Регулярные выражения

Мати Рейнович Пентус, Алексей Андреевич Сорокин

МГУ им. М. В. Ломоносова
весенний семестр 2022/2023 учебного года
Межфакультетский курс
“Введение в компьютерную лингвистику”

Регулярные выражения языка Perl

Найти самое короткое регулярное выражение, эквивалентное данному регулярному выражению.

Ф1. `[a-z]+(?:-[a-z]+)*`

Ф2. `[aou][b-np-tv-z]*[aou][a-z]*`

Ф3. `-(?:\d|--)*-`

Ф4. `\w\s(?:\w\w?\s)*\w?`

Ф5. `[^q]*(?:qu[^q]*)*`

Пример.
Регулярное выражение `wh(at|en|ich)ever` принимает слово **whatever**.
Иными словами, слово **whatever** соответствует регулярному выражению `wh(at|en|ich)ever`.

Корпуса I

Многие языковые корпуса используют CQP (Corpus query processor). Язык запросов для CQP обычно называют CQL (Corpus query language).

```
1. Aranea Web Corpora
http://aranea.juls.savba.sk/
Guest Access
English Araneum Anglicum II   Minus (125 M)
Query type: CQL
[word="[Ii].*"] [word=".*i.*i.*"]{4,} [word="[Ii].*"]

2. CorpusEye English corpus
https://corp.hum.sdu.dk/cqp.en.html
Select an English corpus: Wikipedia A
cqp-speak
[word="St.*g"] [pos="V\b.*"]{2}
[word="he....he"]
```

Корпуса II

3. Leeds collection of Internet corpora
http://corpus.leeds.ac.uk/ruscorpora.html
Russian National Corpus (modern, MSD)
CQP syntax only
[word=".+[сп][пт]р.+"]{4,}
[lemma="тот"] [lemma="кто"]

4. Wikipèdia en català: powered by CQPweb
http://corptedig-glif.upf.edu/cqpweb/wikica/
Query mode: CQP syntax
[word="mi.*ns" & POS=".*\bN.*"]

Perl и Python

В командной строке Windows можно набрать

```
perl -e "while('03/15-03/22' =~ m#(\d\d)/(\d\d)#g) {print qq($2.$1\n);}"
```

или

```
python -c "import re; [print(f'{m[2]}.{m[1]}')
                    for m in re.finditer(r'(\d\d)/(\d\d)', '03/15-03/22')]"
```

Так мы в тексте **03/15-03/22** ищем куски, соответствующие шаблону `(\d\d)/(\d\d)`, и печатаем в другом формате (каждую находку в отдельной строке). Например, **03/15** превращается в **15.03** перед выводом.

Чтобы искать то же в файле abc.txt, можно набрать

```
perl -e "open F, '<abc.txt'; $_ = join('', <F>);
                            while(m#(\d\d)/(\d\d)#g) {print qq($2.$1\n);}"
```

или

```
python -c "import re; [print(f'{m[2]}.{m[1]}')
          for m in re.finditer(r'(\d\d)/(\d\d)', open('abc.txt').read())]"
```

Perl и Python

Регулярные выражения можно также отлаживать на многих сайтах.

https://regex101.com/
http://myregexp.com/
https://regexr.com/
https://www.regextester.com/
https://www.freeformatter.com/regex-tester.html

Документация на русском

https://habr.com/ru/articles/545150/
https://www.php.net/manual/ru/reference.pcre.pattern.syntax.php
...

Формальные языки

Алфавитом называется любое непустое конечное множество.
Пусть зафиксирован какой-нибудь алфавит.
Элементы алфавита называются *символами*.
Конечная последовательность символов называется *словом* (или *строкой*).
Множество, состоящее из слов называется *формальным языком*.
Пример. Рассмотрим алфавит $\{a, b, d, e, n\}$.

- b является символом.
- Последовательность bda является словом.
- Слова add и dad разные.
- Множество $\{da, de, en, ne\}$ является формальным языком.
- Множество $\{a, aab, aaabb, aaaabbb, \ldots\}$ является формальным языком.
- Множество $\varnothing$ является формальным языком. В нём нет ни одного слова.
- Слово длины ноль часто обозначают через ε (или λ).
- Множество $\{\varepsilon\}$ является формальным языком. В нём только одно слово.

Регулярные выражения в теории формальных языков

Каждое регулярное выражение задаёт некоторый язык. Он состоит в точности из тех слов, которые соответствуют этому выражению. Например, `wh(at|en|ich)ever` задаёт язык $\{whatever, whenever, whichever\}$.

- Базовые регулярные выражения: элементы алфавита.
- Также есть константы 0 (пустой язык) и 1 (язык, содержащий только пустое слово ε).
- Двуместные операции: $|$ (объединение) и $\cdot$ (конкатенация): если K и L — языки, то $K \cdot L$ состоит из слов вида uv, где слово u принадлежит языку K и слово v принадлежит языку L.
- Одноместная операция $*$ (итерация, взять любое количество раз): если L — язык, то L^* состоит из слов вида $u_1 \dots u_r$, где r — натуральное число и слова $u_1, \dots, u_r$ принадлежат языку L.
- Например, выражение $(ab)^*$ задаёт язык $\{\varepsilon, ab, abab, ababab, \dots\}$.
- Например, выражение $(a|b)^*$ задаёт язык $\{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$.
- Приоритет операций: итерация, конкатенация, объединение. При этом значок конкатенации можно опускать.

Язык регулярный, если он задаётся регулярным выражением.

Примеры регулярных языков

- Все слова в алфавите $\{a, b\}$: $(a|b)^*$,
- Слова в алфавите $\{a, b, c\}$, где предпоследняя буква — b: $(a|b|c)^*b(a|b|c)$.
- Слова в алфавите $\{a, b, c\}$, содержащие ровно 2 буквы a: $(b|c)^*a(b|c)^*a(b|c)^*$.
- Слова нечётной длины в алфавите $\{a, b\}$: $((a|b)(a|b))^*(a|b)$.
- Слова в алфавите $\{a, b, c\}$, содержащие чётное число букв a: $((b|c)^*a(b|c)^*a)^*(b|c)^*((b|c)^*a(b|c)^*a)^*(b|c)^*$.
- Слова в алфавите $\{a, b, c\}$, где перед a идёт только b: $((b|c)^*ba)^*(b|c)^*((b|c)^*ba)^*(b|c)^*$.
- Непустые слова в алфавите $\{a, b\}$, в которых одинаковые буквы не идут подряд: $(b|1)(ab)^*(a|1)(b|1)(ab)^*(a|1)$.
- Слова в алфавите $\{a, b, c\}$, в которых одинаковые буквы не идут подряд: $(H|1)(cH)^*(c|1)$, где H — ответ на предыдущий пункт.

Регулярные выражения языка Perl

- `wh(at|en|ich)ever` принимает **whichever**
- `n*i` принимает **nnni**
- `in*` принимает **innn**
- `(in)*` принимает **ininin**

Регулярные выражения языка Perl

```
import re
text = "vizitis kelkajn urbojn, inter kiuj estis mia urbo"

ms = re.search(r"(a|e|o|j)*n", text)
if ms:
    print(ms[0])
print(re.sub(r"urbo(j|)(n|)", r"granda\1\2 urbo\1\2", text))
for mf in re.finditer(r"(a|o|j)*n", text):
    print(mf[0], end="/")
```

Эта программа напечатает следующее.

```
ajn
vizitis kelkajn grandajn urbojn, inter kiuj estis mia granda urbo
ajn/ojn/n/an/n/
```

Регулярные выражения языка Perl

- `d.n` принимает **dan**
- `gr[ae]y` эквивалентно `gr(a|e)y` эквивалентно `gray|grey`

  Упражнение. Регулярные выражения `(da|e)na|ene` и `dana|en[ae]` эквивалентны или нет?

- `[^,;]` принимает однобуквенные слова, кроме `,` и `;`
- `[a-f]` эквивалентно `[abcdef]` эквивалентно `(a|b|c|d|e|f)`
- `[- ,]` эквивалентно `(-| |,)`
- `\n` принимает символ новой строки
- `///\n\n///` принимает

  ```
  ///


  ///
  ```

- `[^a]|a` эквивалентно `(.|\n)`

Регулярные выражения языка Perl

- `\s` принимает любой пробельный символ (в том числе пробел и символ новой строки)
- `\d` принимает любую цифру
- `\w` принимает любую букву, цифру, а также символ подчёркивания
- `re.sub(r"(\d\d)\.(\d\d)\.(\d\d\d\d)", r"\3-\2-\1", text)` заменяет **31.12.2019** на **2019-12-31**
- `[\s\da-f]*` принимает **00b5**
- `[-\w]*` принимает **six-year-old**
- `\S` эквивалентно `[^\s]`
- `\D` эквивалентно `[^\d]`
- `\W` эквивалентно `[^\w]`
- `\|` эквивалентно `[|]` эквивалентно `\N{VERTICAL LINE}`
- `\.` эквивалентно `\u002e` эквивалентно `\u002E` эквивалентно `[\56]`
- `\\` принимает `\`

Регулярные выражения языка Perl

- `(da){4}` эквивалентно `dadadada`
- `da{4}` эквивалентно `d(a{4})` эквивалентно `daaaa`
- `(da){2,4}` эквивалентно `dadadada|dadada|dada`
- `(da){2,}` принимает **dadadadada**
- `(da){,4}` эквивалентно `(da){0,4}`
- `(da)?` эквивалентно `(da){,1}` эквивалентно `(da|)`
- `da?` эквивалентно `d(a?)` эквивалентно `(da|d)`
- `(da)*` эквивалентно `(da){0,}`
- `(da)+` эквивалентно `(da){1,}`

Упражнение. Регулярные выражения `[a-z]+(-[a-z]+)*` и `[a-z](-?[a-z])*` эквивалентны или нет?

Регулярные выражения языка Perl

- `(da){2,4}?` эквивалентно `dada|dadada|dadadada`
- `.{1,3}ada` эквивалентно `...ada|..ada|.ada`
- `.{1,3}?ada` эквивалентно `.ada|..ada|...ada`
- `<.*>` находит в `<br><image>` подслово `<br><image>`
- `<.*?>` находит в `<br><image>` подслово `<br>`
- `(da)??` эквивалентно `(|da)`
- `(da)+?` эквивалентно `da(da)*?`

Регулярные выражения языка Perl

- `re.sub(r"<person>(\w+) (\w+) (\w+)</person>", r"<person>\3, \1 \2 (\3, \1)</person>", text)`
  заменяет **<person>Ян Ильич Сом</person>**
  на **<person>Сом, Ян Ильич (Сом, Ян)</person>**
- `re.sub(r"(wh(at|ich) (\w+))", r"\1 (\3)", text)`
  заменяет **which name** на **which name (name)**
- `re.sub(r"(wh(?:at|ich) (\w+))", r"\1 (\2)", text)`
  заменяет **which name** на **which name (name)**
- `re.finditer(r"^\d", "5-1=4")` находит только подслово **5**
- `re.finditer(r"\d$", "5-1=4")` находит только подслово **4**
- `print(re.sub(r"(\d??)(\d?)(\d)", r"\1;\2;\3", "89<357"))`
  напечатает `;8;9<3;5;7`
- `re.finditer(r"\b[A-Z]+\b", "James A. Michener")`
  находит только подслово **A**

Обратные ссылки

В регулярных выражениях языка Perl можно использовать обратные ссылки. Такие выражение не эквивалентны математическим регулярным выражениям.

- `\w*(\w{2,})\1\b` принимает **queue** и **singing**

Из-за обратных ссылок проблема эквивалентности регулярных выражений языка Perl неразрешима.

Примеры регулярных выражений

Пусть $\Sigma = \{C, V, \overline{V}, -\}$ (согласный, безударный гласный, ударный гласный, слогораздел).

- Корректное разбиение на слоги:
  - В каждом слоге ровно одна гласная: $C^*(V|\overline{V})C^*$.
  - Ровно один слог ударный.
  - Пусть X — ударный слог, Y – безударный, тогда искомое выражение $(Y-)^*[(X-(Y-)^*Y)|X]$.
  - Эквивалентно $(Y-)^*X(-Y)^* = (C^*VC^*-)^*C^*\overline{V}C^*(-C^*VC^*)^*$.
- Разбиение слова на слоги, содержащее ровно 1 открытый слог (ударность не учитывается). $(C^*VC^+-)^*(C^*V)(-C^*VC^+)^*$
- “Гармония гласных” (гласные типа V_1 и V_2 не встречаются вместе): $(C|V)^*(V_1(C|V_1|V)^*|V_2(C|V_2|V)^*)$