Przetwarzanie tekstu 1

Podstawowe pojęcia teorii języków formalnych
Wyrażenia regularne

Języki formalne

alfabet

- alfabetem nazywamy skończony zbiór symboli
- oznaczenia: A, Σ
- przykłady:

$$
\begin{aligned}
A &= \{0, 1\} \\
A &= \{+, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
A &= \{a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z,\} \\
A &= \mathtt{ASCII} \\
A &= \mathtt{UTF\text{-}8}
\end{aligned}
$$

Języki formalne

słowo

- słowem nazywamy ciąg symboli alfabetu
- zbiór wszystkich słów nad alfabetem A oznaczamy A^*
- słowo puste (słowo długości 0) oznaczamy przez ε (epsilon)
- przykłady:

dla $A = \{0, 1\}$

$$
\begin{array}{lll}
\varepsilon & \in & A^* \\
0 & \in & A^* \\
1001101 & \in & A^* \\
021 & \notin & A^* \\
\texttt{Ala} & \notin & A^*
\end{array}
$$

Języki formalne

operacje na słowach

- operacje na słowach:

konkatenacja $\alpha \cdot \beta = \alpha\beta$

potęgowanie $\alpha^n = \underbrace{\alpha \cdot \alpha \cdot \alpha \cdot \ldots \cdot \alpha}_{n} = \underbrace{\alpha\alpha\alpha\ldots\alpha}_{n}$

- przykłady operacji na słowach:

$$\begin{aligned} a \cdot bb &= abb \\ a \cdot b^3 &= abbb \\ ab \cdot \varepsilon &= ab \\ (a \cdot b)^3 &= ababab \end{aligned}$$

- własności

$$\begin{aligned} \alpha \cdot \varepsilon &= \varepsilon \cdot \alpha = \alpha \\ \alpha^0 &= \varepsilon \end{aligned}$$

dla każdego słowa α

Języki formalne

język

- język, w lingwistyce formalnej, definiuje się jako zbiór słów nad pewnym alfabetem A (czyli podzbiór A^*).
- język pusty – nie zawierający żadnego słowa – oznaczany jest przez $\emptyset$ (jak zbiór pusty).
- przykłady języków nad alfabetem ASCII:

$L_1 = \{\mathtt{aa}, \mathtt{ab}, \mathtt{ac}, \mathtt{ba}, \mathtt{bb}, \mathtt{bc}, \mathtt{ca}, \mathtt{cb}, \mathtt{cc}\}$
$L_2 = \{\varepsilon, \mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \mathtt{aaaaa}, ...\}$
$L_3 =$ *zbiór wszystkich poprawnych identyfikatorów języka C*
$L_4 =$ *zbiór wszystkich poprawnych zdań języka angielskiego*

Języki formalne

podstawowe operacje na językach

suma	$L_1 \cup L_2 = \{\alpha \mid \alpha \in L_1 \vee \alpha \in L_2\}$
różnica	$L_1 \setminus L_2 = \{\alpha \mid \alpha \in L_1 \wedge \alpha \notin L_2\}$
przecięcie	$L_1 \cap L_2 = \{\alpha \mid \alpha \in L_1 \wedge \alpha \in L_2\}$
dopełnienie	$L^{-1} = A^* \setminus L$
konkatenacja	$L_1 \cdot L_2 = \{\alpha \cdot \beta \mid \alpha \in L_1, \beta \in L_2\}$
potęgowanie	$L^n = \underbrace{L \cdot L \cdot L \cdot \ldots \cdot L}_{n}$
domknięcie	$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots$

Języki formalne

podstawowe operacje na językach – przykłady

Niech $L_1 = \{a, b\}, \qquad L_2 = \{\varepsilon, c, cc\}$

$$
\begin{aligned}
L_1 \cup L_2 &= \{a, b, \varepsilon, c, cc\} \\
L_1 \cap L_2 &= \emptyset \\
L_1 \cdot L_2 &= \{a, ac, acc, b, bc, bcc\} \\
L_1^3 &= \{aaa, aab, aba, abb, baa, bab, bba, bbb\} \\
L_1^* &= \{\varepsilon, a, b, aa, ab, ba, bb, aab, aba, abb, baa, bab, bba, bbb, aaaa...\}
\end{aligned}
$$

Języki formalne

definiowanie języków

- języki definiuje się
  - przez wyliczenie słów
  - za pomocą wyrażeń regularnych
  - za pomocą automatów skończonych
  - za pomocą gramatyk formalnych

Języki formalne

literatura

- John Hopcroft, Jeffrey Ullman, *Wprowadzenie do teorii automatów, języków i obliczeń*, Wydawnictwo Naukowe PWN, 2003

Wyrażenia regularne

definicja

- wyrażeniem regularnym nad alfabetem A jest:

$\emptyset$	
ε	
a	*gdzie* $a \in A$

- jeśli r i s są wyrażeniami regularnymi, to są nimi też (w kolejności rosnących priorytetów operatorów):

$r \mid s$	*suma*
rs	*konkatenacja*
r^*	*domknięcie zwrotne*
(r)	

Wyrażenia regularne

język

- wyrażenie regularne definiuje/opisuje język
- język definiowany wyrażeniem regularnym r będziemy oznaczać $L(r)$

$$
\begin{aligned}
L(\ \emptyset\ ) &= \emptyset \\
L(\ \varepsilon\ ) &= \{\varepsilon\} \\
L(\ a\ ) &= \{a\}
\end{aligned}
$$

$$
\begin{aligned}
L(r|s) &= L(r) \cup L(s) \\
L(\ rs\ ) &= L(r) \cdot L(s) \\
L(\ r^*\ ) &= L^*(r) \\
L(\ (r)\ ) &= L(r)
\end{aligned}
$$

Wyrażenia regularne

języki regularne

- klasa języków, które da się zdefiniować za pomocą wyrażeń regularnych, nosi nazwę języków regularnych
- należą do tej klasy:
  - wszystkie języki skończone
  - język liczb, dat
  - zbiór poprawnych identyfikatorów języka C
- do tej klasy nie należą np.:
  - język wyrażeń arytmetycznych, język wyrażeń regularnych
  - język polski

Wyrażenia regularne

przykłady

r	$L(r)$
a	$\{a\}$
a|bcd	$\{a, bcd\}$
(a|b)cd	$\{acd, bcd\}$
(a|ε)bc	$\{abc, bc\}$
ab$*$	$\{a, ab, abb, abbb, abbbb, abbbbb, abbbbbb, ...\}$
$(ab)^*$	$\{\varepsilon, ab, abab, ababab, abababab, ababababab, ...\}$

Wyrażenia regularne

wyrażenia regularne w systemie Linux (Unix)

- odmiany wyrażeń regularnych w systemie Linux (Unix):

podstawowe (ang. basic)	przestarzałe, wychodzą z użycia	`\(+|-\)\?[0-9]+`
rozszerzone (ang. extended)	najpowszechniejsza odmiana	`(+|-)?[0-9]+`
wyrażenia regularne języka Perl	wypierają 'extended', stanowią ich rozszerzenie	`(+|-)?\d+`
wyrażenia regularne języka elisp (Emacs)	używane w edytorze Emacs	`\\(+\\|-\\)?[0-9]+`

Wyrażenia regularne w systemie Linux (Unix) /extended/

dodatkowe operatory

$+$	*domknięcie dodatnie*	$r+$	$= rr*$
?	*opcjonalność*	$r?$	$= (r \mid \varepsilon)$
$\{m,n\}$	*powtórzenie od m do n razy*	$r\{2,4\}$	$= rr \mid rrr \mid rrrr$
$\{m,\}$	*powtórzenie co najmniej m razy*	$r\{2,\}$	$= rr \mid rrr \mid rrrr \mid \ldots$
$\{,m\}$	*powtórzenie co najwyżej m razy*	$r\{,4\}$	$= \varepsilon \mid r \mid rr \mid rrr \mid rrrr$
$\{m\}$	*powtórzenie m razy*	$r\{4\}$	$= rrrr$

Wyrażenia regularne w systemie Linux (Unix) /extended/

zbiory znaków

`[ ]`	zbiór znaków, np.	`[abc] = (a|b|c)` `[a-d3-6] = (a|b|c|d|3|4|5|6)`
`[^ ]`	dopełnienie zbioru znaków, np.	`[^abc]` - dowolny znak poza a, b i c (i znakiem nowej linii)

Wyrażenia regularne w systemie Linux (Unix) /extended/

predefiniowane zbiory znaków

`[:alnum:]`	znak alfanymeryczny
`[:alpha:]`	litera
`[:cntrl:]`	znak sterujący
`[:digit:]`	cyfra
`[:graph:]`	znak drukowany widoczny
`[:lower:]`	mała litera
`[:print:]`	znak drukowany
`[:punct:]`	znak drukowany nie alfanumeryczny
`[:space:]`	znak odstepu
`[:upper:]`	wielka litera
`[:xdigit:]`	cyfra szesnastkowa

Wyrażenia regularne w systemie Linux (Unix) /extended/

symbole i sekwencje specjalne

- symbole specjalne
  - `.` dowolny znak poza znakiem końca linii
  - `^` początek linii/napisu
  - `$` koniec linii/napisu
- sekwencje specjalne
  - `\<` początek słowa
  - `\>` koniec słowa
  - `\b` granica słowa
  - `\B` miejsce nie będące na granicy słowa
  - `\n` znak nowej linii
  - `\t` znak tabulacji
- znaki wymagajace zabezpieczenia

  `\\ \^ \$ \. \[ \] \| \( \) \* \+ \- \? \{ \}`

Wyrażenia regularne w systemie Linux (Unix) /extended/

Dodatkowa funkcja nawiasów okrągłych

- fragment napisu dopasowany do podwyrażenia ujętego w nawias jest zapamiętywany i można się do niego dalej odwołać
- w dalszej części tego samego wyrażenia regularnego odwołanie takie ma postać sekwencji \$n$, gdzie n jest numerem podwyrażenia , poza nim – w zależności od narzędzia/języka – zwykle $1, $2, ... (np. w językach Perl, Ruby)
- podwyrażenia numerowane są od 1 w górę zgodnie z kolejnością nawiasów otwierających
- znaczenie takiego odwołania jest następujące: w miejscu odwołania musi się pojawić taki sam napis jak ten odpowiadający podwyrażeniu, do którego się odwołujemy
  np.

$$L(\ \texttt{(a|b|c)\1}\ ) = \{\texttt{aa}, \texttt{bb}, \texttt{cc}\}$$
$$L(\ \ \texttt{(a*)b\1}\ \ ) = \{\texttt{b}, \texttt{aba}, \texttt{aabaa}, \texttt{aaabaaa}, \texttt{aaaabaaaa}, ...\}$$

Wyrażenia regularne w systemie Linux (Unix) /extended/

przykłady

- liczba całkowita
  `(+|-)?[0-9]+`
- kod pocztowy
  `[0-9]{2}-[0-9]{3}`
- imiona z nazwiskiem
  `([[:upper:]]([[:lower:]]+|\.) )+[[:upper:]][[:lower:]]+`
- palindrom literowy o długości 5
  `([[:alpha:]])([[:alpha:]])[[:alpha:]]\2\1`
- stała napisowa ujęta w cudzysłowy, mogąca zawierać wewnątrz znaki cudzysłowu poprzedzone odwrotnym ukośnikiem
  `"([^"]|\\")*"`

Wyrażenia regularne w systemie Linux (Unix)

- więcej szczegółów – patrz dokumentacja, np. `info grep`