under an equivariant function, of a set of the form*

$$\coprod_{i \in I} \mathbb{D}^{(n_i)} \qquad \text{for some } \{n_i \in \mathbb{N}\}_{i \in I}$$

Dowód
It is enough to prove the lemma for a set X with one orbit. Choose any $x \in X$. Let $d_1, \ldots, d_n$ be any support of X. Because this is a support it follows that

$$\pi|_{\{d_1,\ldots,d_n\}} = \sigma|_{\{d_1,\ldots,d_n\}} \quad \Rightarrow \quad x \cdot \pi = x \cdot \sigma \qquad \text{for every } \pi, \sigma \in G.$$

Therefore, the relation defined by

$$f = \{(d_1, \ldots, d_n, x) \cdot \pi \in \mathbb{D}^{(n)} \times X : \pi \in G\}$$

is actually an equivariant function from $\mathbb{D}^{(n)}$ to X. The function is clearly surjective, because every element of X is of the form $x \cdot \pi$ for some $\pi \in G$. □

Wniosek 7.1. *Every orbit-finite nominal set is an image, under a partial equivariant function f, of a set of the form*

$$I \times (\mathbb{D} \cup \bot)^n \qquad \text{for some finite set } I \text{ and } n \in \mathbb{N}.$$

Dowód
Suppose that X is a nominal set with k orbits. By Lemma 7.5, X is an image of

$$\coprod_{i \in I} \mathbb{D}^{(n_i)} \tag{4}$$

where I is finite. Let n be the maximal number from $\{n_i\}_{i \in I}$. It is not difficult to see that $\mathbb{D}^{(n_i)}$ is isomorphic to an orbit of $(\mathbb{D} \cup \bot)^n$. It follows that the disjoint union from (4) is isomorphic to an equivariant subset of $I \times (\mathbb{D} \cup \bot)^n$. $\square$

We are now ready to prove Theorem 7.1. Consider a nominal nondeterministic finite automaton

$$\mathcal{A} = (Q, A, I, F, \delta)$$

in the special case when $A = A_{\text{fin}} \times \delta$ for some finite set A_{fin}. We assume that there is only one initial state, call it q_I. Otherwise, we a new initial state, call it q_I, with a trivial action

$$q_I \cdot \pi = q_I,$$

and extend the set of transitions by the equivariant set triples of the form

$$\{(q_I, a, q) : (p, a, q) \in I \text{ for some } p \in I\}.$$

Apply Corollary 7.1 to Q, yielding a partial surjective equivariant function

$$f : \quad C \times (\mathbb{D} \cup \bot)^n \quad \to \quad Q$$

for some finite set Q and $n \in \mathbb{N}$. Because there is just one initial state, we may assume that

$$q_I = f^{-1}(c_I, \bot^n)$$

for some $c_I \in C$. Define an nominal nondeterministic automaton, call it $f^{-1}(\mathcal{A})$, with states $C \times (\mathbb{D} \cup \bot)^n$, initial state $f^{-1}(q_I)$, final states $f^{-1}(F)$ and transitions $f^{-1}(\delta)$. It is easy to see that the automata $\mathcal{A}$ and $f^{-1}(\mathcal{A})$ recognize the same language.

Equivalence for deterministic automata In this section, we prove a deterministic variant of Theorem 7.1:

Twierdzenie 7.2. *Consider an alphabet $A = A_{\text{fin}} \times \mathbb{D}$, where A_{fin} is a finite set. Then the following conditions are equivalent for every language $L \subseteq A^*$:*

1. *L is recognized by deterministic finite memory automaton.*
2. *L is recognized by a nominal deterministic finite automaton.*

This theorem completes also the proof of Theorem 6.3.

We do the same proof as for the nondeterministic version. The only problem is that $f^{-1}(\delta)$ might not in general be deterministic. To solve this problem, we need two additional results.

The first additional result is the observation that in Corollary 7.1, the function f is not just equivariant, but it also preserves least supports. (In general, an equivariant function might decrease the least support.) The second additional result is stated in Lemma 7.6.

Lemat 7.6. *Suppose that $f : X \to X'$ is an equivariant function that preserves supports. Then for every nominal set A, and every equivariant function*

$$\delta' : X' \times A \to X'$$

there exists a function

$$\delta : X \times A \to X$$

such that the following diagram commutes

$$\begin{array}{ccc} X \times A & \xrightarrow{\delta} & X \\ {\scriptstyle f \times Id_A}\downarrow & & \downarrow{\scriptstyle f} \\ X' \times A & \xrightarrow[\delta']{} & X' \end{array} \tag{5}$$

Dowód
Let $Y_1, \ldots, Y_n$ be the orbits of the set $X \times A$.

Consider some $i \in \{1, \ldots, n\}$. Pick a representative $(x_i, a_i) \in Y_i$. In the diagram (5), follow the $f \times \mathrm{Id}_A$ arrow, and then δ', yielding an element

$$x'_i = \delta'(f(x_i), a_i)$$

Because the above element is the result of applying two equivariant functions, the least support of x'_i is a subset of the least support of (x_i, a_i). Because the function f is surjective, there must be some $y_i \in X$ such that

$$f(y_i) = x'_i = \delta'(f(x_i), a_i)$$

Because the function f preserves supports, the least support of y_i is equal to the least support of x'_i, which is included in the least support of (x_i, a_i). It follows that there is an equivariant function

$$\delta_i : Y_i \to X \qquad \text{such that } \delta_i(x_i, a_i) = y_i.$$

Do the construction above for all orbits $Y_1, \ldots, Y_n$, yielding functions $\delta_1, \ldots, \delta_n$. Define δ to be the union of these functions. We now prove that the diagram 5 commutes.

Pick some $(x, a) \in X \times A$. Because

$$(x_1, a_1), \ldots, (x_n, a_n)$$

represent all orbits of $X \times A$, it follows that

$$(x, a) = (x \cdot \pi, a \cdot \pi) \qquad \text{for some } i \in \{1, \ldots, n\} \text{ and some } \pi \in G.$$

Following the down-right path in the diagram 5 from (x, a) yields

$$\delta'(f(x), a) = \delta'(f(x_i \cdot \pi), a_i \cdot \pi) = \delta'(f(x_i), a_i) \cdot \pi.$$

Following the right-down path in the diagram 5 from (x, a) yields

$$f(\delta(x, a)) = f(\delta_i(x), a)) = f(\delta_i(x_i \cdot \pi, a_i \cdot \pi)) = f(\delta_i(x_i, a_i)) \cdot \pi = f(y_i) \cdot \pi,$$

which means that the diagram commutes because $y_i = \delta'(f(x_i), a_i)$.

□

We now prove Theorem 7.2. Let X' be the state space of the nominal deterministic finite automaton from item 2, and let δ' be its transition function. Apply Corollary 7.1 to X', yielding a partial surjective equivariant function $f : X \to X'$ where

$$X = C \times (\mathbb{D} \cup \bot)^n.$$

Because f preserves least supports, we can apply Lemma 7.6, yielding a transition function $\delta : X \times A \to X$. The rest of the proof is the same as in Theorem 7.1.

8 Wykład VIII: Struktury(?)

Spisali: Michał Żak, Piotr Iwaniuk

Dotychczas zajmowaliśmy się zbiorami nominalnymi określonymi przez $(\mathbb{D}, G)$, gdzie G była grupą wszystkich permutacji na $\mathbb{D}$. Teraz zajmiemy się czymś ogólniejszym. Wybierzmy $(\mathbb{D}, G)$, gdzie $\mathbb{D}$ jest zbiorem, a G jest jakąś grupą bijekcji, niekoniecznie wszystkich.

$\mathbb{D}$	**G**
$\mathbb{N}$	wszystkie permutacje
$\mathbb{Q}$	wszystkie permutacje zachowujące porządek
$\mathbb{Z}$	$\{i \in \mathbb{Z}\}$
$\emptyset$	$\{1\}$
$\mathbb{Z} \times \mathbb{Z}$	$\mathbb{Z} \times \mathbb{Z}$
$\mathfrak{A}$ – graf	wszystkie automorfizmy $\mathfrak{A}$
$\mathcal{R}$ – graf losowy	wszystkie automorfizmy $\mathcal{R}$

Dla każdego uniwersum symetrii $(\mathbb{D}, G)$ możemy badać:

(a) G-zbiory,

(b) zbiory $(\mathbb{D}, G)$-nominalne.

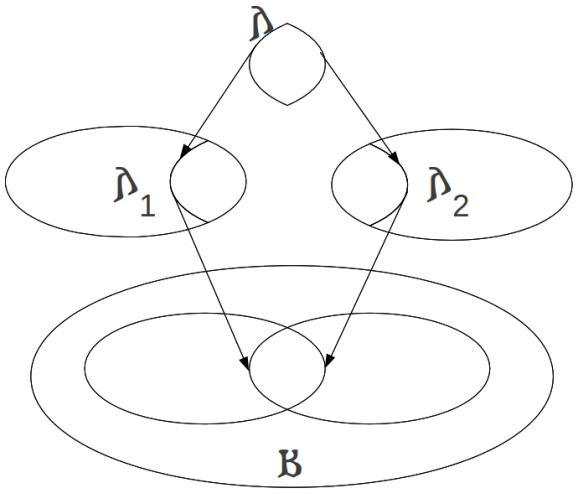
Definicja 8.1 (Zbiór $(\mathbb{D}, G)$-nominalny)**.** *Zbiór $(\mathbb{D}, G)$-nominalny to:*

- *X – dziedzina,*
- *działanie grupy G*

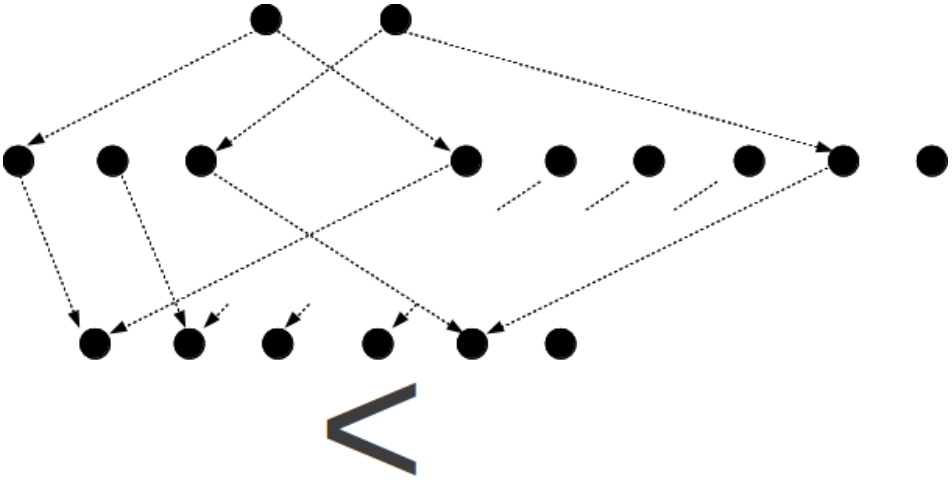
takie, że dla każdego $x \in X$ istnieje pewien $C \subseteq_{fin} \mathbb{D}$ taki, że:

$$\pi_{|C} = id_{|C} => x \cdot \pi = x.$$

Definicja 8.2 (Amalgamacja)**.** *Mówimy, że rodzina struktur $\mathcal{K}$ jest amalgamantna jeśli, dla dowolnych $\mathfrak{A}, \mathfrak{A}_1, \mathfrak{A}_2 \in \mathcal{K}$, oraz włożeń: $f_1 : \mathfrak{A} \to \mathfrak{A}_1$ i $f_2 : \mathfrak{A} \to \mathfrak{A}_2$ istnieje struktura $\mathfrak{B}$ oraz włożenia $g_1 : \mathfrak{A}_1 \to \mathfrak{B}$ i $g_2 : \mathfrak{A}_2 \to \mathfrak{B}$ takie, że $f_1; g_1 = f_2; g_2$*



Przykład 8.1. *Rodzina skończonych liniowych porządków jest amalgamantna*



Twierdzenie 8.1. *Jeśli $\mathcal{K}$ jest rodziną skończonych struktur relacyjnych taką, że*

- *$\mathcal{K}$ jest zamknięta na branie podzbiorów*
- *$\mathcal{K}$ jest amalgamantna*

to istnieje przeliczalna struktura uniwersalna dla $\mathcal{K}$

Twierdzenie 8.2. *Jeśli U_1, U_2 są przeliczalnymi strukturami uniwersalnymi dla $\mathcal{K}$ i $f : U_1 \to U_2$ jest skończonym częściowym izomorfizmem to, f rozszerza się do pełnego izomorfizmu.*

Wniosek 8.1. *Każdy częściowy automorfizm rozszerza się do pełnego automorfizmu.*

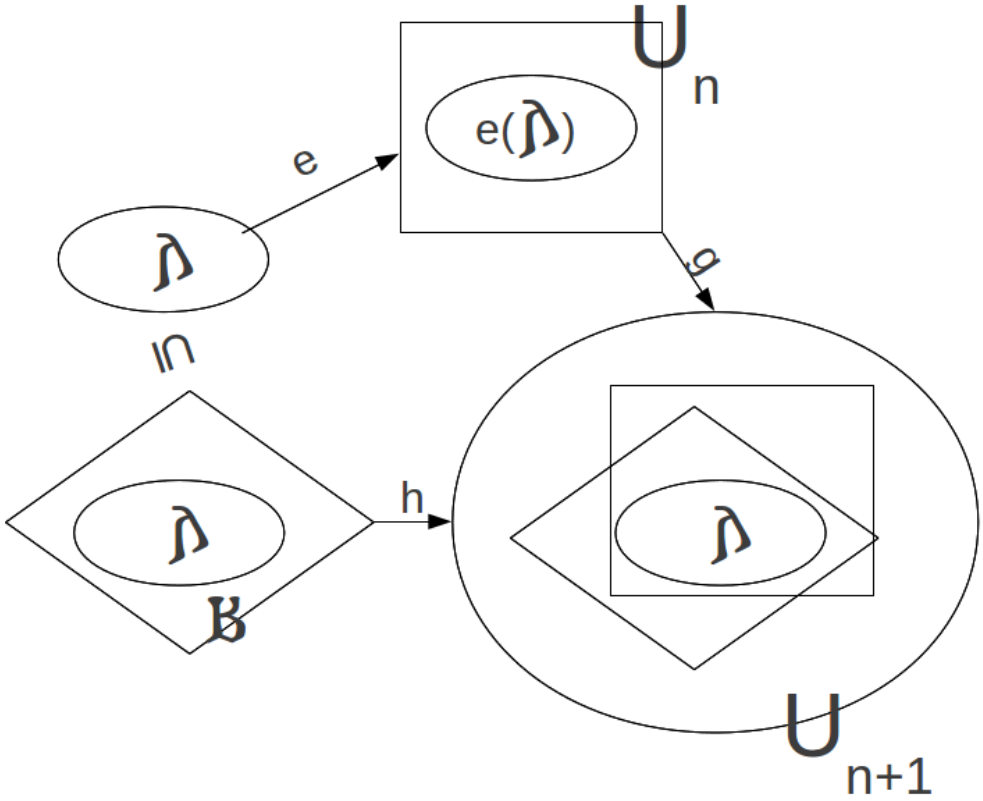
Wniosek 8.2. *Struktura uniwersalna dla $\mathcal{K}$ z twierdzenia 8.1 jest wyznaczona jednoznacznie z dokładnością do izomorofizmu.*

9 Wykład IX: ???

Dowód (Twierdzenie 8.1)
Ponieważ klasa $\mathcal{K}$ jest przeliczalna, możemy jej elementy ustawić w ciąg i przez $\mathcal{K}_n$ oznaczać będziemy zbiór n pierwszych elementów tego ciągu,
Skonstruujemy ciąg skończonych struktur:

$$U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots \in \mathcal{K}$$

taki, że dla dowolnych: $\mathfrak{A} \subseteq \mathfrak{B} \in \mathcal{K}_n$, dowolne włożenie $e : \mathfrak{A} \to U_n$ rozszerza się do $f : \mathfrak{B} \to U_{n+1}$
możemy przyjąć $U_0 = \emptyset$ dla dowolnego n rozważmy struktury U_n i $\mathfrak{A} \subseteq \mathfrak{B} \in \mathcal{K}_n$ oraz włożenie $e : \mathfrak{A} \to U_n$.



Ponieważ zawieranie jest szczególnym przypadkiem włożenia, na mocy amalgamancji klasy $\mathcal{K}$ istnieją włożenia g i h oraz struktura, którą oznaczymy przez U_{n+1} taka, że $h_{|\mathfrak{A}} = e; h$ przy czym bez straty ogólności włożenie g możemy zastąpić inkluzją.

Wystarczy pokazać, że

$$U = \bigcup_n U_n$$

spełnia warunki twierdzenia.

- każda struktura $\mathfrak{A} \in \mathcal{K}$ należy do pewnego $\mathcal{K}_n$ wobec tego istnieje włożenie $e_1\mathfrak{A} \in U_n$ dla pewnego n, więc i włożenie $e_2\mathfrak{A} \in U$
- drugi warunek jest oczywisty, ze względu na własność ciągu U

□ taką strukturę uniwersalną dla $\mathcal{K}$ będziemy nazywać ??kresem Fraïsségo

Dowód (Twierdzenie 8.2)

□

10 Wykład IX i połowa X

Spisał: Wojciech Czerwiński

10.1 Wprowadzenie

Niniejszy rozdział jest ostatnim dotyczącym własności zbiorów nominalnych. Następne będą już dotyczyły automatów alternujących oraz logiki FO (First Order), wówczas zbiory nominalne będą dla nas jedynie dobrym językiem do opisywania pewnych faktów.

Celem tego rozdziału jest uogólnienie Twierdzenia 5.2 i udowodnienie go. Przypomnijmy, że Twierdzenie 5.2 mówiło, że każdy jednoorbitowy zbiór nominalny jest izomorficzny z $\mathbb{D}^{(n)}$ mod S dla S będącej podgrupą S_n, czyli grupy permutacji $\{1, 2, \ldots, n\}$. To twierdzenie jednak dotyczyło sytuacji, gdy przyjmujemy, że grupa G to wszystkie permutacje zbioru danych $\mathbb{D}$, tak zwanej symetrii równościowej. Teraz uogólnimy twierdzenie także na niektóre inne symetrie, będziemy mieli do czynienia z $(\mathbb{D}, G)$, gdzie G spełnia pewne założenia, ale niekoniecznie jest grupą wszystkich permutacji $\mathbb{D}$.

Plan rozdziału jest następujący:

- będziemy badać własność najmniejszego wsparcia
- udowodnimy, że $(\mathbb{Q}, \text{mono})$, gdzie mono to wszystkie monotoniczne bijekcje $\mathbb{Q}$ spełnia własność najmniejszego wsparcia
- sformułujemy pojęcie uogólniające dotychczasowe $\mathbb{D}^{(n)}$ mod S
- sformułujemy i udowodnimy twierdzenie.

Przypomnijmy, że na zeszłym wykładzie pokazaliśmy, że jeśli klasa $\mathcal{K}$ skończonych struktur relacyjnych spełnia warunki:

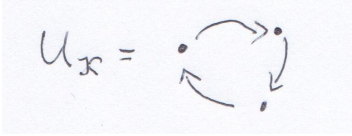
- jest zamknięta na podstruktury
- spełnia amalgamację

to istnieje uniwersalna przeliczalna struktura $\mathcal{U}$. $\mathcal{U}$ zależy od $\mathcal{K}$, więc będziemy pisać $\mathcal{U}_\mathcal{K}$. Będziemy się teraz zajmować sytuacją, gdy w $(\mathbb{D}, G)$ zbiór danych $\mathbb{D}$ to elementy $\mathcal{U}_\mathcal{K}$, a grupa G to wszystkie automorfizmy $\mathcal{U}_\mathcal{K}$.

10.2 Najmniejsze wsparcie

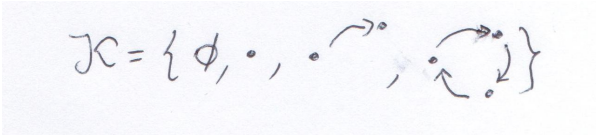
Jak wiemy (z definicji) w zbiorach nominalnych każdy element ma skończone wsparcie, nie zawsze jednak istnieje wsparcie najmniejsze. Przyjrzymy się teraz temu zagadnieniu w przypadku, gdy struktura uniwersalna to $\mathcal{U}_\mathcal{K}$, najpierw na przykładach.

Przykład 1. Rozważmy $\mathcal{U}_\mathcal{K} = \{(1,2), (2,3), (3,1)\}$. Poszukajmy najpierw $\mathcal{K}$ takiego, by $\mathcal{U}_\mathcal{K}$ pasowało do niego.



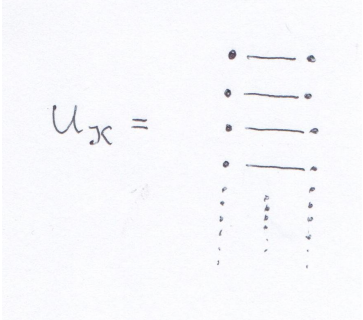
Odpowiedzią jest $\mathcal{K}$ będące wszystkimi skończonymi podstrukturami $\mathcal{U}_\mathcal{K}$ (taka odpowiedź nie zawsze będzie poprawna, ale zawsze będzie jedyną możliwą). Czyli $\mathcal{K} = \{\emptyset, 1 \text{ punkt}, \{(1,2), (2,1)\}, \{(1,2), (2,3), (3,1)\}\}$.

Zatem $(\mathbb{D}_{\mathcal{K}}, \mathcal{U}_{\mathcal{K}}) = (\mathbb{Z}_3, \mathbb{Z}_3)$. Rozważmy $X = \mathbb{Z}_3$. Wówczas aby f zachowywało X wystarczy aby f zachowywało 0, 1 lub 2. Czyli każde jednoelementowe wsparcie jest dobre, ale puste nie. Zatem w tej sytuacji nie ma najmniejszego wsparcia.



Przykład 2. Można byłoby myśleć, że powyższy przykład był patologiczny - bo skończony. Zobaczymy tutaj nieskończony przykład, w którym również nie istnieje najmniejsze wsparcie.

Niech $\mathcal{U}_{\mathcal{K}}$ to nieskończony przeliczalny zbiór par połączonych krawędzią. Wówczas $\mathcal{K}$ to zbiór skończonych



podstruktur $\mathcal{U}_{\mathcal{K}}$. Wsparcie konkretnej krawędzi to albo jej lewy koniec albo jej prawy koniec, puste nie jest dobre; czyli najmniejszego wsparcia nie ma.

Przykład 3. Rozważmy $\mathcal{K}$ będące skończonymi liniowymi porządkami. Wówczas struktura uniwersalna $\mathcal{U}_{\mathcal{K}} = \mathbb{Q}$. Mamy więc $(\mathbb{D}_{\mathcal{K}}, G_{\mathcal{K}}) \cong (\mathbb{Q}, G)$, gdzie G to wszystkie monotoniczne bijekcje $\mathbb{Q}$. Okazuje się, że $(\mathbb{Q}, G)$ ma własność najmniejszego wsparcia co będzie przedmiotem naszej uwagi przez następne 1/3 wykładu.

Twierdzenie 10.1. *$(\mathbb{Q}, G)$, gdzie G to wszystkie monotoniczne bijekcje $\mathbb{Q}$, posiada własność najmniejszego wsparcia.*

Dowód

Schemat dowodu jest analogiczny jak w dowodzie Twierdzenia 5.1, o istnieniu najmniejszego wsparcia w przypadku gdy G jest grupą wszystkich bijekcji.

Rozważmy zbiór nominalny X i jego element x. Wystarczy pokazać, że jeżeli $C \subseteq_{\text{fin}} \mathbb{D}$ jak i $E \subseteq_{\text{fin}} \mathbb{D}$ wspierają x, to jego wsparciem jest również $C \cap E$. Fakt ten zaś sprowadza się do następującego lematu:

Lemat 10.1. *Dla dowolnych $C, E \subseteq_{fin} \mathbb{D}$ zachodzi*

$$G_{C \cap E} \leqslant G_C + G_E,$$

gdzie przez G_S oznaczamy grupę permutacji stałych na zbiorze S, przez $G_1 + G_2$ najmniejszą grupę zawierającą G_1 i G_2 a przez $\leqslant$ relację podgrupy.

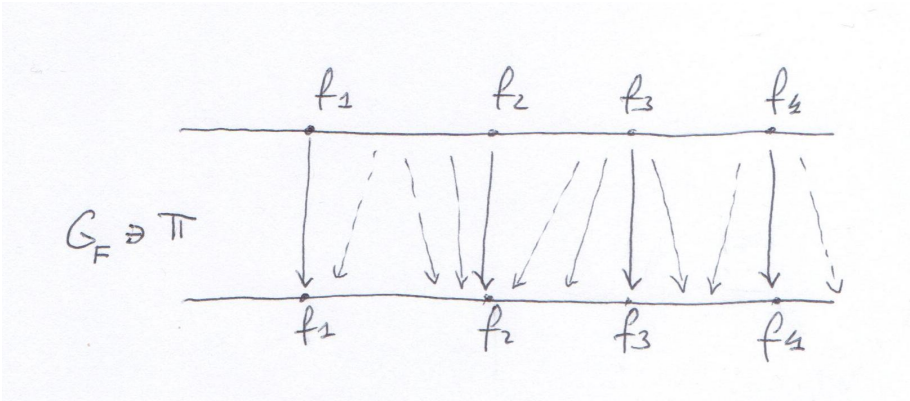
Załóżmy bowiem, że C i E są skończonymi wsparciami dla x. Niech $\pi \in G_{C\cap E}$. Wówczas z lematu mamy:

$$\pi = \sigma_1 \circ \ldots \circ \sigma_k,$$

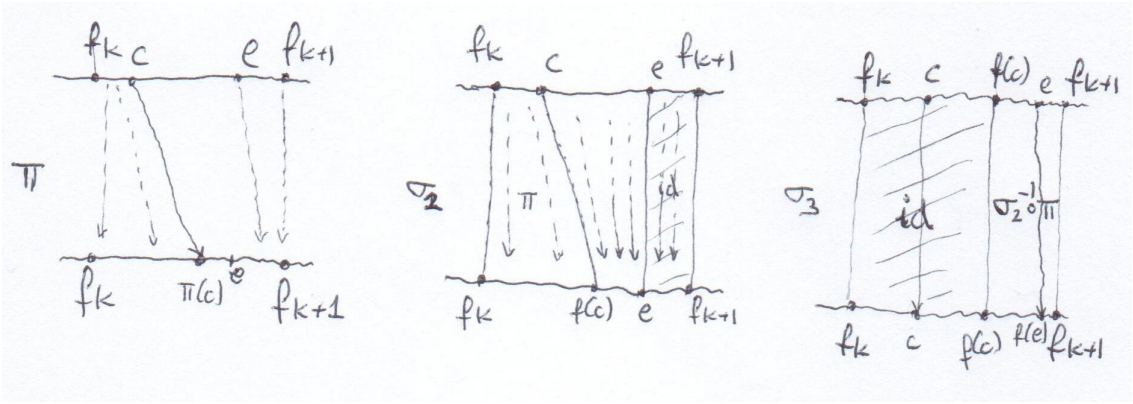
gdzie wszystkie $\sigma_i \in G_C \cup G_E$. Mamy więc $\sigma_i x = x$, a zatem $\pi x = x$, co dowodzi, że $C \cap E$ jest także wsparciem dla x. Aplikując ten lemat dla różnych skończonych wsparć elementu x znajdziemy najmniejsze z nich. Do wykazania twierdzenia pozostało zatem pokazanie lematu.

Podobnie jak w dowodzie Twierdzenia 5.1 ograniczymy się do udowodnienia przypadku, w którym C i E równią się bardzo niewiele: mianowicie $C = F \cup \{c\}$, $E = F \cup \{e\}$, gdzie $F = C \cap E$. Z tego szczególnego przypadku bardzo łatwo przez indukcję pokazać tezę lematu, nieprzekonanych odsyłam do dowodu Twierdzenia 5.1 (sama końcówka).

Niech $F = \{f_1, \ldots, f_n\}$, załóżmy bez straty ogólności, że $f_i < f_j$ dla $i < j$. Rozważmy $\pi \in G_F$, naszym celem jest znalezienie skończonego rozkładu na elementy z $G_C \cup G_E$. Ponieważ π jest monotoniczne i zachowuje F, to zachowuje również wszystkie przedziały $(-\infty, f_1)$, (f_i, f_{i+1}), (f_n, ∞) - może jedynie zamieniać elementy w ich obrębie.



Spójrzmy na elementy c i e. Jeżeli należą one do różnych przedziałów, powiedzmy $c \in (f_i, f_{i+1})$ oraz $e \in (f_j, f_{j+1})$, to nie ma problemu. Bierzemy wówczas σ_1 równe π wszędzie poza (f_i, f_{i+1}), a na tym przedziale stałe oraz σ_2 równe π na przedziale (f_i, f_{i+1}), a wszędzie poza nim stałe. Wówczas $\pi = \sigma_1 \circ \sigma_2$, $\sigma_1 \in G_C$, $\sigma_2 \in G_E$. Rzeczywisty problem pojawia się jedynie gdy c oraz e należą do tego samego przedziału.



Załóżmy więc bez straty ogólności, że $c, e \in (f_k, f_{k+1})$ (nieograniczone przedziały skrajne rozwiązuje się w ten sam sposób) oraz, że $c < e$. Niech σ_1 będzie równa π poza (f_k, f_{k+1}), a stała na tym przedziale,

$\sigma_1 \in G_C$, pozostaje problem uzyskania π na przedziale (f_k, f_{k+1}). Jeśli $f(c) < e$, to za σ_2 bierzemy następującą permutację: przedział (f_k, c) mapujemy tak jak π, przedział (c, e) mapujemy liniowo (może być dowolnie monotonicznie) na $(f(c), e)$, przedział (e, f_{k+1}) mapujemy identycznościowo. Wówczas σ_3 konstruujemy następująco: przedział (f_k, c) mapujemy identycznościowo, a przedział (c, f_{k+1}) mapujemy tak, by na przedziale (f_k, f_{k+1}) spełnione było $\sigma_2 \circ \sigma_3 = \pi$.

Mamy wówczas $\pi = \sigma_1 \circ \sigma_2 \circ \sigma_3$ oraz $\sigma_1, \sigma_3 \in G_C$, $\sigma_2 \in G_E$. Jeżeli zaś $f(c) \geqslant e$, to mamy $f(e) > e > c$ i robimy konstrukcję symetrycznie - tym samym wskazując szukany rozkład π na elementy z $G_C \cup G_E$. $\square$

10.3 Charakteryzacja zbiorów jednoorbitowych

Powracamy teraz do charakteryzacji zbiorów nominalnych jednoorbitowych, będziemy najpierw dochodzić do tego jak powinno wyglądać pojęcie analogiczne do $\mathbb{D}^{(n)} \mod S$, a potem sformułujemy i udowodnimy twierdzenie.

Odpowiednik $\mathbb{D}^{(n)}$ Rozważamy klasę struktur relacyjnych $\mathcal{K}$. Niech $\mathfrak{A} \in \mathcal{K}$. Zdefiniujmy

$$\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}} = \{e : \mathfrak{A} \to \mathbb{D}_{\mathcal{K}}, \text{e jest zanurzeniem}\},$$

gdzie przez zanurzenie mamy na myśli, że obraz $e(\mathfrak{A})$ jest izomorficzny z $\mathfrak{A}$. Zauważmy, że w przypadku symetrii równościowej $\mathcal{K}$ było po prostu zbiorem n elementów, więc wówczas $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}} = \mathbb{D}^{(n)}$. W ogólności zatem $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$ jest pojęciem uogólniającym $\mathbb{D}^{(n)}$ na dowolne symetrie. Zajmiemy się teraz poszukiwaniem operacji odpowiadającej $\mod S$.

Odpowiednik modulo S Przyjrzyjmy się najpierw $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$. Dla dowolnego automorfizmu danych $\pi \in G_{\mathcal{K}}$ definiujemy naturalnie działanie π na elemencie e jako złożenie funkcji e oraz π. Zauważmy, że przy tym działaniu wsparciem dla e jest $image(e)$, który jest skończony. Tym samym widzimy, że $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$ jest zbiorem nominalnym.

Rozważmy S będące podgrupą automorfizmów struktury $\mathfrak{A}$. Zdefiniujmy relację na $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$ następująco:

$$e \equiv_S e' \text{ wtw. } e = s \cdot e' \text{ dla pewnego } s \in S,$$

można prosto sprawdzić, że jest ona relacją równoważności.

Niech X będzie zbiorem nominalnym oraz $\equiv$ ekwiwariantną relacją równoważności. Niech

$$X/_{\equiv} = \{[x]_{\equiv} : x \in X\}.$$

Wówczas definiujemy działanie π na $[x]_{\equiv}$ jako $[x \cdot \pi]_{\equiv}$, jest ono dobrze zdefiniowane gdyż $\equiv$ jest ekwiwariantna. Przy tak zdefiniowanym działaniu nietrudno zauważyć, że również funkcja $X \mapsto X/_{\equiv}$ jest również ekwiwariantna.

Zatem $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}/_{\equiv_S}$ jest również zbiorem nominalnym jako ekwiwariatny obraz zbioru nominalnego. Zauważmy też, że $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$ ma dokładnie jedną orbitę, gdyż dla dowolnych $e, e' \in \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$ istnieje π takie, że $e = e' \cdot \pi$ (wynika to z własności struktury uniwersalnej). Obraz zbioru nominalnego jednoorbitowego jest również jednoorbitowy, więc także $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}/_{\equiv_S}$ jest jednoorbitowy. Jest on zatem idealnym kandydatem na uogólnienie $\mathbb{D}^{(n)} \mod S$.

Twierdzenie 10.2. *W symetriach, w których spełnione jest założenie istnienia najmniejszego wsparcia każdy jednoorbitowy zbiór nominalny jest izomorficzny z*

$$\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}/_{\equiv_S}$$

dla pewnego $\mathfrak{A} \in \mathcal{K}$ i pewnej podgrupy S grupy automorfizmów $\mathfrak{A}$.

Dowód
Poczynimy najpierw pomocniczą obserwację. Niech $f : X \to Y$ będzie ekwiwariantną funkcją „na". Wówczas naturalnie zdefiniowana relacja *kernel* f jest ekwiwariantna i co więcej $X/_{kernel\ f}$ jest izomorficzne z Y.

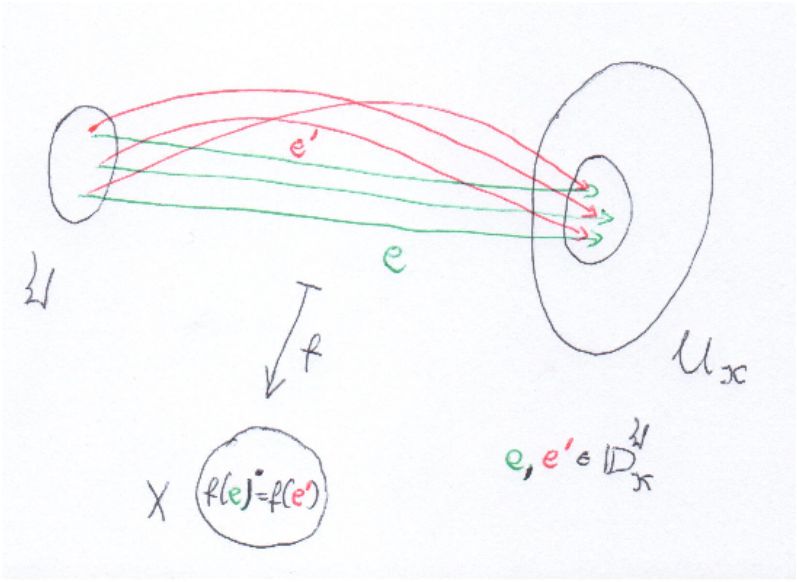
Rozważmy jednoorbitowy zbiór nominalny X oraz pewien $x \in X$. Mamy $X = x \cdot G_{\mathcal{K}}$. Niech $C \subseteq \mathbb{D}_{\mathcal{K}}$ będzie najmniejszym wsparciem x oraz niech $\mathfrak{A}$ będzie podstrukturą $\mathcal{U}_{\mathcal{K}}$ indukowaną przez zbiór C.

Rozważmy funkcję $f : \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}} \to X$. Pozostaje określić $f(e)$ dla danego $e \in \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$. Mamy $e : C \to \mathbb{D}_{\mathcal{K}}$, rozszerza się on do przynajmniej jednego automorfizmu $\pi : \mathbb{D}_{\mathcal{K}} \to \mathbb{D}_{\mathcal{K}}$, może jednak istnieć również potencjalnie inny $\sigma : \mathbb{D}_{\mathcal{K}} \to \mathbb{D}_{\mathcal{K}}$ rozszerzający e. Określimy $f(e) = x \cdot \pi$. Aby uzasadnić poprawność konstrukcji należy uzasadnić, że $x \cdot \pi = x \cdot \sigma$. To jednak jest prawdą, gdyż π oraz σ są równe na C wspierającym element x.

Mamy więc ekwiwariantną funkcję „na" $f : \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}} \to X$, czyli X jest izomorficzne z $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}/_{kernel\ f}$. Pozostaje więc do pokazania jedynie to, że *kernel* f jest równe $\equiv_S$ dla pewnej podgrupy S grupy automorfizmów $\mathfrak{A}$. Inaczej mówiąc szukamy grupy S takiej, że: $f(e) = f(e')$ wtw. $e = s \cdot e'$ dla pewnego $s \in S$.

Lemat 10.2. *Jeśli* $f(e) = f(e')$*, to* $image(e) = image(e')$*.*

Dowód
Wiemy, że $image(e)$ wspiera e, więc wspiera również $f(e)$. Analogicznie $image(e')$ wspiera $f(e') = f(e)$. Z istnienia najmniejszego wsparcia wnioskujemy, że $image(e) \cap image(e')$ wspiera $f(e) = f(e')$. Niech $|C| = n$, czyli również $|image(e)| = |image(e')| = n$. Jeśli $image(e) \neq image(e')$, to $|image(e) \cap image(e')| < n$, a jest wsparciem dla x; sprzeczność z założeniem, że C jest najmniejszym wsparciem dla x. □



Zdefiniujemy teraz S:

$$S = \{e \cdot (e')^{-1} : f(e) = f(e'),\ e, e' \in \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}\}.$$

Zauważmy, że skoro $f(e) = f(e')$, to z Lematu 10.2 $image(e) = image(e')$, czyli istotnie S jest podzbiorem automorfizmów $\mathfrak{A}$. Nim uzasadnimy, że jest to grupa sformułujmy lemat.

Lemat 10.3. *Niech* $s \in S$*. Następujące warunki są równoważne:*

- $f(e) = f(s \cdot e)$ *dla pewnego* $e \in \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$
- $f(e) = f(s \cdot e)$ *dla wszystkich* $e \in \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$

Dowód
Implikacja z dołu w górę jest jasna, pokażemy przeciwną. Niech $f(e) = f(s \cdot e)$ dla pewnego $e \in \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$. Rozważmy pewne $e' \in \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$, chcemy pokazać dla niego analogiczną równość. Ponieważ $\mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$ jest jednoorbitowe, to $e' = e \cdot \pi$ dla pewnego π. Zatem:

$$f(e') = f(e \cdot \pi) = f(e) \cdot \pi = f(s \cdot e) \cdot \pi = f(s \cdot e \cdot \pi) = f(s \cdot e'),$$

co należało pokazać. Korzystamy tu z faktu, że f jest ekwiwariantna, czyli $f(e \cdot \pi) = f(e) \cdot \pi$ dla każdych e oraz π. □

Uzasadnijmy teraz, że S to grupa. W sposób oczywisty posiada jedynkę oraz odwrotność. Zamknięcie na złożenie wynika z Lematu 10.3. Jeżeli $s_1, s_2 \in S$, to $f(e) = f(s_i \cdot e)$ dla $i \in \{1, 2\}$ i dowolnego e, więc $f(e) = f(s_2 \cdot e) = f(s_1 \cdot s_2 \cdot e)$ dla dowolnego e.

Pokażmy teraz, że istotnie *kernel* f jest równe $\equiv_S$.

Fakt 10.1. *$f(e) = f(e')$ wtedy i tylko wtedy, gdy $e = s \cdot e'$ dla pewnego $s \in S$.*

Dowód
Najpierw implikacja w lewo. Skoro $s \in S$, to $e_0 = s \cdot e'_0$ dla pewnych $e_0, e'_0 \in \mathbb{D}^{\mathfrak{A}}_{\mathcal{K}}$ takich, że $f(e_0) = f(e'_0)$. A zatem $f(e'_0) = f(s \cdot e'_0)$, czyli z Lematu 10.3 także $f(e') = f(s \cdot e') = f(e)$.

Teraz implikacja w prawo. Mamy $f(e) = f(e') = f(e' \cdot e^{-1} \cdot e)$. Z Lematu 10.2 wiemy, że $image(e) = image(e')$, czyli $e' \cdot e^{-1}$ jest automorfizmem $\mathfrak{A}$, a więc należy do S, co dowodzi tezy. □

To kończy dowód Twierdzenia 10.2. □

11 Wykład XI

Spisał: Michał Skrzypczak

Naszym celem w ramach tego wykładu będzie uściślenie i udowodnienie twierdzenia o następującej postaci.

Twierdzenie 11.1. *Pod pewnymi założeniami niepustość jest rozstrzygalna dla alternujących automatów z danymi.*

Te dodatkowe założenia są następujące:

1. Ustalamy, że symetria to $(\mathbb{D}, G)$ gdzie G to grupa wszystkich permutacji $\mathbb{D}$ (tzw. symetria równościowa).
2. Rozważamy automaty z jednym rejestrem. Nominalnie można to wyrazić tak, że każdy stan automatu ma jedno-elementowy suport.
3. Automat nie może *zgadywać* danych, czyli relacja przejścia jest postaci $\delta\colon Q \times A \to \mathbb{P}_{\text{fin}}(Q)$.

Warunek trzeci można zilustrować następującymi przykładami:

- automat o przejściach $\delta(q, d) = \mathbb{D}$ zgaduje daną,
- automat o przejściach $\delta((d_1, d_2), d) = \{(d_1, d), (d_2, d), \epsilon\}$ nie zgaduje żadnej *nowej* danej.

Pierwsze z podanych założeń można nieco osłabić. Przy czym aktualnie nie jest znana dobra charakteryzacja symetrii dla których twierdzenie jest spełnione. Każde z dwóch pozostałych założeń jest konieczne.

Dalsza część tego rozdziału jest poświęcona dowodowi powyższego twierdzenia. Przyjmijmy, że $\mathcal{A}$ jest automatem alternującym spełniającym założenia twierdzenia.

11.1 Dealternacja

Zacznijmy od operacji nazywanej *dealternacją* — ujęcia modelu alternującego w sposób nie-growy. Przyjmijmy na potrzeby tego rozdziału następującą definicję.

Definicja 11.1. *Konfiguracją nazywamy skończony zbiór stanów automatu $\mathcal{A}$.*

Intuicyjnie, konfiguracja odpowiada zbiorowi stanów z których automat ma obowiązek akceptować. Konfiguracja początkowa x_0 to $\{q_I\}$, zaś konfiguracje końcowe to takie x_f które są zawarte w F — zbiorze stanów akceptujących automatu.

Następująca definicja zadaje pojęcie przejścia pomiędzy konfiguracjami.

Definicja 11.2. *Dla dwóch konfiguracji x, x' powiemy że istnieje pomiędzy nimi przejście po literze a (ozn. $x \xrightarrow{a} x'$) gdy:*

- *dla każdego stanu $q \in x \cap Q_\exists$ istnieje stan q' taki, że $q \xrightarrow{a} q'$ i $q' \in x'$,*
- *dla każdego stanu $q \in x \cap Q_\forall$ i każdego stanu q' dla którego $q \xrightarrow{a} q'$ zachodzi $q' \in x'$.*

Dodatkowo, przez $x \to x'$ oznaczać będziemy, że istnieje litera a dla której $x \xrightarrow{a} x'$.

Jak łatwo sprawdzić istnienie przejścia $x \xrightarrow{a} x'$ jest równoważne istnieniu strategii dla gracza $\exists$ która z każdego stanu w x gwarantuje dotarcie do któregoś stanu w x' przy wczytaniu litery a.

Fakt 11.1. *Słowo $a_1 a_2 \dots a_n$ jest akceptowane przez automat $\mathcal{A}$ wtw. istnieje ciąg konfiguracji $x_0, x_1, \dots, x_n$, taki że*

- $x_0 = \{q_I\}$,

- $x_n \subseteq F$,
- $x_{i-1} \xrightarrow{a_i} x_i$.

Ponieważ wprowadzone wcześniej definicje są dostatecznie abstrakcyjne (nie wnikają w strukturę danych jako takich), ma miejsce następujące spostrzeżenie.

Fakt 11.2. *Zbiór konfiguracji jest zbiorem nominalnym. Zdefiniowane powyżej relacje $\xrightarrow{a}$ i $\rightarrow$ na konfiguracjach są ekwiwariantne.*

Dzięki faktowi 11.1 niepustość automatu alternującego sprowadza się do znalezienia odpowiedniego *świadka*. Wysławia to następujący lemat.

Lemat 11.1. *Język rozpoznawany przez automat alternujący $\mathcal{A}$ jest niepusty wtedy i tylko wtedy gdy*

$$\{q_I\} \rightarrow^* x \subseteq F,$$

gdzie $\rightarrow^*$ to domknięcie przechodnie relacji $\rightarrow$.

11.2 Półalgorytmy

Algorytm rozstrzygający niepustość danego automatu alternującego $\mathcal{A}$ będzie miał następującą postać:

Wczytaj automat alternujący $\mathcal{A}$. Równolegle wykonuj dwa półalgorytmy:

- przeszukuj możliwe konfiguracje x dla których $\{q_I\} \rightarrow^* x$ i sprawdzaj czy $x \subseteq F$. Jeśli tak, zakończ pracę i odpowiedz *NIEPUSTY*.
- przeszukuj *świadków pustości*[1] i gdy znajdziesz jakiegoś, odpowiedz *PUSTY*.

Pozostaje więc jedynie pokazać, że istnieje adekwatne pojęcie *świadków pustości* o dobrych własnościach obliczeniowych:

- zbiór wszystkich możliwych świadków powinien być rekurencyjnie przeliczalny,
- problem czy dany świadek zaświadcza o pustości danego automatu powinien być rozstrzygalny.

Zacznijmy od definicji obiektu który zaświadcza o pustości danego automatu $\mathcal{A}$, jednak zbiór takich obiektów nie jest rekurencyjnie przeliczalny.

Definicja 11.3. *Zbiór konfiguracji $X \subseteq \mathbb{P}_{\text{fin}}(Q)$ jest* podziałem *jeśli*

1. *X jest zbiorem ekwiwariantnym,*
2. *$\{q_I\} \in X$,*
3. *$X \cap \mathbb{P}_{\text{fin}}(F) = \emptyset$ — X nie zawiera żadnej konfiguracji końcowej,*
4. *gdy $x \in X$ i $x \rightarrow y$ to $y \in X$,*
5. *X jest zamknięty na nadzbiory: jeśli $x \in X$ i $x \subseteq y$ to $y \in X$.*

Nieformalnie, zbiór X powinien zawierać te konfiguracje x z których nie da się dojść do żadnej konfiguracji akceptującej. W szczególności, wszystkie strzałki $\rightarrow$ z X powinny prowadzić do X. Zauważmy jednak, że dla $x \notin X$ może być tak, że $x \rightarrow y$ i $y \in X$.

Fakt 11.3. *Automat $\mathcal{A}$ rozpoznaje język pusty wtw. gdy istnieje podział X zbioru konfiguracji dla tego automatu.*

Jak zaznaczyliśmy wcześniej, zbiór wszystkich podziałów nie jest rekurencyjnie przeliczalny. W związku z tym będziemy szukać takiej reprezentacji podziałów na której da się operować algorytmicznie.

[1] To nie jest żadna sekta :)

11.3 Well-Quasi-Orders

Okazuje się, że by reprezentować podział X, wystarczy przechowywać skończenie wiele jego *minimalnych* elementów. Wynika to z faktu, że odpowiedni porządek na konfiguracjach należy do klasy tzw. well-quasi-orders.

Zanim formalnie zdefiniujemy wymienione pojęcia, wprowadźmy kilka pomocniczych definicji. Wpierw zauważmy, że zbiór wszystkich konfiguracji $\mathbb{P}_{\text{fin}}(Q)$ jest uporządkowany przez inkluzję $\subseteq$. Następujące definicje pozwolą nam łatwiej operować zbiorami konfiguracji.

Definicja 11.4. *Niech C będzie dowolnym zbiorem konfiguracji $C \subseteq \mathbb{P}_{\text{fin}}(Q)$. Przez* $\min(C)$ *oznaczam zbiór elementów minimalnych C, zaś $\uparrow C$ to zbiór*

$$\{x \in \mathbb{P}_{\text{fin}}(Q) : \ \exists_{y \in C}\ y \subseteq x\}.$$

Lemat 11.2. *Dla każdego podziału $X \subseteq \mathbb{P}_{\text{fin}}(Q)$ zachodzi równość*

$$X = \uparrow \min(X).$$

Dowód. Ponieważ $\mathbb{P}_{\text{fin}}(Q)$ jest dobrze ufundowany (nie ma nieskończonego zstępującego łańcucha), więc poniżej każdego elementu $x \in X$ istnieje jakiś element minimalny $x' \in \min(X)$. Więc $X \subseteq \uparrow \min(X)$. Ponieważ X jest zamknięty w górę, więc $X \supseteq \uparrow \min(X)$. □

Naszym celem jest pokazanie, że podział X można reprezentować za pomocą zbioru $\min(X)$. Pozostaje pokazać następujące fakty:

- zbiór $\min(X)$ jest zawsze skończenie-orbitowy,
- patrząc tylko na $\min(X)$ można sprawdzić czy X jest podziałem.

Ponieważ $\min(X)$ jest antyłańcuchem, więc pierwszy fakt wynika wprost z następującego lematu.

Lemat 11.3. *Gdy $Z \subseteq \mathbb{P}_{\text{fin}}(Q)$ jest ekwiwariantnym antyłańcuchem ze względu na inkluzję, to Z jest skończenie-orbitowy.*

Dowód tego lematu pozostawiamy na później.

Przejdźmy teraz do sprawdzenia własności podziału. Niech Z będzie skończenie-orbitowym zbiorem konfiguracji. Załóżmy, że $Z = \{z_1, z_2, \ldots, z_n\} \cdot G$. Niech $X = \uparrow Z$. Przytoczmy własności jakie musi spełniać X by być podziałem:

1. X jest zbiorem ekwiwariantnym,
2. $\{q_I\} \in X$,
3. $X \cap \mathbb{P}_{\text{fin}}(F) = \emptyset$ — X nie zawiera żadnej konfiguracji końcowej,
4. gdy $x \in X$ i $x \to y$ to $y \in X$,
5. X jest zamknięty na nadzbiory: jeśli $x \in X$ i $x \subseteq y$ to $y \in X$.

Poniższa lista mówi jak sprawdzać te własności w sposób efektywny na podstawie $(z_1, \ldots, z_n)$.

1. tak, X jest ekwiwariantny z definicji,
2. $\{q_I\} \in X$ wtw. gdy $\exists_i\ z_i = q_I$ — bo $\{q_I\}$ jest ekwiwariantny,
3. $X \cap \mathbb{P}_{\text{fin}}(F) = \emptyset$ wtw. $\forall_i\ z_i \notin \mathbb{P}_{\text{fin}}(F)$, bo F jest ekwiwariantny,
4. o tym zaraz...

5. tak, X jest zamknięty na nadzbiory z definicji.

Pozostaje pokazać jak na podstawie $(z_1, \ldots, z_n)$ sprawdzić czy zbiór X jest zamknięty ze względu na $\rightarrow$. Zauważmy, że następujące warunki są równoważne:

- dla każdych $x \in X, x \rightarrow y$ zachodzi $y \in X$,
- dla każdych $x \in Z, x \rightarrow_{\min} y$ zachodzi $y \in X$, gdzie $\rightarrow_{\min}$ oznacza przejście do minimalnej możliwej konfiguracji,
- dla każdego i oraz $z_i \rightarrow_{\min} y$ zachodzi $y \in X$,
- dla każdego i oraz $z_i \rightarrow_{\min} y$ istnieje takie j, że $y \in \uparrow z_j \cdot G$.

Zauważmy, że ostatnia z własności może być sprawdzona algorytmicznie, gdyż dla danego z_i jest skończenie wiele możliwych y takich, że $z_i \rightarrow_{\min} y$, a każde z_j i y to skończone zbiory konfiguracji.

Wobec tego algorytm poszukujący świadka, że $L(\mathcal{A}) = \emptyset$, może sprawdzać kolejne skończone zbiory konfiguracji $\{z_1, z_2, \ldots, z_n\}$ i na każdym z nich testować powyższe warunki. Jeśli język jest pusty to istnieje podział X i algorytm w końcu znajdzie $\min(X)$ i odpowie tak. Oczywiście jeśli algorytm znajdzie jakiegoś świadka $\{z_1, \ldots, z_n\}$ spełniającego powyższe warunki to $L(\mathcal{A}) = \emptyset$.

11.4 Kontrprzykład

W tym podrozdziale pokażemy, że lemat 11.3 jest fałszywy gdy dopuścimy automaty o dwóch rejestrach.

Fakt 11.4. *Zbiór konfiguracji $C = \{x_1, x_2, \ldots\} \cdot G$ gdzie $x_i = \{(1,2), (2,3), \ldots, (i-1, i), i, 1)\}$ jest ekwiwariantnym antyłańcuchem, ale nie jest skończenie-orbitowy.*

Dowód. Oczywiście zbiór ten z definicji jest ekwiwariantny. Dodatkowo, jest on antyłańcuchem, wynika to z definicji elementów x_i. Jednak C nie jest skończenie-orbitowy gdyż dla każdej pary $i \neq j$, zbiory $x_i \cdot G$ i $x_j \cdot G$ są różnymi orbitami. □

12 Wykład XII: Logika pierwszego rzędu

Spisał: Grzegorz Sobczak

Celem wykładu jest przedstawienie logiki pierwszego rzędu ($FO(<,\sim)$)nad nieskończonym alfabetem. Będziemy używali słów z danymi - $w \in (A_{fin} \times \mathbb{D})$.

Przykład 12.1. *Wszystkie pozycje o etykiecie a mają różną daną.*

$$\varphi = \forall_x \forall_y (a(x) \wedge a(y) \wedge x \neq y) \implies x \not\sim y$$

Przykład 12.2. *Dla etykiet a oraz b ($a \in A_{fin}$ oraz $b \in A_{fin}$)*

$$\psi = \forall_x a(x) \exists_y (b(y) \wedge x \sim y \wedge x < y)$$

Będziemy rozpatrywać funkcję $erase - data : (A_{fin} \times \mathbb{D})^* \to {A_{fin}}^*$, która będzie rzutowała słowo $w \in (A_{fin} \times \mathbb{D})^*$ na swoje etytkiety. Przykładowo dla $A_{fin} = \{a, b\}$ mamy

$$erase - data(L_\psi) = \epsilon + {A_{fin}}^* b,$$

gdzie ψ jest zdefiniowane tak jak w Przykładzie 12.2. Natomiast dla $\psi' = \forall_x a(x) \exists_y (b(y) \wedge x \sim y)$ mamy $erase - data(L_{\varphi \wedge \psi'})$ = takie słowa, że liczba wystąpień a jest mniejsze niż liczba wystąpień b, tj.

$$w \in erase - data(L_{\varphi \wedge \psi'}) \Leftrightarrow \sharp_a \leqslant \sharp_b$$

Twierdzenie 12.1. *Spełnialność jest nierozstrzygalna dla logiki pierwszego rzędu $FO(<\sim)$.*

Dowód będzie polegał na redukcji z problmu pustości Maszyny Turinga [można też redukować z maszyny z dwoma licznikami z zero-testem]. Maszynę Turinga można kodować w jako słowo z danymi.

Lemat 12.1. *Istnieje formuła $\varphi \in FO(\leqslant, \sim)$ taka że $w \models \varphi$ wtedy i tylko wtedy, gdy w jest kodowaniem biegu akceptującego.*

Mamy:

$$\varphi = \forall_x \forall_y x \sim y \implies (\forall_{x'} succ(x, y) \implies (\forall_{y'} succ(y, y') \implies x' \sim y'))$$

Powyższe można zapisać za pomocą 3 zmiennych

$$\forall_x \forall_y x \sim y \implies (\forall_{x'} succ(x, x') \implies (\forall_x succ(y, x) \implies x' \sim x))$$

Twierdzenie 12.2. *Spełnialność jest rozstrzygalna dla $FO_2(<, \sim, succ)$ (logiki pierwszego rzędu używającej dwóch zmiennych i predykatu succ).*

Ostrzeżenie. Raczej nie da się tego udowodnić bez predykatu *succ*, ponieważ

$$succ(x, y) = x < y \wedge \forall_z \neg (x < z < y).$$

Przykład 12.3. *Dla alfabetu $A_{fin} = \{inc, dec\}$ mamy*

$$\begin{aligned} \varphi &= \forall_x \forall_y (dec(x) \wedge dec(y) \wedge x \neq y) \implies x \sim y \\ \psi &= \forall_x dec(x) \exists_y (inc(y) \wedge x \sim y \wedge x < y) \end{aligned} \tag{6}$$

stosując funkcję $erase - data$ mamy

$$erase - data(L_{\varphi \wedge \psi}) = \textit{dla każdego sufiksu } \sharp_{inc} \geqslant \sharp_{dec}$$

Definicja 12.1 (Automat z wieloma licznikami)**.** *Automat taki słada się z:*

1. *zbioru stanów Q,*
2. *stanu początkowego q_I oraz stanu końcowego $F \in Q$,*
3. *alfabetu wejściowego A,*
4. *skończonego zbioru C liczników,*
5. *skończonego zbioru przejść delta*

$$\delta \subseteq Q \times A \times \{inc(c), dec(c) : c \in C\}^* \times Q$$

Automat ten akceptuje słowo gdy są spełnione następujące kryteria:

1. *stan końcowy*
2. *wszystkie liczniki równe 0*

Zbiór przejść można interpretować w następujący sposób: Automat jest w stanie $q \in Q$ i czyta literę a, wówczas wielokrotnie zwiększa (inc), bądź zmniejsza (dec) odpowiednie liczniki i przechodzi do stanu $q' \in Q$. Bieg jest sekwencją przejść takich, że każdy licznik jest większy lub równy 0 ($\forall_{c \in C} val(c) \geqslant 0$).

Przykładowo automat taki może rozpoznać następujący język:

$$a^{n_1} b^{m_1} \# a^{n_2} b^{m_2} \# ... \# a^{n_k} b^{m_k}$$

Języki rozpoznawane przez automat z wieloma licznikami są:

- zamknięte ze względu na sumę zbiorów
- zamknięte ze względu na przecięcie
- zawierają wszystkie języki regularne
- nie są zamknięte ze względu na dopełnienie
- zamknięte ze względu na operacje $shuffle$, $shuffle^*$

Operacja $shuffle$ można zdefiniować następująco:

$$shuffle(L, K) = \{w_1 v_1 w_2 v_2 ... w_n v_n : w_1 ... w_n \in L, v_1 ... v_n \in K\}$$

Na przykład $shuffle(a^*, b^*) = (a + b)^*$. $shuffle^*(L) = \bigcup shuffle(L, ..., L)$

Lemat 12.2. *Dla każdego automatu A z wieloma licznikami można policzyć formułę $\varphi \in FO_2(<, \sim, succ)$ (w czasie wielomianowym - PTIME) taki, że A jest niepusty, gdy φ jest spełnialna.*

Możemy założyć, że automat A, przy każdym przejściu, wykorzystuje conajwyżej 1 licznik (1 operację dec lub inc). Wówczas typowe przejścia to

- przejście z wczytaniem litery - $(p, a, inc(z), q)$
- przejście puste (bez litery) - $(p, \ , inc(z), q)$

Dowód twierdzenia 12.2. Dla każdej formuły $FO_2(<, \sim, succ)$ będziemy znajdować autmoat z wieloma licznikami A, taki że $erase - data(L_\varphi) = L_A$. Kolejne kroki w dowodzie:

1. Znormalizowanie φ
   (a) kombinacja boolowska z formuł $\forall\exists$ oraz $\forall\forall$

2. Wprowadzenie nowego modelu automatu dla słów z danymi
3. Przekształcenie znormalizowanej formuły w model

□

Przykład 12.4. *Jakaś dana występuje conajmniej 4 razy*

$$\exists_x(\exists_y y > x \wedge y \sim x \wedge (\exists_x x > y \wedge y \sim x \wedge (\exists_y y > x \wedge y \sim x)))$$

Wprowadzenie automatu z danymi A, $L_A \subseteq (A_{fin} \times \mathbb{D})^*$

- wejście: słowo z danymi
- alfabet wejściowy A_{fin}
- alfabet pomocniczy B
- $L_{global} \subseteq (A_{fin} \times B)^*$
- $L_{local} \subseteq (A_{fin} \times B)^*$

Automat ten akceptuje słowo w gdy:

1. istnieje słow $v \in B^*$
2. $|v| = |w|$
3. $erase - data(w \times v) \in L_{global}$
4. $\forall_{d \in \mathbb{D}} erase - data(\pi_d(w \times v)) \in L_{local}$, czyli dla każdej danej $d \in \mathbb{D}$ słowo $w' = \pi_d(w \times v)$ złożone z liter o danej d należy do L_{local}.

Przykładowo, załóżmy, że mamy słowo w:

$$w = \begin{array}{cccccccc} a & b & a & b & a & a & b \\ 1 & 3 & 1 & 2 & 1 & 2 & 1 \end{array}$$

Wówczas algorytm zgaduje niedeterministycznie jakieś słowo $v \in B$, np. dla $B = \{x, y\}$

$$v = xxyyxxy$$

Wówczas mamy:

$$w \times v = \begin{array}{ccccccc} a & b & a & b & a & a & b \\ 1 & 3 & 1 & 2 & 1 & 2 & 1 \\ x & x & y & y & x & x & y \end{array}$$

$$\pi_1(w \times v) = \begin{array}{cccc} a & a & a & b \\ 1 & 1 & 1 & 1 \\ x & y & x & y \end{array}$$

$$erase - data(\pi_1(w \times v)) = \begin{array}{cccc} a & a & a & b \\ x & y & x & y \end{array}$$

Przykład 12.5. *L_A = "Każda dana jest różna". L_{local} zawiera wszystkie słowa 1-literowe*