Języki formalne i złożoność obliczeniowa

---== ZAJĘCIA 1 ==---

ε – znak pusty

Prowadzący: Profesor Maciej Kandulski

Jeden wspólny EGZ pisemny – jedna ocena dla wykładu/labu/ćwiczeń

Na 3 wystarczy dobre poznanie pojęci i definicji. Obowiązuje nas tylko to co jest omawiane na slajdach/zajęciach. Egzamin będzie z całości materiału przedmiotu a więc i zadań wymagających obliczeń. Poziom zadań będzie podobny do przykładów omawianych na zajęciach.

Uwaga! Za głupie, nieprzemyślane odpowiedzi będą minusowe punkty.

IDEA WYKŁADU:

- Zajmujemy się SKOŃCZONYMI ciągami znaków.
- SŁOWO == Skończony ciąg znaków z alfabetu.
- Gramatykę języka sprawdza AUTOMAT (tak jak gramatykę języka programowania sprawdza kompilator)
- Ciągi się GENERUJE! (Tworzy)
- Maszyna Turinga
- Urządzenia formalne generujące słowa (ciągi znaków) to GRAMATYKA
- Urządzenia formalne które sprawdzi poprawność słów (ciągów znaków) to AUTOMAT
- Interesują nas takie gramatyki i automaty które będą stosunkowo często rozpoznawać, że ciągi znaków są OKEY.
- Przykładem automatu i jego wykorzystania jest znalezienie określonego ciągu znaków w tekście (CTRL + F).
- Będziemy tworzyć algorytmy w pseudokodzie. (Będziemy otrzymywać gotowe funkcje i procedury oraz tworzyć własne – które będą możliwe do wykorzystania w następnych zadaniach)
  PISANIE PSEUDOKODU!

Sprawdzenie, czy program jest okey następuje w czasie parsowania podczas wykonywania tzw. Analizy leksykalnej, która to podany ciąg znaków z wejścia dzieli na

TOKENY (podstawowe, atomowe, znaczące elementy języka) są podczas wykonywania tu analizy syntaktycznej, która ma za zadanie …

???Gramatyki generują automaty rozpoznane???

POJĘCIA:

- ***ALFABET*** – dowolny niepusty i skończony zbiór znaków atomowych(znaki nazywamy rzeczami) to znaczy takich w skład których nie wchodzą inne symbole z tego zbioru.
  Alfabet w literaturze oznacza się literami Σ, Σ_1 … lub V, V_1 … Na zajęciach będziemy stosować ten drugi zapis!
- ***SŁOWEM*** nad alfabetem V nazywamy dowolny skończony ciąg symboli alfabetu V.
- ***V**** - tak oznaczamy ZBIÓR WSZYTSKICH możliwych do utworzenia SŁÓW nad alfabetem V.
- Słowa będziemy najczęściej oznaczać literami P,Q,R,X,Y,Z być może z indeksami.
- **ε** - Do V* należy, zgodnie z definicją słowo składające się z ZERA symboli alfabetu V. Nazywamy je słowem pustym i oznaczamy je znakiem „ε".
  SPACJA NIE JEST SŁOWEM PUSTYM – spacja jest znakiem!

$$V^+ = V^* \setminus \{ \varepsilon \}$$

V^+ <- zbiór słów niepustych. (podzbiór słów alfabetu V)

Przykłady:

- $V_1 = \{ 0 , 1 \}$ to: ε, 0 , 1, 111, 00101010, 002 $\in V_1^*$

- V_2 = {alfabet j.pl} wówczas: klops, kloss, kos ,ąęó , amand $\in V_2^*$ (Uwaga: cały słownik j.pl jest podzbiorem V_2^*.

- V_3 będzie zbiorem wszystkich znaków z klawiatury wówczas każdy poprawnie zapisany program w języku programowania jest słowem nad V_3

- Niech V_4 będzie zbiorem słów ustalonego słownika j.pl. Słowa tego słownika traktujemy jako elementy niepodzielne (atomy). Wówczas niektóre słowa nad V_4 sa poprawnymi zdaniami języka polskiego. (choć oczywiście nie wszystkie) - po procesie tokenizacji następuje sprawdzanie „gramatyczne" np.: czy każdy rozpoczęty nawias ma odpowiadający mu nawias zamykający.

PODSTAWOWE OPERACJE NA SŁOWACH:

Słowa reprezentujemy jako stringi!

1) KONKATENACJA:

Dla dowolnych słów -> (P , Q $\in$ V*) konkatenację P i Q oznaczamy „ PQ ”
Konkatenacja zdefiniowana jest w następujący sposób:

i) Jeśli P = $a_1 \dots a_n$ i Q = $b_1 \dots b_m$ to PQ = $a_1 \dots a_n b_1 \dots b_m$
ii) Jeśli P = ε to PQ = Q // Jeśli Q = ε to PQ = P

ε – jest elementem neutralnym operacji konkatenacji słów

ε konkatenacja z ε = ε ε = ε (1*1=1)

Własności operacji konkatenacji:

1) Łączność: dla dowolnych słów P,Q i R zachodzi P(QR) = (PQ)R
2) Brak przemienności: NIE jest tak że dla dowolnych P,Q zachodzi PQ = QP
3) Brak elementu odwrotnego (-P) takiego że P(-P) = ε

2) PODSŁOWO:

Inaczej - podciąg. Niech P $\in$ V*. Słowo Q nazywamy podsłowem słowa P jeśli istnieją słowa Q1 i Q2 $\in$ V* takie, że P= Q_1QQ_2 , jeśli nadto Q_1 (Q_2) jest słowem pustym to Q nazywamy prefiksem (sufiksem) słowa P.

To że Q jest podsłowem słowa P oznaczamy symbolicznie Q $\sqsubset$ P

P $\sqsubset$ P || εPε = P || ε ε ε εa = a

3) DŁUGOŚĆ SŁOWA:

Długością słowa P $\in$ V* nazywamy liczbę naturalną |P| określaną indukcyjnie w następujący sposób:

i) | ε| = 0
ii) |Pa| = |P| + 1
iii) Długość słowa to po prostu liczba wystąpień symboli z V w tym słowie.

Wyzwanie 1: Ile jest podsłów w słowie „abcabca"

A,b,c -3

(Ab, bc,ca) -3 (Abc, bca, cab) – 3 (Abca, bcab, cabc) -3

(Abcab, bcabc, cabca) -3 (Abcabc, bcabca) -2

(Abcabca) -1 ε - 1

Odp: Liczba podsłów jest równa 19.

$|PQ| = |P| + |Q|$ -> mamy na pierwszy rzut oka oczywistą równość, ale dowodzimy ją indukcyjnie po długości słowa .

4) *POTĘGA SŁOWA: = Konkatenacja słowa N razy!*

Niech $P \in V^*$ i niech $n \geq 0$ n-tą potegą słowa P nazywamy słowo oznaczone P^n zdefiniowane indukcyjnie w następujący sposób:

i) $P^0 = \varepsilon$
ii) $P^{n+1} = P^N P$

Widać że:

1) Jeśli $P \in V$, to dla $n \geq 0$ mamy $P^n = aaa...a$, gdzie a wzięte jest n razy.
2) Dla dowolnego $P \in V^*$ mamy $P^1 = P$ $P^3 = PPP$ $P^0 = \varepsilon$
$P^1 = P^{0+1} = \varepsilon P = P$

5) *ODBICIE ZWIERCIADLANE SŁOWA*:

Niech $P \in V^*$, odbiciem zwierciadlanym słowa P (rewersem P) nazywamy słowo oznaczone P^{-1} zdefiniowane indukcyjnie w następujący sposób:

i) $\varepsilon^{-1} = \varepsilon$
ii) $(Pa)^{-1} = aP^{---1}$

Własności operacji odbicia zwierciadlanego:

1) $(PQ)^{-1} = Q^{-1} P^{-1}$ (indukcja po długości słowa Q)
2) $(P^n)^{-1} = (P^{-1})^n$
3) $(P^{-1})^{-1} = P$

Definicja algorytmu na odwrotność słowa ---->

$G \circ F = F^{-1} \circ G^{-1}$

Podstawowe operacje na JĘZYKACH:

1) ***Język nad alfabetem V:***
   a) Nazywamy dowolny podzbiór $L \subset V^*$ (JĘZYKI SĄ ZBIORAMI!)
   b) Ciągi znaków (słowa) $\in$ L -> to słowa które są ok
   c) Dopełnienie L – to błędny syf. (czyli nieakceptowalne ciągi znaków)
   d) Automat skończenie stanowy ->sprawdza czy ciąg znaków (słowo) należy do języka. Czy jest poprawne/akceptowalne(urządzenie formalne)
   e) Skoro języki są zbiorami (obiektami typu SET [zbiorem]) to można na nich wykonywać wszystkie operacje teoriomnogościowe. Takie jak:
      - $L_1 \cap L_2$ (tylko to co się powtarza) iloczyn zbiorów [i]
      - $L_1 \cup L_2$ (wszystkie unikalne słowa – suma zbiorów) – [lub]
      - $L_1 \setminus L_2$ (Wszystkie elementy z L1 których nie ma w L2) różnica
      - L_1^- (dopełnienie zbioru L1)

Wyzwanie 2: – Wypisz powyższe operacje na Językach wiedząc, że:

L1 = { ε , a , b , ab, bb] && L2 = { ε, b, ab, bbb}

Odp.: Dopełnienie: = { ε ,a , a^2, a^3… a^n : n>= 0}

Wyzwanie 3: – Wypisz powyższe operacje na Językach wiedząc, że:

L1 = $\{a^n b^m : n \geq m \geq 0\}$, L2 = $\{ a^n b^m : m \geq n \geq 0 \}$

L1 = np.: { ε , a , ab, a^{1000})

- $L_1 \cup L_2 = \{a^m b^n : n,m \geq 0\}$
- $L_1 \cap L_2 = \{a^n b^n : n,m \geq 0\}$
- $L_1 \setminus L_2 = \{a^n b^m : n > m \geq 0\}$

2) ***OPERACJA KONKATENACJI JĘZYKÓW:***
   - Konkatenacja wszystkich słów 1-szego języka z wszystkimi słowami 2-giego języka.
   - Konkatenacja języków $L_1, L_2 \in V^*$ oznaczamy L_1L_2 tak, że $L_1L_2 = \{ P_1P_2 : P_1 \in L_1 , P_2 \in L_2 \}$
   - Warto rysować tabelkę która pomoże wizualnie w konkatenacji języków.
   - Konkatenacja L_1L_2 ma co najmniej m*n elementów, $m \in L_1$ $n \in L_2$

Przykład 1:

L_1 = {ab , abaa , aab , aaaba} == { ab, a^2b. a^3ba }

L_2 = { b , aba }

L_1L_2 = { ab , aba^2 , a^2b , a^3ba }

Przykład 2:

L_1 = { a , ab }

L_2 = { ε , b , ab}

L_1L_2 = { a, ab , aab, abb , $(ab)^2$ }

L_1L_2 – ma co NAJWYŻEJ m * n elementów!

L_1 \\\\\\ L_2	ε	b	ab
a	a	~~ab~~	a^2b
ab	ab	ab^2	(ab)2

Przykład 3:

- $L_1 = \{ a^n : n \geq 0 \}$ $L_2 = \{ b^n : n \geq 0 \}$ UWAGA n to zmienna lokalna! Nie daj się nabrać!
  Odp: $L_1L_2 = \{ a^n b^m \ \ m,n \geq 0 \}$ Narysuj tabelkę!

3) ***OPERACJA POTĘGI JĘZKÓW:***

- $L \subset V^*$ $L_1L_2...L_3 = L^n$
- Definicja potęgi języka jest taka sama jak definicja potęgi słowa ale zamiast symboli słowa – używamy oczywiście symboli języka:
  i) $L^0 = \{ \varepsilon \}$
  ii) $L^{n+1} = L^nL$

  $L\emptyset = \emptyset$ (∅ - zbiór pusty działa jak zero! Coś * 0 = 0) – tutaj konkatenacja

  $L\{\varepsilon\} = L$ – tutaj konkatenacja

  ∅ - Język pusty ∅ == { }

  $L^1 = L^{0+1} = L^0L = \{ \varepsilon\}L = L$

Wyzwanie 4: $L = \{a , ab\}$ oblicz L^n dla n = 1,0,2,3

$L^0 = \{ \varepsilon\}$

$L^1 = L^{0+1} = L = \{a, ab\}$

$L^2 = LL = \{ aa , aab, aba , abb\}$

$L^3 = LLL = \{ a^3 , a^3b , a^2ba, aabab, aba^2, abaab, (ab)^3 \}$

Zobacz!: Widzimy, że potęgowanie języka może wprowadzać nowe elementy i może nie zawierać starych elementów.

Wyzwanie 5: $L = \{a^n : n \geq 0\}$ oblicz: L^n dla n = 0,1,2,3

UŻYJ TABELKI! – Zobacz!: po wyniku widać, że nie zawsze potęgowanie języka dodaje nowe elementy, czasem nie robi nic!

Wyzwanie 6: $L=\{ a^n : n > 0 \}$ Oblicz L^n dla n = 0,1,2,3

$P \in L_1L_2$ Zobacz!:„Nie zawsze potęgowanie języka dodaje nowe

= elementy, czasem je USUWA! I Jest ich coraz mniej!

/ \

P_1 P_2 $L = \{a, ab, abc\}$

| | Pewne P powstałe z L^3 „aababc"

L_1 L_2 P = a|ab|abc (a-P1 ; ab – P2 ; abc – P3)

UWAGA: to, że zachodzi $P \in L^n$ oznacza, że słowo P można podzielić na n podsłów (niekoniecznie takich samych) z których każde należy do L. Formalnie:

Gdy $P \in L^n$ to P możemy podzielić na n podsłów (niekoniecznie takich samych) wtedy i tylko wtedy, gdy $P=P_1P_2...P_n$ i wszystkie $P_i \in L$

Przykład 1:

Niech $L = \{a , ab\}$ Dla jakiego N zachodzi::

a | ab | ab | a | ab | ab $\in L^n$? (aababaabab)

P_1 P_2 P_3 P_4 P_5 P_6 N = 6! Podsłów.

Przykład 2:

- UWAGA! – Zobacz!: zarówno N jak i P_i – NIE muszą być wyznaczone jednoznacznie!
- Niech L = { a, ab, ba, aab, baab } dla jakiego N zachodzi Abaabaab $\in L^n$?
- Odpowiedź jest niejednoznacznym rozkładem tego słowa. Nawet w ramach tego samego wykładnika podziały mogą być różne! Na EGZAMINIE napisz wszystkie możliwe N i po jednym przykładzie dla każdego z nich! Np.: „ n= 3 bo…"
- Warto zrobić sobie tabelę i sprawdzić jaki jest minimalny i maksymalny wykładnik. W tym konkretnym przykładzie n = 3,4,5

  3 – a | baab | aab

  4 – a | baab | a | ab

  5 – a | ba | a | ba | ab

4) ***OPERACJA DOMKNIĘCIA KLEEN'EGO JĘZYKA:***

- Mając język L można obliczyć wszystkie jego potęgi!

$L^* = L_1 \cup L_2 \cup L_3 \cup \ldots \cup \ldots$ itd.

Uzyskany język, będący sumą wszystkich potęg języka L nazywamy ***domknięciem Kleen'ego*** języka L i oznaczamy **L***
Mamy zatem następującą definicję:

A) DOMKNIĘCIE KLEEN'EGO: (znjadują się tu słowa puste)
Domknięciem Kleen'ego języka L nazywamy język
$L^* = (\cup_L{}^n \ : n \geq 0)$
Bankowo w domknięciu Kleen'ego znajdziemy słowo puste. Ale jest też szansa, że pojawi się poprzez wystąpienie w samym języku L !

B) POZYTYWNE DOMKNIĘCIE KLEEN'EGO:
$L^+ = (\cup_L{}^n : n \geq 1)$
Zabranie ε z n,nie oznacza że w zbiorze L nie ma słowa pustego! (ε)
Oczywiście zachodzi $L^+ \subset L^*$ (pozytywne domknięcie zawiera się w domknięciu)
KIEDY $L^+ = L^*$? WtW, gdy $\varepsilon \in L$

Wyzwanie 7: (ZADANIE DOMOWE):

Oblicz domknięcie Klee'nego i pozytywne domknięcie Kleen'ego języków:
$L_1 = \{a\}$ oraz $L_2 = \{ \varepsilon , a\}$
Która z tych konstrukcji jest bardziej ekonomiczna (algorytmicznie?) ?

---== ZAJĘCIA 2 ==---

2 Strony nieprzepisane – ważne patrz na kartce!

Generacją ciągów znaków zajmuje się – gramatyka <- urządzenie formalne

Rozpoznawaniem poprawności ciągu znaków zajmuje się – AUTOMAT <- urządzenie formalne.

1) Pojęcie STANU:

Nieformalnie - stan urządzenia to gotowość tego urządzenia do wykonania pewnej czynności pod wpływem pewnego zewnętrznego impulsu. Ten sam impuls – ale różne stany – powodują inne reakcje!

Np.: impuls pilota – ale różne stany telewizora [włączony, czuwanie, działanie] – dadzą inne reakcje/efekty.

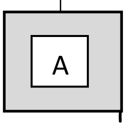
Tasiemka teoretyczna: jest nieskończona w prawo! Tasiemka to odpowiednik pamięci. Pojedyncza kratka to „pole" / rejestr

„Politechniczny" model automatu skończenie stanowego:

A	B	B	A						

<--- Głowica która umie czytać symbol i potrafi przechodzić tylko w prawo

1) Odczytać symbol 2) Przejść w prawo



<----- Urządzenie sterujące które ZAWSZE jest w jakimś stanie. Otrzymuje impuls od głowicy

$(q_0,a) \to q'$ + głowica w prawo (tak wygląda jeden cykl pracy maszyny zmienno stanowej)

$(q',b) \to q''$ + głowica w prawo

Te cykle są robione tak długo aż maszyna dotrze do pierwszego rejestru pustego. Wtedy zatrzymuje się . Zatrzymuje się jednak w konkretnym stanie. Stan który mówi nam czy dane słowo jest okey czy nie spełnia warunków. Żeby słowo okazało się spełniające warunki to cykl powinien zakończyć się na tak zwanym stanie akceptacyjnym. Inne stany to stany odrzucające.

q_0 -> stan początkowy – jest to jeden wspólny stan dla wszystkich przebiegów automatu. Jest to stan z którego zaczyna obliczenia maszyny dla każdego ze słów.

Automat zawiera więc:

- Stany (zbiór)
- Wyróżniony stan początkowy q_0
- Wskazane stany akceptujące
- Alfabet wejścia (skończony i niepusty zbiór)
- RDRZEŃ – serce automatu -> musimy zadać automatowi zbiór reakcji. (funkcja przejścia)

2) FORMALNA DEFINICJA! Automatu skończenie stanowego:

Deterministycznym automatem skończenie stanowym nazywamy uporządkowana piątkę: (GOTYCKIE A) „$\mathfrak{A}$" = < K , T , δ ,q_0 , H > w której:

1) K --- jest niepustym i skończonym zbiorem stanów.
2) T --- jest niepustym i skończonym alfabetem wejścia (taśmy)
3) $q_0 \in K$ --- jest wyróżnionym stanem początkowym
4) $H \subset K$ --- jest wyróżnionym zbiorem stanów akceptacyjnych (końcowych)
5) δ --- jest to FUNCKJA PRZEJŚCIA taką, że δ : K * T -> K (Para stan symbol)
   δ jest funkcją czyli każdej parze K*T musi przypisać wartość -> K. np.: [(q_0,a) -> q']
   Ponieważ zbiory K i T są skończone, to możemy przedstawić funkcję przejścia w postaci tabeli.
   Np.: k = {q_0 , q_1 , q_2 }
   T = { a , b } H = { q_2}

Strzałką oznaczamy stan początkowy

δ\T	a	b
-> q_0	q_2	q_0
q_1	q_0	q_1
$\underline{q_2}$	q_2	q_1

Podkreśleniem wskazujemy stany akceptujące

ZADANIE 1. Sprawdź czy dla powyższego deterministycznego automatu skończenie stanowego słowo „abba" jest słowem poprawnym.

a | b | b | a

1) (q_0 , a) -> q_2
2) (q_2 , b) -> q_1
3) (q_1 , b) -> q_1
4) (q_1 , a) -> q_0 Kończymy przejście na stanie q_0 który nie jest stanem akceptującym – a więc JEST stanem ODRZUCAJĄCYM.
   "abba" $\notin L(\mathfrak{A})$

ZADANIE 2. Sprawdź czy dla powyższego automatu słowo „babaa" jest słowem poprawnym. Odp: „babaa" ∈ L($\mathfrak{A}$)

ZADANIE 3. Sprawdź czy dla automatu określonego tabelką. Słowo „acbc" jest słowem poprawnym.

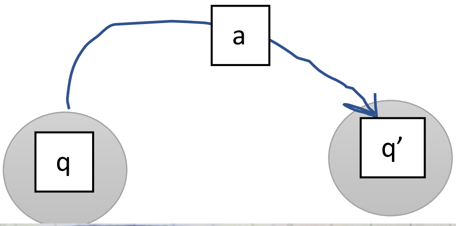
δ	a	b	c
-> q_0	q_1	q_0	q_2
q_1	q_2	q_3	q_1
q_2	q_2	q_3	q_0
$\underline{q_3}$	q_0	q_1	q_3

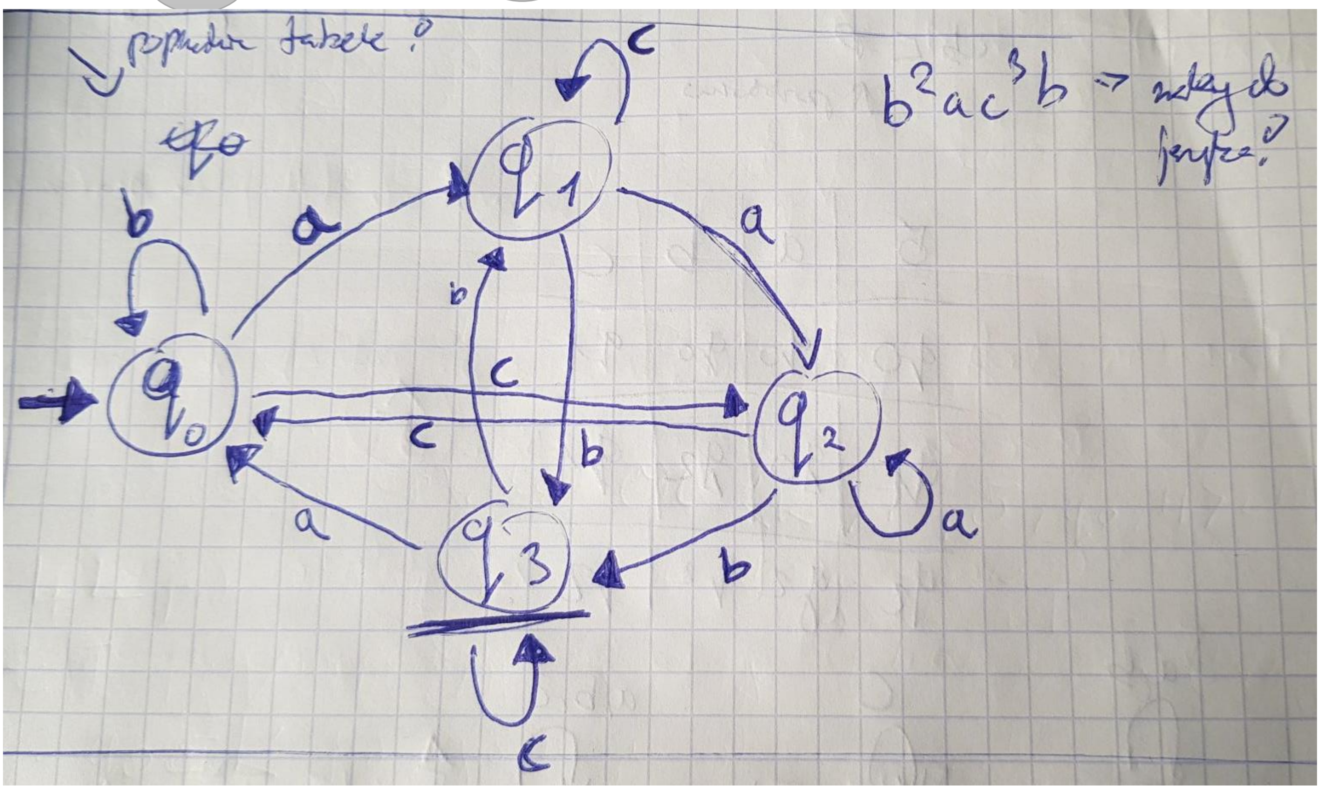
Odp: „acba" ∈ L($\mathfrak{A}$)

---

3) Zasada tworzenia diagramu automatu (grafu):

$\delta (q , a) = q'$





ZADANIE 4:

Narysuj diagram niedeterministycznego automatu skończenie stanowego, który rozpoczyna swoja pracę od stanu początkowego q_0 i przechodzi przez słowo $P \in \{ a , b , c \}^*$ zatrzymuje się w stanie końcowym akceptacyjnym wtedy i tylko wtedy, gdy liczba wystąpień symbolu „b" w P jest większa lub równa 2 tj.: $\#_b(P) >= 2$. Jak będzie wyglądać sprawa gdy $\#_b(P) = 2$?

$\#_b(P)$ – oznacza ilość wystąpień symbolu b w słowie P

$\#_b$ abbcb = 3

$\#_c$ abbcb = 1

$\#_d$ abbcb = 0

ZADANIE 5:

{…} zatrzymuje się w stanie końcowym wtedy i tylko wtedy gdy: „cb" $\sqsubset$ P (cb jest podsłowem P)

ZADANIE 6:

{…} zatrzymuje się w stanie końcowym wtedy i tylko wtedy gdy P zaczyna i kończy się na ciąg „ab"

ZADANIE 7:

{…} zatrzymuje się w stanie końcowym wtedy i tylko wtedy gdy symbol „a" występuje w P parzystą liczbę razy. Formalny zapis -> $2|\#_a P$

Słowo puste musi być akceptowalne! Bo 0 (wystąpień a) to parzysta cyfra.

Dlaczego deterministyczny automat skończenie stanowy jest deterministyczny?

- Ponieważ nie ma on możliwości wyboru! Determinizm działa jak funkcja. Jedne dane wejściowe dadzą zawsze te same dane wyjściowe! K * T -> K
- Automat NIEdeterministyczny będzie miał możliwość wyboru stanu.
- Determinizm automatu a kształt diagramu:

Z każdego stanu wychodzą krawędzie znaczone wszystkimi symbolami taśmy i dla każdego symbolu jest to <u>dokładnie jedna</u> krawędź. Naruszenie tego warunku skutkuje powstaniem automatu skończenie stanowego niedeterministycznego- to znaczy takiego w którym wartościami funkcji przejścia są zbiory stanów: δ: K*T – P(K)

Przechodzenie po grafie deterministycznego automatu jest jak nitka (szurek) (przechodzenie po sznurku).

A | B | B | A

Natomiast przechodzenie po grafie niedeterministycznym – jest drzewem!

δ	a	b
-> q_0	{ q_0 , q_1 }	∅
$\underline{q_1}$	{ q_0 , q_1 }	{ q_1 } (to też jest wybór – ale tylko jeden możliwy)

4) ZBIÓR POTĘGOWY – to rodzina wszystkich podzbiorów (zbiór zbiorów)

A = { a , b , c } -> zbiór

P(A) - > zbiór potęgowy zbioru A

P(A) = { ∅ , {a} , {b} , {c} , {ab} , {ac} , {bc}, A }

2^n – liczba elementów zbioru potęgowego zbioru o liczbie wartości n

A ma 3 elementy czyli P(A) ma 2^3 = 8 elementów.

Informacje przed formalną definicja JĘZYKA akceptowanego przez automat skończenie stanowy:

Lokalna zachowanie automatu vs. Globalne zachowanie automatu.

δ – funkcja przejścia w jednej chwili patrzy tylko na jeden symbol, jedną literę.

5) Rozszerzona funkcja przejścia automatu $\hat{\delta}$:

Niech $\mathfrak{A}$ = <K , T , δ , q_0 , H > będzie deterministycznym automatem skończenie stanowym. Rozszerzoną funkcją przejścia automatu $\mathfrak{A}$ nazywamy $\hat{\delta}$^ K*T -> K zdefiniowaną indukcyjnie w następujący sposób:

i) $\hat{\delta}(q, \varepsilon) = q$
ii) $\hat{\delta}(q, Pa) = \delta(\hat{\delta}(q,P),a)$
iii) Funkcja: $\hat{\delta}$: K x T* -> K jest rozszerzeniem funkcji δ: KxT -> K
Ponieważ na argumentach funkcji δ przyjmuje te same wartości co funkcja δ:

$\bar{\delta}(q,a) = \bar{\delta}(q, \varepsilon a) = \delta(\bar{\delta}(q, \varepsilon),a) = \delta(q,a)$

$\bar{\delta}(q,a) = \delta(q,a)$ | punkt i

iv) WŁASNOŚĆ: $\bar{\delta}(q,PQ) = \bar{\delta}(\bar{\delta}(q,P),Q)$

ZADANIE 8: Niech automat $\mathfrak{A}$ będzie zadany poniższą funkcją przejścia.

Oblicz $\bar{\delta}(q_0$, aba) oraz $\bar{\delta}(q_1$, abbaba)

δ	a	b
-> q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_0	q_2

$\bar{\delta}(q_0$, a) patrz na definicje! Tutaj P = ε!!!

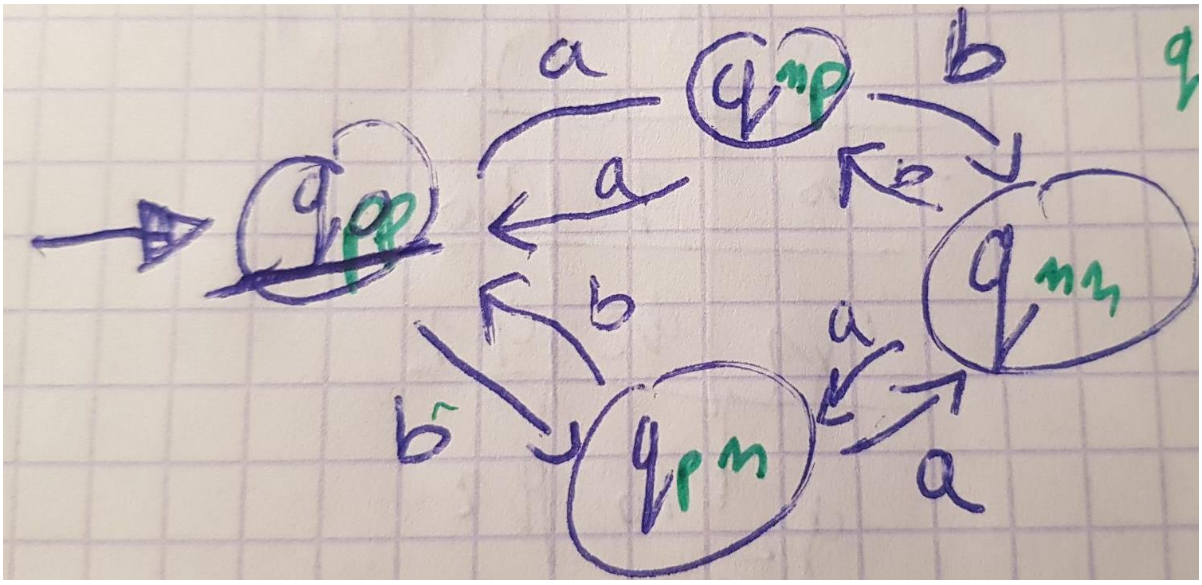
6) DEFINICJA: JĘZYK akceptowalny przez deterministyczny automat $\mathfrak{A}$:

Niech $\mathfrak{A}$ = <K , T , δ , q_0 , H> będzie deterministycznym automatem skończenie stanowym. Językiem akceptowalnym przez $\mathfrak{A}$ nazywamy zbiór L($\mathfrak{A}$) taki, że:

$L(\mathfrak{A}) = \{ P \in T^* : \bar{\delta}(q_0 , P) \in H \}$

ZADANIE 9: Zarysuj diagram/graf automatu det. Skoń. Stan. $\mathfrak{A}$ takiego, że $L(\mathfrak{A}) = \{ P \in \{ a,b \}^* : 2|\#_aP \wedge 2|\#_bP \}$

p – parzysta n - nieparzysta



---== Ćwiczenia 1 ==---

W wykonywaniu ćwiczeń i wszelkiego rodzaju zadań, najlepiej rozpocząć od tych najprostszych możliwych przykładów i następnie rozbudowywać do coraz trudniejszych. Dokańczaj robienie grafu z ogromną ostrożnością!

ZADANIE 10: Narysuj diagram niedeterministycznego automatu skończenie stanowego, $\mathfrak{A}$ takiego, że $L(\mathfrak{A}) = P \in \{ a, b \}^* : 2 | \#_a P \wedge 2 \nmid \#_b P$. PODPOWIEDŹ: zmień nazwy stanów akceptacyjnych! Np.: q_{pp} -> q_{pn}.

ZADANIE 11: {...} $\mathfrak{A}$ takiego, że $L(\mathfrak{A}) = P \in \{ a, b, c \}^* : ac \sqsubset P \;\; cb \sqsubset P\}$

ZADANIE 12: {...} $\mathfrak{A}$ takiego, że $L(\mathfrak{A}) = P \in \{ a , b \}^* : aba \not\sqsubseteq P$.

ZADANIE 13: {...} $\mathfrak{A}$ takiego, że $L(\mathfrak{A}) = P \in \{ a , b \}^* : 2 \nmid \#_a P\}$

ZADANIE 14: {...} $\mathfrak{A}$ w którym $T = \{ a , b, c \}$ i takiego, że $L(\mathfrak{A}) = \emptyset$

ZADANIE 15: {...} w którym $T = \{ a, b, c \}$ i $P \in L(\mathfrak{A})$ wtedy i tylko wtedy kiedy liczba 5 dzieli P (zapisujemy to jako $|P|$).

ZADANIE 16: {...} $T = \{ 0, 1, 2, 3 \}$ i $a_1 a_2 \dots a_n \in L(\mathfrak{A})$ gdy $4 | \Sigma\, a_1 a_2 \dots a_n$ (np.: $10021 \in L(\mathfrak{A})$ ALE -> $1302102 \notin L(\mathfrak{A})$).

ZADANIE 17: {...} w którym $T = \{ 0, 1 \}$ i $P \in L(\mathfrak{A})$ wtedy i tylko wtedy gdy, P w zapisie binarnym koduje liczbę naturalną podzielną przez 4 (dopuszczamy istnienie wielu zer po lewej stronie!) UWAGA -> NA EGZAMINIE! -> Najważniejsza jest świadomość, że słowo puste jest kłopotliwe, jak robisz zadanie i zakładasz, że słowo puste jest jakiś (np. podzielne przez 4) to MUSISZ TO ZAŁOŻENIE NAPISAĆ!!!

ZADANIE 18: {…} w którym T = { 0, 1 } i P ∈ L(𝔄) wtedy i tylko wtedy gdy P w zapisie binarnym koduje liczbę naturalną podzielną przez 3.

ZADANIE 19: (Domowe) {…} T={ 0, 1 } i P ∈ L(𝔄) wtedy i tylko wtedy gdy w P Podsłowo 00 i 11 albo występują jednocześnie, albo nie występują w cale.

---== ZAJĘCIA 3 ==---

ZADANIE 20: {…} 𝔄 W którym T = { 0, 1 } i P ∈ L(𝔄) wtw gdy w P symbol 1 występuje na miejscu drugim od prawej strony. Wskazówka: Dobrze przemyśl nazwanie stanów!

Miejsce n-te od prawej strony = 2^n stanów automatu deterministycznego. Bardzo możliwe podobne zadanie na egzaminie! Np.: „{…} występuje na miejscu trzecim od prawej strony" itp. itd.

ZADANIE 21: Treść podobna do ZAD20. Ale tym razem rozważ automat, który zaakceptuje podobny język, ale który może wybrać / odgadnąć następny stan. (Niedeterministyczny automat skończenie stanowy) gdy N-te miejsce po prawej stronie = n+1 stanów całego automatu.

Wniosek: Zatem w dwóch poprzednich zadaniach i dla naszych automatów – liczba stanów automatu deterministycznego wynosi $2^n/2$ (2^{n-1}), gdzie n jest liczbą stanów automatu niedeterministycznego.

---

1) IDEA automatu NIEdeterministycznego skończenie stanowego:

- Funkcja przejścia nied. aut. skoń. st. każdej parze (stan , symbol) przyporządkowuje ZBIÓR stanów. K x T -> P(K)
  Gdzie zbiór P(A) jest zdefiniowany następująco:
  P(A) = { B : B ⊂ A }

  Przypomnienie:
- jeśli A = { a, b, c } to:
  P(A) = { ∅ , {a} , {b} , {c} , {a,b} , {a,c} , {b,c} , {a,b,c} }
  Jeśli $\overline{\overline{A}}$ = n to $\overline{\overline{P(A)}}$ = 2^n (dwie kreski -> oznaczają, że mówimy o ilości elementów danego zbioru)
- ∅ ⊂ A oraz A ⊂ A oraz ∅ = { }

2) DEFINICJA FORMALNA! Automatu niedeterministycznego skończenie stanowego:

Definicja jest niemalże taka sama! Ale zmiany zachodzą w punkcie 3 i 5!

NIEdeterministycznym automatem skończenie stanowym nazywamy uporządkowana piątkę: „$\mathfrak{A}$" = < K , T , δ , Q_0 , H > w której:

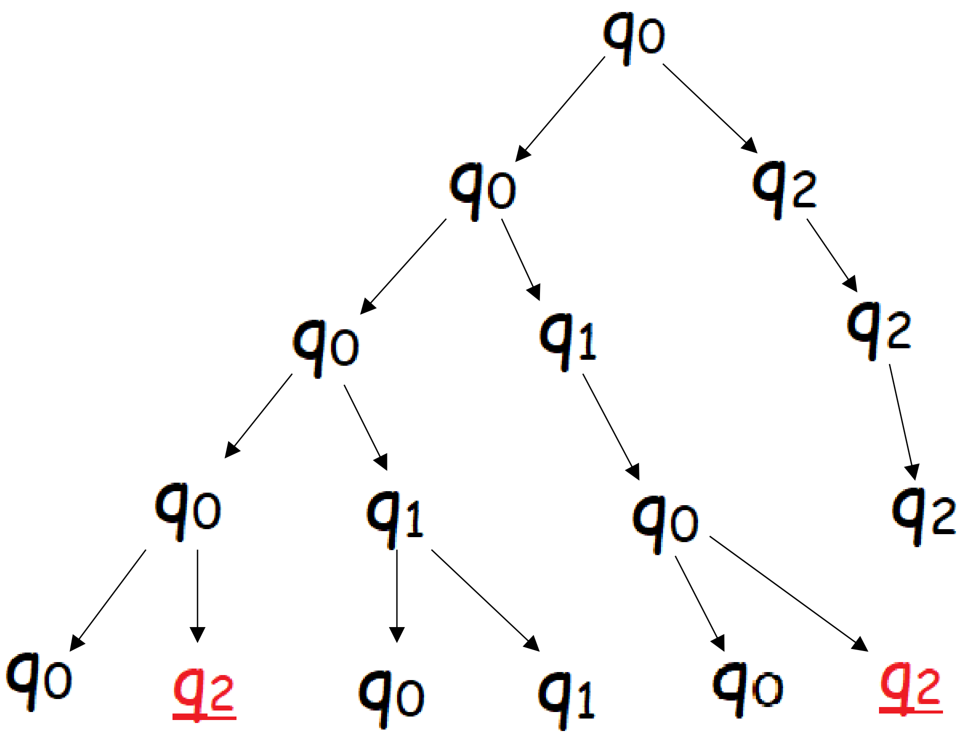
1) K --- jest niepustym i skończonym zbiorem stanów.
2) T --- jest niepustym i skończonym alfabetem wejścia (taśmy)
3) Q_0 ∈--- jest wyróżnionym zbiorem stanów początkowym
4) H ⊂ K --- jest wyróżnionym zbiorem stanów akceptacyjnych (końcowych)
5) δ --- jest to niedeterministyczna FUNCKJA PRZEJŚCIA taka, że δ : K * T -> P(K) .

Automat nied. może rozpocząć się od różnych stanów początkowych!

Drzewiasta reprezentacja przejścia niedeterministycznego automatu skończenie stanowego: Niech nied.aut.sk.st. -> $\mathfrak{A}$ będzie zadany następującą tabelą przejścia. δ: K x T -> P(K). Weźmy słowo „abba" – zbadajmy możliwość przebiegu $\mathfrak{A}$ nad P.

δ	a	b
-> q_0	{ q_0 , q_2 }	{ q_0 , q_1 }
-> q_1	{ q_0 , q_1 }	{ q_0 }
q_2	∅	{ q_2 }

- na zbiorze pustym (∅) automat się zawiesza!!!



Zauważ, że każda wartość funkcji przejścia należy do zbioru P(K).

Można wyróżnić takie przejścia gdzie stany końcowe są odrzucające lub akceptujące.

Wyróżniamy 2 rodzaje akceptacji słów:

1) Miękka akceptacja słowa: gdy możemy znaleźć co najmniej 1 stan końcowy akceptacyjny.
2) Twarda akceptacja słowa: Jeśli wszystkie stany końcowe są akceptujące

   H ≠ ∅

W niedet. aut.skoń.stan. będziemy korzystać z miękkiego akceptowania!

Możliwym jest również zamiana tego wykresu drzewiastego -> w wykres liniowy. Czyli Zamiana niedeterministycznego w deterministyczny!

---

3) ROZSZERZONA funkcja aut. Niedeterministycznego:

Niech $\mathfrak{A}$ = < K , T , δ ,Q_0 , H > będzie niede.au.sk.st. Rozszerzoną funkcją przejścia automatu $\mathfrak{A}$ nazywamy funkcję $\tilde{\delta}$ P(K) x T $^{\times}$ -> P(K) zdefiniowaną indukcyjnie:

i) $\tilde{\delta}$ (A, ε) = A

ii) $\tilde{\delta}$ (A , Pa) = ∪ $\bigcup_{q \in \tilde{\delta}(A,P)} \tilde{\delta}(q, a)$

4) Język akceptowany przez automat niedeterministyczny:

Niech $\mathfrak{A}$ = < K , T , δ ,Q_0 , H > będzie niede.au.sk.st Językiem akceptowanym przez $\mathfrak{A}$ nazywamy zbiór L($\mathfrak{A}$) taki że, L($\mathfrak{A}$) = { P ∈ T* : $\tilde{\delta}$ (Q_0 , P) ∩ H ≠ ∅ }

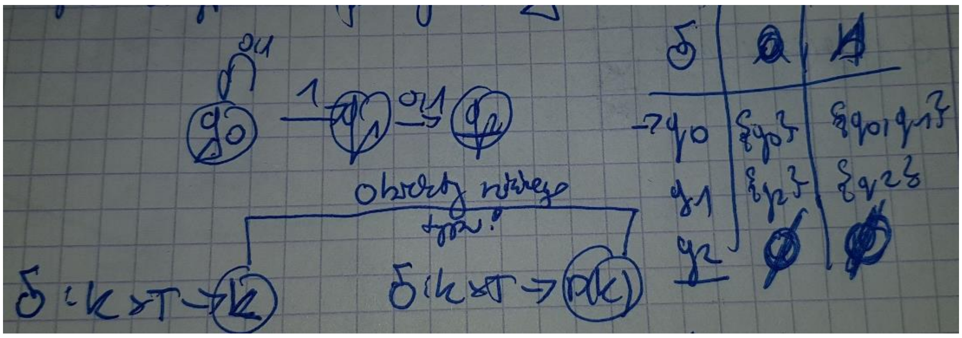
Jeśli w zbiorze H znajduje się chociaż jeden stan akceptujący to słowo akceptujemy (gdy zachodzą też te inne warunki -> słowo będzie zaakceptowane)

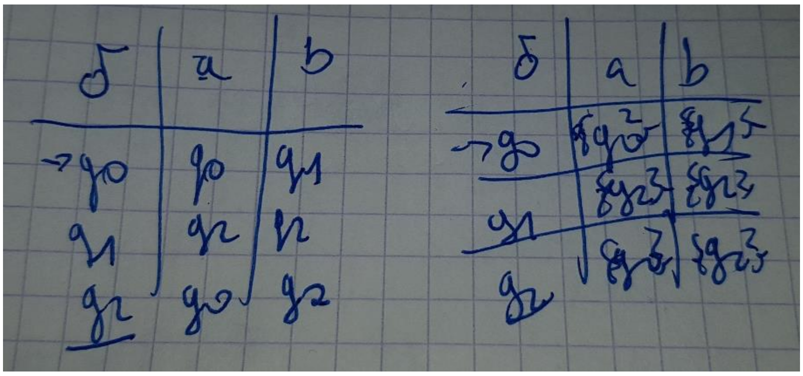
ZADANIE 22: Narysuj diagram przejścia deterministycznego i niedeterministycznego aut. Sk. St. $\mathfrak{A}$ w którym T = { 0, 1 } i P ∈ L($\mathfrak{A}$) wtw, gdy symbol 1 występuje na miejscu drugim od prawej strony!

δ : K * T -> K δ : K * T -> P(K) K i P(K) to obiekty różnego typu!

Każdy automat deterministyczny $\mathfrak{A}$ = < K , T , δ ,q_0 , H >, można przerobić na automat niedeterministyczny $\mathfrak{A}$ = < K , T , δ ,Q_0 , H >, tak że, L($\mathfrak{A}$) = L($\mathfrak{A}'$) i

Q_0 = { q_0 } oraz δ(q, a) = P wtw gdy δ` (q , a) = {P}



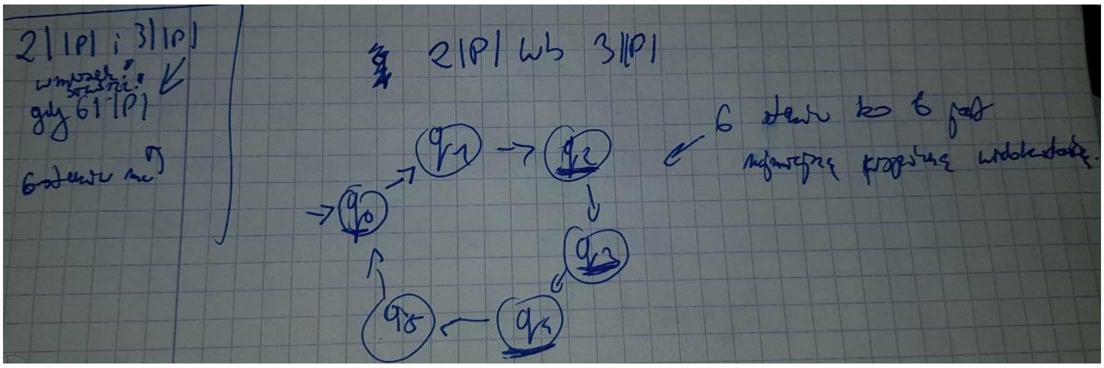


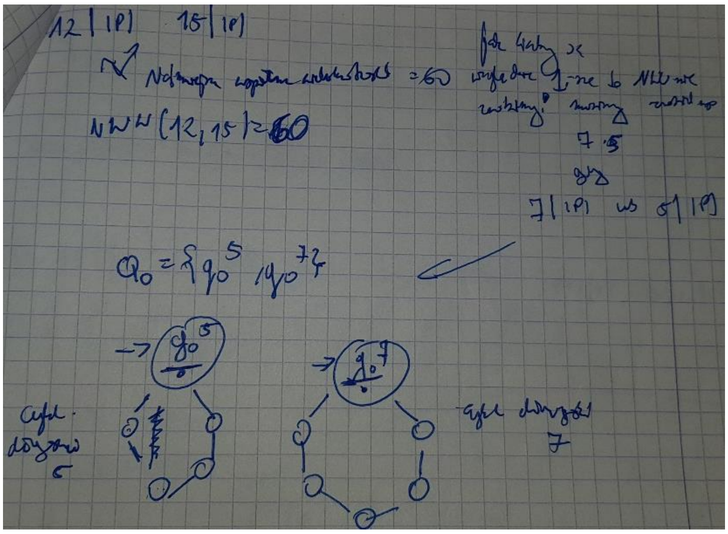
Wniosek: Dla każdego języka akceptowanego przez det. au. sk. st. $\mathfrak{A}$ możemy skonstruować niedeterministyczny au. sk. Stanowy $\mathfrak{A}'$ który zaakceptuje nam ten sam język. $\mathcal{L}_{det} \subset \mathcal{L}_{NIEdet}$

$\mathcal{L}_{det}$ -> jest to zbiór wszystkich języków akceptowanych przez automaty skończenie stanowe deterministyczne.

Zachodzi również $\mathcal{L}_{NIEdet} \subset \mathcal{L}_{det}$ oznacza to że: $\mathcal{L}_{NIEdet} = \mathcal{L}_{det}$!

Czyli inkluzja zachodzi! (Twierdzenie Scotta). Co w takim razie wnosi niedeterminizm? Wnosi wygodę w konstruowaniu automatów i jest mniej tych stanów.





5) Twierdzenie Scotta ma dwie inkluzje: łatwą i trudną:

1) Łatwa -> . $\mathcal{L}_{det} \subset \mathcal{L}_{NIEdet}$

2) Trudna -> $\mathcal{L}_{NIEdet} \subset \mathcal{L}_{det}$

Twierdzenie Scotta -> $\mathcal{L}_{NIEdet} = \mathcal{L}_{det}$

Ceną jaką płacimy za determinizację jest wykładniczy wzrost liczby stanów automatu deterministycznego.

ZADANIE 23:

Dowód: (konstrukcja):

a) Niedeterministyczny aut, sk. st. $\mathfrak{A}$
b) konstruujemy deterministyczny aut. sk . st.
$\mathfrak{A}$` =< K` , T` , δ` , q_0` , H` > dla którego zajdzie L($\mathfrak{A}$) = L($\mathfrak{A}'$).

Kładziemy:
T ` = T
K` = P(K)
q_0` = Q_0
H` = { $A \in K`$: $A \cap H \neq \emptyset$ }
δ `(A,a) = $\cup_{q \in A} \delta(q, q)$

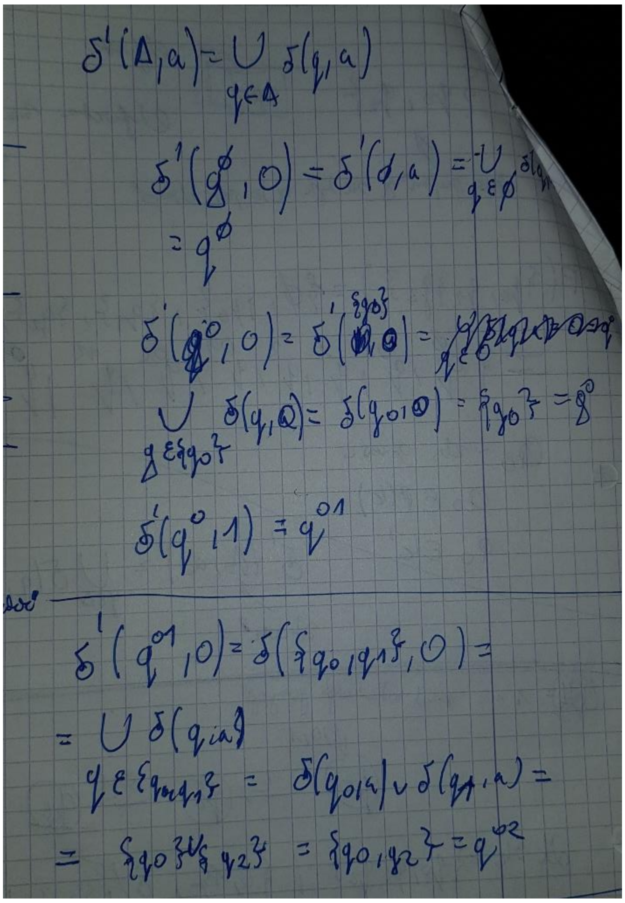
Uwaga!: { q_0 , q_1 } -> nie przejmuj się, że jest to zbiór! Nasza etykieta dla tego konkretnego stanu może być następująca: q_{01} !

ZADANIE : L($\mathfrak{A}$) = „1 na przedostatniej pozycji"
$\mathfrak{A}$: < K={ q_0 , q_1 , q_2 } , T = { 0, 1 } , δ , Q_0 = { q_0 } , H = { q_2 } >

δ	0	1
-> q_0	{ q_0 }	{ q_0 , q_1 }
q_1	{ q_2 }	{ q_2 }
q_2	∅	∅

$\mathfrak{A}$` : K` = { $q\emptyset$, q_0 , q_1 , q_2 , q_{01} , q_{02} , q_{12} , q_{012} }
T ` = { 0, 1}
δ`
q_0` = q_0
H` = { q_2 , q_{12} , q_{012} , q_{02} }

δ `	0	1
$q\emptyset$	$q\emptyset$	$q\emptyset$
q_0	q_0	q_{01}
q_1	q_2	q_2
q_2	$q\emptyset$	$q\emptyset$
q_{01}	q_{02}	q_{012}
q_{02}	q_0	q_{01}
q_{12}	q_2	q_2
q_{012}	q_{02}	q_{012}


$\delta'(\Delta,a) = \bigcup_{q\in\Delta} \delta(q,a)$

$\delta'(q^{\emptyset},0) = \delta'(\emptyset,a) = \bigcup_{q\in\emptyset} \delta(q,a) = q^{\emptyset}$

$\delta'(q^{0},0) = \delta'(\{q_0\},0) = \bigcup_{q\in\{q_0\}} \delta(q,0) = \delta(q_0,0) = \{q_0\} = q^{0}$

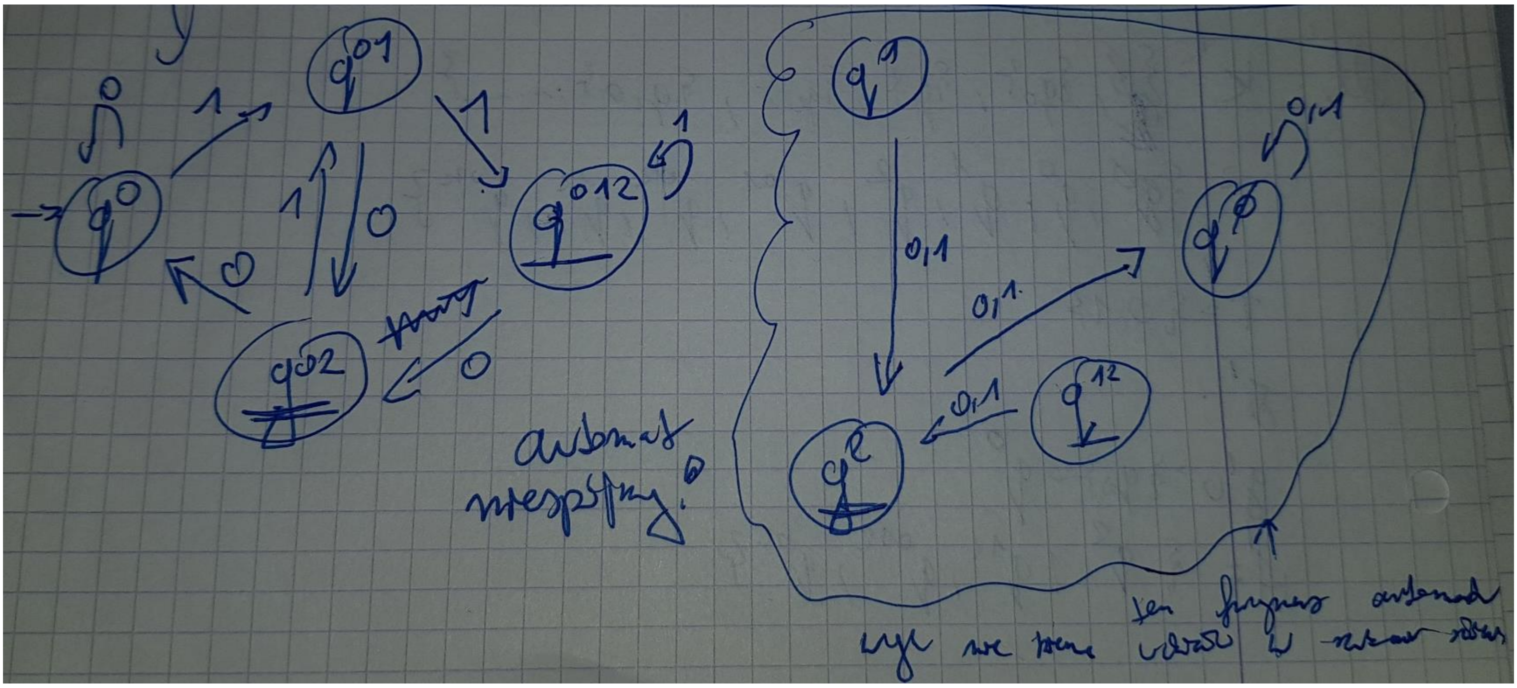
$\delta'(q^{0},1) = q^{01}$

$\delta'(q^{01},0) = \delta(\{q_0,q_1\},0) = \bigcup_{q\in\{q_0,q_1\}} \delta(q,a) = \delta(q_0,a) \cup \delta(q_1,a) = \{q_0\} \cup \{q_2\} = \{q_0,q_2\} = q^{02}$

Stany akceptujące są tam gdzie w indexie jest 2 (w tym przypadku!)

Narysujmy graf powyższego automatu det.

Sk. St. $\mathfrak{A}$`



Zadanie domowe! Wybrać 2 z 3 poniższych automatów zadanych diagramami -> i je zdeterminizować!