Automaty i gramatyki

Marcin Kubica

2004/2005

Niniejsze notatki do wykładów oparte są na podanej literaturze. Podane w nawiasach kwadratowych numery podrozdziałów, ćwiczeń itp. odnoszą się właśnie do wymienionych poniżej pozycji.

Wykłady oznaczone gwiazdką należy pominąć na zajęciach z JAO, a uwzględnić na wykładach z TO.

Wykład 1. Wprowadzenie

Celem niniejszego wykładu jest:

- ....
- poznanie podstawowych środków opisu składni (wyrażenia regularne, gramatyki bezkontekstowe),
- poznanie narzędzi wspomagających przetwarzanie danych o określonej składni (flex, bison).

W trakcie badań nad obliczalnością pojawiło się wiele modeli, które miały za zadanie ujęcie pojęcia *obliczalności*. Możemy je uszeregować wg. rosnącej siły wyrazu:

1. automaty skończone, wyrażenia regularne (stała skończona pamięć),
2. automaty stosowe (stała skończona pamięć + nieograniczony stos),
3. bez ograniczeń:
   - maszyny Turinga,
   - systemy Posta,
   - rachunek λ (Alonzo Church, Stephen C. Kleene), i inne.

Modele te powstały na długo przed tym zanim powstały komputery.

Równocześnie Noam Chomsky, lingwista, próbował sformalizować pojęcia *gramatyki* i *języka*. Stworzył on hierarchię języków o rosnącej złożoności:

1. gramatyki liniowe,
2. gramatyki bezkontekstowe,
3. gramatyki kontekstowe,
4. gramatyki dowolne.

Hierarchia ta w zadziwiający sposób odpowiada wymienionym modelom obliczeń:

- gramatyki liniowe = automaty skończone = wyrażenia regularne,
- gramatyki bezkontekstowe = automaty stosowe,
- gramatyki kontekstowe = liniowo ograniczone maszyny Turinga,
- dowolne gramatyki = maszyny Turinga.

Czy może być to tylko przypadek?

Należy pamiętać, że wszystko to są *modele* interesujących nas pojęć. Umiejętność posługiwania się niektórymi z nich przydaje się w praktyce, a w przypadku innych najistotniejsze będą reprezentowane przez nie pojęcia.

1.1 Literatura

- [HU] John E. Hopcroft, Jeffrey D. Ullman, *Wprowadzenie do teorii automatów, języków i obliczeń*, PWN, 2003 r.
- [K] Dexter C. Kozen, *Autoamata and Computability*,
- A. V. Aho, R. Sethi, J. D. Ullman, Kompilatory, WNT, 2002,
- W. M. Waite, G. Goos, Konstrukcja kompilatorów, WNT, 1989.

1.2 Podstawowe pojęcia

We wprowadzeniu wspomnieliśmy o *obliczeniach* i *językach*. Gdy mówimy o obliczaniu, to zwykle mamy na myśli obliczenie wartości takiej czy innej funkcji. Gdy zaś mówimy o języku, to mamy na myśli (potencjalnie nieskończony) zbiór napisów złożonych ze znaków ustalonego (skończonego) alfabetu. Jak połączyć te dwa pojęcia?

Def. 1 *Problem decyzyjny* to funkcja, która możliwym danym wejściowym przyporządkowuje wartości logiczne tak/nie. □

Język nie jest niczym więcej niż problemem decyzyjnym określonym dla napisów nad ustalonym alfabetem.

Def. 2 Alfabet, napis itp.

- *Alfabet* to dowolny niepusty, skończony zbiór. Może to być np. zbiór cyfr dziesiętnych, zbiór znaków ASCII, czy zbiór bitów $\{0, 1\}$. Alfabet będziemy zwykle oznaczać przez Σ, a elementy alfabetu przez $a, b, c, \ldots$.
- *Napis* nad alfabetem Σ, to dowolny skończony ciąg elementów Σ. W szczególności, pusty ciąg jest napisem nad każdym alfabetem. Oznaczamy go przez ε. Napisy będziemy zwykle reprezentować przez zmienne $x, y, z, \ldots$.
- *Długość* napisu oznaczamy przez $|\cdot|$. Np. $|ala| = 3$, $|\varepsilon| = 0$.
- *Sklejanie* napisów zapisujemy pisząc po sobie sklejane składowe. Np., jeżeli $x = ala, y = ma, z = kota$, to $xyz = alamakota$. Sklejanie napisów jest łączne, tzn. $x(yz) = (xy)z$, a ε jest elementem neutralnym sklejania, tzn. $x\varepsilon = \varepsilon x = x$.
- Przez a^n oznaczamy n-krotne powtórzenie symbolu a. Np. $a^5 = aaaaa$. $a^0 = \varepsilon$, $a^{n+1} = a^n a$.
- Podobnie, przez x^n oznaczamy n-krotne powtórzenie napisu x. $x^0 = \varepsilon$, $x^{n+1} = x^n x$.
- Przez $\#_a(x)$ oznaczamy *liczbę wystąpień* symbolu a w napisie x, np. $\#_a(ala) = 2$.
- *Prefiksem* napisu nazywamy dowolny jego początkowy fragment, tzn. x jest prefiksem y wtw., gdy istnieje takie z, że $xz = y$. Jeżeli ponadto $x \neq \varepsilon$ i $x \neq y$, to mówimy, że x jest *właściwym* prefiksem y.

□

Def. 3 Zbiory (języki) będziemy zwykle oznaczać przez $A, B, C, \ldots$. Operacje na zbiorach (napisów):

- suma $\cup$, przecięcie $\cap$, odejmowanie $\setminus$,
- dopełnienie $\overline{A} = \Sigma^* \setminus A$,
- złączenie (konkatenacja) zbiorów $AB = \{xy : x \in A, y \in B\}$,
- potęgowanie zbiorów: $A^0 = \{\varepsilon\}$, $A^{n+1} = A^n A$,
- $A^* = \bigcup_{n \geq 0} A^n$; w szczególności Σ^* to *zbiór wszystkich napisów* nad alfabetem Σ; przyjmujemy, że $\emptyset^* = \{\varepsilon\}$,
- $A^+ = AA^*$.

□

Zwróćmy uwagę na pewne proste własności operacji na zbiorach:

- $\{\varepsilon\}A = A\{\varepsilon\} = A$,
- $\emptyset A = A\emptyset = \emptyset$,
- $A(B \cup C) = AB \cup AC$.

1.3 Języki jako problemy decyzyjne

Język, to podzbiór zbioru Σ^*. Możemy więc język traktować jak problem decyzyjny, dla zbioru danych Σ^*.

Czy dla każdego języka istnieje rozpoznający go automat skończony / automat stosowy / program komputerowy. Zdecydowanie nie!

Zbiór Σ^* jest przeliczalny, gdyż jesteśmy w stanie ponumerować wszystkie słowa: puste, jednoliterowe, dwuliterowe, itd. Czyli $|\Sigma^*| = \aleph_0$. Język to podzbiór Σ^*. Czyli wszystkich możliwych języków jest $2^{\aleph_0}$. Z drugiej strony każdy mechanizm rozpoznający można opisać za pomocą słowa — program komputerowy to po prostu słowo. Tak więc języków jest ilościowo więcej niż mechanizmów rozpoznających. Stąd, tylko niektóre języki można rozpoznawać automatycznie, niezależnie od wybranego modelu obliczeń.

1.4 Powtórzenie pojęć dotyczących relacji

Kiedy relacja jest:

- zwrotna,
- przechodnia,
- symetryczna,
- antysymetryczna.

Grafowa interpretacja powyższych pojęć. Pojęcie domknięcia relacji:

- zwrotniego,
- przechodniego,
- symetrycznego

i ich interpretacja grafowa.

1.5 Praca domowa

- Jak jest różnica między $\emptyset$ i $\{\varepsilon\}$? Ile słów zawierają te języki?
- Podaj wszystkie słowa z języka $\{aba, ba, a\}\{ba, baba\}$.
- Podaj które prefiksy słowa *ababbbabab* są równocześnie jego sufiksami.

Ćwiczenia

- Ćwiczenia na podstawy matematyczne (można wykorzystać przed pierwszym wykładem):
  - Operacje na zbiorach. Niech $A, B \subset X$. Narysować diagram przedstawiający te zbiory. Uszeregować poniższe zbiory zgodnie z zawieraniem. Które są sobie równe? Które nie są porównywalne? Rozwiązanie można przedstawić w postaci DAG-a.
    * $A \cup B$,
    * $A \cap B$,
    * X,
    * A,
    * B,
    * $\emptyset$,
    * $X \setminus A$,
    * $A \cap B \cap (X\ A)$,
    * $(A \cup B) \setminus (A \cap B)$,
    * $X \setminus (A \cup B)$,
    * $X \setminus (A \cap B)$,
    * $(A \setminus B) \cup (B \setminus A)$,
    * $(X \setminus A) \cup (X \setminus B)$,
    * $(X \setminus A) \cap (X \setminus B)$,
    * $X \setminus ((X \setminus A) \cup (X \setminus B))$,
    * $X \setminus ((X \setminus A) \cap (X \setminus B))$.
  - Pojęcia prefiksu, postfiksu, podciągu i podciągu spójnego (podsłowa). (W razie potrzeby wyjaśnić te pojęcia). Wybieramy dowolny numer telefonu i traktujemy go jak ciąg cyfr. Podać:
    * wszystkie prefiksy tego ciągu,
    * wszystkie postfiksy tego ciągu,
    * kilka podciągów (niekoniecznie spójnych); ile jest wszystkich?
    * kilka spójnych podciągów (podsłów); ile jest wszystkich?
  - Wybieramy zbiór 2–3 elementowy. Jakie są jego podzbiory? Ile podzbiorów ma zbiór n-elementowy?
- proste przećwiczenie wprowadzonych pojęć na przykładach, np.:
  - $\{a, ab, ba, baba\} \cap \{ab, ba\}^2$,
  - $\{a, ab, ba\} \setminus \{a\}^*\{bb\}^*\{a\}^*$.

- ćwiczenia na sprawdzanie różnych tożsamości (w przypadku prawdziwych uzasadnienie, w przypadku fałszywych kontrprzykład; jeśli zachodzi zawieranie, to w którą stronę?):
  - $(A \cup B) \cup C = A \cup (B \cup C)$,
  - $(A \cup B) \cap C = A \cup (B \cap C)$,
  - $(A \cup B) \cap C \subseteq A \cup (B \cap C)$,
  - $A(B \cup C) = AB \cup AC$,
  - $A(B \cap C) = AB \cap AC$,
  - $A(B \cap C) \subseteq AB \cap AC$,
  - $(AB)^* = A^*B^*$,
  - $(A \cup B)^* = (A^*B^*)^*$,
  - $A^{*+} = A^{+*}$,
  - $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$.
- Co to za języki:
  - $(A \cap A^*)^*$,
  - $A^* \setminus A^+$,
  - $(A \cap B)^* \setminus (A^* \cup B^*)$,
  - $\{a, ba, bb, ab\}^*$,
  - $(\Sigma\Sigma)^*$, (słowa parzystej długości),
  - $\overline{(\Sigma\Sigma)^*}$, (słowa nieparzystej długości),
  - $\Sigma(\Sigma\Sigma)^*$ (też słowa nieparzystej długości),
  - ΠΙΣΣΑ (a to jest po grecku pizza :-).
- Używając symboli i operacji wprowadzonych na pierwszym wykładzie (czyli operacji na słowach, zbiorach, językach i definicji zbiorów) zdefiniować języki:
  - słowa zawierające tyle samo liter a i b,
  - słowa nad alfabetem $\{a, b\}$ (nie)parzystej długości,
  - alarmowe, skrócone i międzymiastowe numery telefoniczne,
  - inne ...
- [HU ćw.1.7] Podać domknięcie zwrotnio-przechodnie i domknięcie symetryczne relacji $\{(1, 2), (2, 3), (3, 4), (5, 4)\}$.
- Pokazać, że następujące relacje są relacjami równoważności:

- słowa x i y są w relacji wtw., gdy $|x| = |y|$,
- słowa x i y są anagramami,
- języki A i B są w relacji wtw., gdy $A^* = B^*$.

- Pokazać, że następujące relacje są częściowymi porządkami:
  - relacja zawierania $\subseteq$ na językach nad ustalonym alfabetem,
  - relacja porządku leksykograficznego na słowach,
  - relacja bycia prefiksem.

Wykład 2. Wzorce i yrażenia regularne

2.1 wprowadzenie

Opowiastka o wyrażeniach regularnych w Unixie. Zastosowanie w programach (`grep`, `sed`, `awk`, `lex`).

2.2 Wzorce

W Unixie możemy spotkać pod nazwą *wyrażenia regularne* bogaty język do definiowania wzorców:

- x – znak 'x',
- "x" – x brane dosłownie,
- $\backslash x$ – jak w C,
- . – dowolny znak (oprócz końca wiersza),
- [xyz] – dowolny ze znaków 'x', 'y' i 'z',
- [a–z] – dowolny znak od 'a' do 'z', (można łączyć, np. [abk–lA–Z]),
- [^xyz] – dowolny znak oprócz 'x', 'y' i 'z',
- α^* – zero lub więcej powtórzeń α,
- α^+ – jedno lub więcej powtórzeń α,
- $\alpha?$ – zero lub jedno wystąpienie α,
- $\alpha\{x, y\}$ – od x do y powtórzeń α,
- $\alpha_1 \mid \alpha_2$ – α_1 lub α_2,
- «EOF» – koniec pliku,
- (α),
- ...

Przykład:

- zapisy dziesiętne liczb naturalnych,
- identyfikatory w wybranym języku programowania,
- numery rejestracyjne (prywatne i służbowe).

Mając dany wzorzec możemy zastanawiać się nad odpowiedziami na następujące pytania:

- Jak sprawdzić, czy słowo pasuje do wzorca?
- Jak znaleźć wszystkie wystąpienia wzorca w tekście?
- Czy jakiekolwiek słowo pasuje do wzorca?
- Jaki język tworzą słowa pasujące do wzorca?
- Czy dla każdego języka istnieje opisujący go wzorzec? Nie.
- Czy wszystkie operacje są potrzebne? Nie.

2.3 Języki opisywane przez wzorce

Mając dany wzorzec, możemy określić język złożony ze wszystkich słów pasujących do danego wzorca. Definicja ta jest indukcyjna ze względu na strukturę wzorca.

Def. 4 Niech α i β będą wzorcami. Przez $L(\alpha)$ oznaczamy język złożony z tych słów, które pasują do wzorca α.

- $L(\varepsilon) = \{\varepsilon\}$,
- $L(x) = L(\text{“}x\text{”}) = \{x\}$,
- $L(.) = \Sigma$,
- $L([a_1 a_2 \ldots a_k]) = \{a_1, a_2, \ldots, a_k\}$,
- $L([a - b]) = \{a, \ldots, b\}$ — zależy od kolejności znaków w kodowaniu ASCII,
- $L([\hat{}a - b]) = \Sigma \setminus L([a - b])$
- $L(\alpha\beta) = L(\alpha)L(\beta)$,
- $L(\alpha|\beta) = L(\alpha) \cup L(\beta)$,
- $L(\alpha^*) = L(\alpha)^*$,
- $L(\alpha^+) = L(\alpha)^+ = L(\alpha)L(\alpha)^*$,

- $L(\alpha?) = L(\alpha) \cup \{\varepsilon\}$,
- $L(\alpha\{x, y\}) = L(\alpha)^x \cup L(\alpha)^{x+1} \cup \ldots \cup L(\alpha)^y$,
- ...

□

2.4 Wyrażenia regularne

Wzorce, tak jak są zdefiniowane w Uniksie, mają przede wszystkim być wygodne w użyciu. Okazuje się jednak, że wiele z konstrukcji występujących we wzorcach może być zastąpionych prostszymi konstrukcjami. W ten sposób, każdy wzorzec można przerobić na taki, który został zapisany przy użyciu tylko pewnego minimalnego zestawu konstrukcji. Takie wzorce, które zapisano przy użyciu tylko tego minimalnego zestawu operacji, nazywamy *wyrażeniami regularnymi.*

Def. 5 Wyrażenia regularne, to takie wzorce, które są zbudowane tylko przy użyciu:

- ε,
- $\emptyset$,
- a, dla $a \in \Sigma$,
- $\alpha\beta$,
- $\alpha|\beta$,
- α^*,
- (α).

Formalnie, wzorzec $\emptyset$ nie występuje w Uniksie. Możemy go utożsamić np. z $[z - a]$. □

Wyrażenia regularne będą nam potrzebne wówczas, gdy będziemy się zajmowali tym, co można wyrazić za pomocą wzorców i ogólnymi właściwościami języków, które można opisywać za pomocą wzorców. Dzięki ograniczeniu zestawu możliwych konstrukcji do niezbędnego minimum, będzie to prostsze.

2.5 Równoważność wzorców i wyrażeń regularnych

Fakt: Niech A będzie językiem. A można opisać wzorcem wtw., gdy można go opisać wyrażeniem regularnym.

Dowód; Jeżeli A można opisać wyrażeniem regularnym, to oczywiście można też opisać za pomocą wzorca.

Implikacja w drugą stronę jest mniej oczywista. Należy sprawdzić, że wszystkie konstrukcje występujące we wzorcach można zapisać używając wyrażeń regularnych.

2.6 Tożsamości dotyczące wzorców

Def. 6 Powiemy, że dwa wzorce α i β są równoważne, $\alpha \equiv \beta$, wtw., gdy $L(\alpha) = L(\beta)$. □

Przykładowe tożsamości dotyczące wzorców:

- $\alpha(\beta \mid \gamma) \equiv \alpha\beta \mid \alpha\gamma$,
- $(\beta \mid \gamma)\alpha \equiv \beta\alpha \mid \gamma\alpha$,
- $\emptyset\alpha \equiv \alpha\emptyset \equiv \emptyset$,
- $\alpha^+ \equiv \alpha \mid \alpha\alpha^+ \equiv \alpha \mid \alpha^+\alpha$,
- $(\alpha\beta)^*\alpha \equiv \alpha(\beta\alpha)^*$,
- $(\alpha^*)^* \equiv \alpha^*$,
- $(\alpha^*\beta^*)^* \equiv (\alpha \mid \beta)^*$.

2.7 Języki regularne

Def. 7 Powiemy, że język A jest *regularny*, wtw., gdy istnieje wyrażenie regularne α opisujące A, czyli $A = L(\alpha)$. □

Fakt: Każdy język skończony jest regularny.

Fakt: Suma języków regularnych jest regularna.

Fakt: Sklejenie (konkatenacja) języków regularnych jest regularne.

Fakt: Jeśli A jest regularny, to A^* też.

2.8 Praca domowa

- Podaj wyrażenie regularne równoważne wzorcowi: $([a - c] \mid a(b?))^+$.
- Podaj wzorzec opisujący poprawne numery indeksów studentów, np. postaci `s0124`.
- Uprość następujący wzorzec $(a \mid b)(aa \mid ab \mid bb \mid ba)^*(a \mid b)$.

Ćwiczenia

- [K Hw. 3.1] Dla poniższych języków podaj wzorce/wyrażenia regularne:
  - $\{x : x$ zawiera parzystą liczbę symboli $a\}$,
  - $\{x : x$ zawiera nieparzystą liczbę symboli $b\}$,
  - $\{x : x$ zawiera parzystą liczbę symboli a lub nieparzystą liczbę symboli $b\}$,
- Podaj wzorce/wyrażenia regularne opisujące język złożony ze słów:
  - nad alfabetem $\{a, b\}$, które nie zawierają podsłowa aa,
  - nad alfabetem $\{a, b, c\}$, które nie zawierają podsłowa aa,
  - nad alfabetem $\{a, b\}$, które zawierają podsłowo $bbab$,
  - nad alfabetem $\{a, b, c\}$ złożonych tylko z jednego rodzaju symboli,
  - nad alfabetem $\{a, b, c\}$ złożonych co najwyżej z dwóch rodzai symboli.
- [HU ćw.2.11, a) b)] Opisz języki reprezentowane przez następujące wyrażenia regularne:
  - $(11 \mid 0)^*(00 \mid 1)^*$,
  - $(1 \mid 01 \mid 001)^*(\varepsilon \mid 0 \mid 00)$,
- [K Ex. 11] Dla poniższych języków podaj wzorce/wyrażenia regularne:
  - $\{x \in \{a, b\}^* : x$ nie zawiera podsłowa $a\}$,
  - $\{x \in \{a, b\}^* : x$ nie zawiera podsłowa $ab\}$,
- [K Ex. 18] Porównaj wyrażenia regularne pod kątem równości/zawierania się odpowiadających im języków. Odpowiedzi uzasadnij. Udowodnij pomocnicze równanie $a(ba)^* = (ab)^*a$.

1)	$(0 \mid 1)^*$	$0^*\vert 1^*$,
2)	$0(120)^*12$	$01(201)^*2$,
3)	$\emptyset^*$	ε^*,
4)	$(0^*1^*)^*$	$(0^*1)^*$,
5)	$(01 \mid 0)^*0$	$0(10 \mid 0)^*$.

- [K Ex. 19] Oznaczmy $\alpha = (a \mid b)^*ab(a \mid b)^*$. Podaj wzorzec opisujący język $\overline{L(\alpha)}$, zakładając, że:
  1. $\Sigma = \{a, b\}$,
  2. $\Sigma = \{a, b, c\}$.

Wykład 3. Automaty skończone

3.1 Intuicje

Stan i przejście — intuicje. Zakładamy, że przejścia są natychmiastowe — jest to matematyczna abstrakcja. Przykłady systemów ze stanami i przejściami:

- kostka Rubik'a,
- zegarek cyfrowy,
- sterownik windy,
- wszechświat (?).

3.2 Automaty deterministyczne

Przykład: Konstrukcja automatu modelującego działanie video. Modelujemy podstawowe typy pracy i klawisze: play, stop, pause, fwd i rew.

Przykład: Problem misjonarzy i kanibali. Na brzegu znajduje się trzech misjonarzy i trzech kanibali. Mają do dyspozycji łódkę. W łódce mogą się zmieścić maksymalnie dwie osoby. Wiosłować umieją wszyscy kanibale i jeden misjonarz. Jeżeli na którymkolwiek brzegu zostanie więcej kanibali, niż misjonarzy, to misjonarze zostaną ... zjedzeni.

Obserwujemy sekwencje ruchów łódki wraz z jej pasażerami. Skonstruować diagram systemu akceptującego takie sekwencje ruchów, które prowadzą do przepłynięcia wszystkich na drugi brzeg.

Def. 8 Automat skończony (deterministyczny):

$$M = \langle Q, \Sigma, \delta, s, F \rangle$$

gdzie:

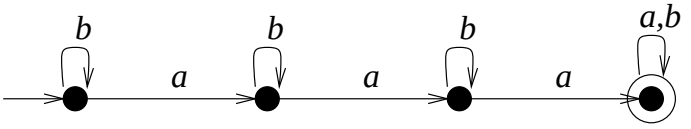
- Q to skończony zbiór *stanów*,
- Σ to skończony *alfabet wejściowy*,
- $\delta : Q \times \Sigma \to Q$ to *funkcja przejścia*; $\delta(q, a)$ to stan do którego przechodzimy ze stanu q na skutek wczytania znaku a,
- s to stan początkowy,
- $F \subseteq Q$ to zbiór stanów akceptujących.

□

Aby określić automat musimy podać informacje o wszystkich pięciu elementach. Możemy to zrobić formalnie, lub w postaci tabelki:

		a	b
$\rightarrow$	0	1	0
	1	2	1
	2	3	2
	$3F$	3	3

lub też diagramu:



Zaletą takich diagramów jest np. to, że nie ma konieczności nazywania stanów.

Dane dla automatu to dowolny napis $x \in \Sigma^*$. Opis działania: Zaczynamy w stanie początkowym i zmieniamy stany zgodnie ze znakami na wejściu. Po wczytaniu całego słowa jesteśmy w jakimś stanie. Słowo jest akceptowane, wtw., gdy stan końcowy jest akceptujący.

Przykład: Działanie powyższego automatu dla słów *baabbab* i *babbab*.

Def. 9

$$\delta^* : Q \times \Sigma^* \rightarrow Q$$

$$\delta^*(q, \varepsilon) = q$$

$$\delta^*(q, xa) = \delta(\delta * (q, x), a)$$

Automat *akceptuje słowo* x wtw., gdy $\delta^*(s, x) \in F$.
Język akceptowany przez automat, to:

$$L(M) \equiv \{x \in \Sigma^* : \delta^*(s, x) \in F\}$$

□

Przykład: Automat akceptujący $(a \mid b)^* aa(a \mid b)^*$. Stany reprezentują zliczane kolejne z rzędu literki a.

Przykład: Automat akceptujący $\{x \in \{0,1\}^* : x$ w układzie binarnym jest podzielne przez 3 $\}$. Stany reprezentują reszty modulo 3. Formalny dowód poprawności.

$$x0 = 2x \equiv (2(x \mathrm{mod} 3)) \mathrm{mod} 3$$

$$x1 = 2x + 1 \equiv (2(x \mathrm{mod} 2) + 1) \mathrm{mod} 3$$

3.3 Automaty niedeterministyczne[JFA,JAO]

3.3.1 Intuicje

„Rozklekotany" automat deterministyczny. Nie wiadomo dokładnie jaki ruch wykona, może wykonać *niedeterministycznie* jeden z wielu ruchów. Symulacja: diagram to plansza, po której przesuwamy pionki. Jeśli ze stanu, w którym jest pionek wychodzi wiele krawędzi oznaczonych symbolem wyjściowym, to stawiamy pionki we wszystkich stanach docelowych; jeżeli z danego stanu nie wychodzi żadna taka krawędź, to pionek znika. Podobnie możemy mieć więcej niż jeden stan początkowy — na początku symulacji ustawiamy pionki we wszystkich stanach początkowych.

Automat akceptuje słowo jeśli po zakończeniu symulacji w którymkolwiek ze stanów akceptujących będzie znajdować się pionek. Innymi słowy, automat akceptuje słowo jeśli *może* się zdarzyć takie obliczenie, które prowadzi do jego zaakceptowania.

Można o tym pomyśleć jak o przeplatających się dwóch procesach: zgadywania rozwiązania i jego weryfikacji.

Przykład: Nieformalny. Automat akceptujący słowa, w których przedostatni znak to 1. Symulacja dla słowa 0110010.

3.3.2 Definicja automatu

Def. 10 Automat niedeterministyczny, to piątka $M = \langle Q, \Sigma, \delta, S, F \rangle$, gdzie:

- Q to skończony zbiór stanów,
- Σ to skończony alfabet wejściowy,
- δ to funkcja przejścia $\delta : Q \times \Sigma \to P(Q)$ ($\cong$ relacja $\delta \subseteq Q \times \Sigma \times Q$),
- $S \subseteq Q$ to zbiór stanów początkowych,
- $F \subseteq Q$ to zbiór stanów akceptujących.

Rozszerzenie funkcji przejścia $\delta^* : P(Q) \times \Sigma^* \to P(Q)$ definiujemy następująco:

$$\delta^*(A, \varepsilon) = A$$

$$\delta^*(A, ax) = \delta^*(\delta(A, a), x)$$

Automat M akceptuje słowo x wtw., gdy $\delta^*(S, x) \cap F \neq \emptyset$.
Język akceptowany przez automat, to:

$$L(M) = \{x \in \Sigma^* : \delta^*(S, x) \cap F \neq \emptyset\}$$

□

Automaty deterministyczne możemy traktować jako szczególny przypadek automatów niedeterministycznych.

Fakt: Niech $M = \langle Q, \Sigma, \delta, s, F\rangle$ będzie automatem deterministycznym. Wówczas automat niedeterministyczny $M' = \langle Q, \Sigma, \delta', \{s\}, F\rangle$, gdzie $\delta'(q, a) = \{\delta(q, a)\}$ akceptuje dokładnie ten sam język

$$L(M) = L(M')$$

Kilka własności funkcji δ^*:

Lemat: Dla dowolnych $x, y \in \Sigma^*$ mamy $\delta^*(A, xy) = \delta^*(\delta^*(A, x), y)$.

Dowód: Przez indukcję ze względu na $|y|$.

Lemat: Funkcja δ^* jest rozłączna ze względu na sumowanie zbiorów, tzn.:

$$\delta^*(\bigcup_i A_i, x) = \bigcup_i \delta^*(A_i, x)$$

Dowód: Przez indukcję ze względu na $|x|$.

3.3.3 Determinizacja automatu

Pokażemy, że siła wyrazu automatów niedeterministycznych jest dokładnie taka sama, jak deterministycznych. W tym celu musimy pokazać, że dla każdego automatu niedeterministycznego istnieje równoważny mu automat deterministyczny.

Zauważmy, że w trakcie symulacji działania automatu niedeterministycznego nie ma znaczenia, czy na danym polu znajdzie się jeden, czy więcej pionków. Istotny jest jedynie zbiór pól, na których w danym kroku znajdują się pionki. Nasza konstrukcja będzie polegała na zbudowaniu automatu deterministycznego, którego stanami będą *zbiory* stanów automatu niedeterministycznego.

Przykład: Nieformalny. Automat akceptujący słowa, w których przedostatni znak jest równy 1. Stany nieosiągalne nie mają znaczenia na działanie automatu, więc można ograniczyć się do 4 stanów osiągalnych, co dokładnie odpowiada automatowi zapamiętującemu dwa ostatnie znaki w słowie.

Przykład: Nieformalny. Automat akceptujący słowa o długości podzielnej przez 3 lub 5. Jedyny niedeterminizm polega na wybraniu stanu początkowego. Automat deterministyczny będzie miał 256 stanów, choć tylko 15 będzie osiągalnych.

3.3.4 Konstrukcja automatu potęgowego

Niech $M = \langle Q, \Sigma, \delta, S, F\rangle$ będzie automatem niedeterministycznym. Jego determinizacją będziemy nazywać automat deterministyczny $M' = \langle P(Q), \Sigma, \delta', S, F'\rangle$, gdzie:

$$\delta'(A, a) = \delta^*(A, a)$$

$$F' = \{A \subseteq Q : A \cap F \neq \emptyset\}$$

Pokażemy, że oba automaty akceptują te same języki.

Lemat: Dla dowolnych $A \subseteq Q$ i $x \in \Sigma^*$ zachodzi

$$\delta^*(A, x) = \delta'^*(A, x)$$

Dowód: Indukcja ze względu na $|x|$.
Korzystając z tego lematu, dowód, że $L(M) = L(M')$ jest natychmiastowy.

3.4 Automaty z ε-przejściami

Możemy jeszcze rozszerzyć pojęcie automatu niedeterministycznego, wprowadzając tzw. ε-przejścia. Są to przejścia, które automat może w każdej chwili wykonywać, nie wczytując nic z wejścia.

Przykład: Automat akceptujący $a^*b^*c^*$. Porównanie z automatem deterministycznym.

Def. 11 Automat z ε-przejściami, to taka piątka $M = \langle Q, \Sigma, \delta, S, F\rangle$, gdzie:

- Q jest skończonym zbiorem stanów,
- Σ to skończony alfabet,
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$ to funkcja przejścia,

- $S \subseteq Q$ to zbiór stanów początkowych, oraz
- $F \subseteq Q$ to zbiór stanów akceptujących.

Przez $D(A)$ oznaczamy zbiór stanów, które można osiągnąć z A poprzez ε-przejścia. Rozszerzenie funkcji przejścia $\delta^* : P(Q) \times \Sigma \to P(Q)$ definiujemy następująco:

$$\delta^*(A, \varepsilon) = D(A)$$

$$\delta^*(A, ax) = \delta^*(\delta(D(A), a))$$

Język akceptowany przez automat z ε-przejściami to:

$$L(M) = \{x \in \Sigma^* : \delta^*(S, x) \cap F \neq \emptyset\}$$

□

Jak łatwo zauważyć, automaty bez ε-przejść są tylko szczególnym przypadkiem automatów z ε-przejściami. Pokażemy jednak, że ε-przejścia nie zwiększają siły wyrazu i że zawsze można je wyeliminować. Zauważmy, że z powyższej definicji δ^* wynika, że wystarczy w tym celu:

- zastąpić S przez $D(S)$,
- dla każdego $q \in Q$ i $a \in \Sigma$ zastąpić $\delta(q, a)$ przez $D(\delta(q, a))$.

Zauważmy, że:

$$D(\bigcup_i A_i) = \bigcup_i D(A_i)$$

czyli

$$\delta^*(A, xa) = \bigcup_{q \in \delta^*(A,x)} D(\delta(q, a))$$

Można pokazać, że

$$\delta^*(S, x) = \delta'^*(D(S), x)$$

3.5 Równoważność wzorców i automatów skończonych

Twierdzenie: Język A jest regularny wtw. gdy istnieje taki automat skończony M, że $A = L(M)$.

Dowód: Dowód rozbijemy na dwie implikacje:

1. Wiemy, że siła wyrazu wzorców i wyrażeń regularnych jest taka sama. Wystarczy więc, że pokażemy, jak dla wyrażenia regularnego α skonstruować automat skończony akceptujący $L(\alpha)$. Indukcyjnie po strukturze wyrażenia regularnego α z budujemy automat z ε-przejściami, o jednym stanie początkowym i akceptującym, który akceptuje słowa pasujące do wzorca.

   - ε, $\emptyset$, a — proste.
   - $\alpha \mid \beta$, α^*, $\alpha\beta$, — rysunek,

2. Dla danego deterministycznego automatu skończonego pokażemy wyrażenie regularne opisujące język akceptowany przez automat. Budujemy wyrażenia regularne opisujące fragmenty automatu: $\alpha^X_{u,v}$ opisuje te słowa, za pomocą których można przejść z u do v zatrzymując się po drodze jedynie w stanach z X. Wyrażenie $\alpha^X_{u,v}$ definiujemy indukcyjnie ze względu na $|X|$.

   (a) Oznaczmy przez $a_1, \ldots, a_k$ wszystkie krawędzie prowadzące z u do v. Wówczas:

   $$\alpha^{\emptyset}_{u,v} = \begin{cases} a_1 \mid \ldots \mid a_k \mid \varepsilon & ; \quad u = v \\ a_1 \mid \ldots \mid a_k & ; \quad u \neq v \end{cases}$$

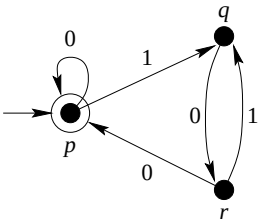
   (b) Niech $q \in X$.

   $$\alpha^X_{u,v} = \alpha^{X-\{q\}}_{u,v} \mid \alpha^{X-\{q\}}_{u,q} (\alpha^{X-\{q\}}_{q,q})^* \alpha^{X-\{q\}}_{q,v}$$

   Niech $F = \{f_1, \ldots, f_l\}$. Wówczas:

   $$L(M) = L(\alpha^Q_{s,f_1} \mid \ldots \mid \alpha^Q_{s,f_l})$$

Przykład: [K 9.3/53] Przekształcenie automatu w równoważne wyrażenie regularne.



Język akceptowany przez automat jest opisany wyrażeniem.

$$\alpha^{\{p,q,r\}}_{p,p} = \alpha^{\{p,r\}}_{p,p} \mid \alpha^{\{p,r\}}_{p,q} (\alpha^{\{p,r\}}_{q,q})^* \alpha^{\{p,r\}}_{q,p}$$

Patrząc na automat możemy wywnioskować:

$$\alpha^{\{p,r\}}_{p,p} = 0^*$$

$$\alpha_{p,q}^{\{p,r\}} = 0^*1$$

$$\alpha_{q,q}^{\{p,r\}} = \varepsilon \mid 01 \cup 000^*1 = \varepsilon \mid 00^*1$$

$$\alpha_{q,p}^{\{p,q,r\}} = 000^*$$

Stąd:

$$\alpha_{p,p}^{\{p,q,r\}} = 0^* \mid 0^*1(\varepsilon \mid 00^*1)^*000^*$$

Dalej, wyrażenie to można uprościć do $\varepsilon \mid (0 \mid 10)^*0$.

3.6 Własności domknięcia klasy języków regularnych

Jak już wcześniej powiedzieliśmy, jeżeli A i B są językami regularnymi, to regularne są również: AB, $A \cup B$ i A^*.

Fakt: Niech A i B będą językami regularnymi. Wówczas następujące języki są również regularne:

- $\overline{A}$,
- $A \cap B$,

Dowód:

- Niech M będzie deterministycznym automatem skończonym akceptującym A. Automat M' akceptujący $\overline{A}$ konstruujemy zmieniając w M zbiór stanów akceptujących na jego dopełnienie, czyli jeżeli:

  $$M = \langle Q, \Sigma, \delta, s_0, F \rangle$$

  to

  $$M' = \langle Q, \Sigma, \delta, s_0, Q \setminus F \rangle$$

- Z praw De Morgana mamy:

  $$A \cap B = \overline{\overline{A} \cup \overline{B}}$$

3.7 Podsumowanie

Reasumując, siła wyrazu: wzorców, wyrażeń regularnych, automatów deterministycznych, niedeterministycznych i z ε-przejściami jest dokładnie taka sama. Wszystkie one opisują dokładnie klasę języków regularnych.

3.8 Praca domowa

- Podaj deterministyczny automat skończony akceptujący słowa nad alfabetem $\{0, 1\}$ mające na początku 00.
- Podaj niedeterministyczny automat skończony akceptujący słowa nad alfabetem $\{0, 1\}$ mające na końcu 00.
- Podaj automat skończony z ε-przejściami akceptujący język opisany wyrażeniem regularnym $(a^*b^*) \mid (ab)*$.

Laboratorium

Ćwiczenie wyrażeń regularnych w Uniksie (bez Lexa):

- `grep` — przypomnienie działania, opcja `-E` i wyszukiwanie różnych wzorców,
- `sed` — krótko o idei działania, fragment funkcjonalności, skupiając się na poleceniu `s`,
- `awk` — można o nim opowiedzieć, ale to kobyła, więc tylko jeśli jest mnóstwo czasu na to.