Reguläre Ausdrücke

by

Dr. Günter Kolousek

Allgemeines

- Eine Sprache ist eine Menge von Wörtern
  - Frage: Was ist ein Wort?
- Reguläre Ausdrücke (engl. regular expressions, kurz: regex) dienen dazu Sprachen zu beschreiben.
  - eigentlich: ganz bestimmte Arten von Sprachen
- Eine Menge (von Wörtern), die durch einen regulären Ausdruck beschrieben wird, heißt reguläre Menge.
- Anwendungen: Suchen von Strings (z.B. vim, Emacs, grep,…), lexikalische Analyse ($\rightarrow$ Compiler),…

Definition regulärer Mengen

Rekursive Definition!

Ein regulärer Ausdruck ist…

- das Zeichen ϵ (Epsilon), das die Menge kennzeichnet, die den Leerstring als einziges Element enthält. ϵ bezeichnet die leere Zeichenkette.
  - Menge: $\{\epsilon\}$
- ein einzelnes Zeichen **a** (aus dem Alphabet) ist ein regulärer Ausdruck, der die Menge kennzeichnet, die nur das Wort a enthält.
  - Menge: $\{a\}$

Definition regulärer Mengen – 2

Wenn r und s reguläre Ausdrücke sind, dann sind auch die folgenden Ausdrücke regulär (Operationen):

1. Oftmalige Verkettung:
   - r^n … genau n-Mal r
     z.B.: regex: $r^2 = rr$, Menge: $\{rr\}$, d.h. die Menge, die das Wort rr enthält (siehe nächster Punkt)
   - r^* … 0,1 oder n-Mal r, d.h. ϵ oder Verkettung von beliebig vielen r: $\{\epsilon, r, rr, rrr, \ldots\}$
   - r^+ … 1 oder n-Mal r oder Verkettung von beliebig vielen r mindestens jedoch ein r: $\{r, rr, rrr, \ldots\}$
2. rs … Verkettung: r und s
3. $r|s$ … Vereinigung: r oder s

Definition regulärer Mengen – 3

- Operatorreihenfolge (Prioritäten) gemäß der angegebenen Reihenfolge:
  - oftmalige Verkettung vor einer Verkettung
  - Verkettung vor einer Vereinigung

  D.h. wie in der Mathematik üblich:
  - Potenzrechnung vor Punktrechnung
  - Punktrechnung vor Strichrechnung
- Arten von Zeichen
  - Terminalzeichen: gewöhnliche Zeichen
  - Metazeichen: Zeichen, die Teil von Beschreibungssprache

Beispiele regulärer Ausdrücke

- Wörter gegeben durch folgende Menge $\{a, b\}$
  - $a|b$
- Wörter gegeben durch folgende Menge $\{aa, ab, ba, bb\}$
  - $(a|b)^2$
- Wörter über dem Alphabet $\{a, b\}$, die mit einem a beginnen
  - $a(a|b)^*$
- Wörter über dem Alphabet $\{a, b, c\}$, die aus einer beliebigen Folge von a und b beginnen (mindestens Länge 1) und optional mit einem c enden können:
  - $(a|b)^+(c|\epsilon)$
- Bezeichner einer Sprache
  - $(a|b|..|z|A|B|..|Z)(a|b|..|Y|Z|0|..|9)^*$

Reguläre Definition

- andere Schreibweise für reguläre Ausdrücke, um leichter kompliziertere Muster zu beschreiben.
- besteht aus einer Reihe von Definitionen der Form:
  $d_1 \rightarrow r_1$
  $d_2 \rightarrow r_2$
  ...
  $d_n \rightarrow r_n$
- $d_1, d_2, \ldots$ Namen von regulären Ausdrücken
- $r_1, r_2, \ldots$ die regulären Ausdrücke selbst.

Anwendungen

Achtung: leicht verschiedene Syntax je Tool!

- `grep`
  - sucht nach Zeichenketten mittels regex
    - Unix-Kommandos verwenden oft regex!
  - gibt alle Zeilen aus, die gefundene Zeichenketten enthalten
  - beherrscht sowohl die BRE als auch die ERE.
  - BRE … basic regular expression: Metazeichen `?`, `+`, `{`, `|`, `(` und `)` sind mit `\` zu maskieren, um ihre ”Metafunktionalität” zu erhalten.
  - ERE … extended regular expression: Metazeichen habe ihre ”normale” Metafunktionalität.
- Programmiersprachen: Python, Perl, PHP, Java,…
- Editoren: Eclipse, Emacs, Sublime,…

Basiskonstrukte

- `.` … ein beliebiges Zeichen
- `^` … Anfang der Zeile
- `$` … Ende der Zeile
- Folgende Metazeichen werden mit `\` maskiert.
  - z.B.: `\[` oder `\$`
  - aber viele Zusatzbedeutungen:
    - `\A` … Anfang des Strings, `\Z` … Ende des Strings
    - `\d` … ein Ziffern-Zeichen (in ASCII `[0-9]`) (Gegenteil davon: `\D`) (nicht in `grep`)
    - `\w` … ein Wort-Zeichen (in ASCII `[a-zA-Z0-9_]`) (Gegenteil davon: `\W`)
    - `\s` … ein Whitespace-Zeichen (Gegenteil davon: `\S`) (nicht in `grep`)
    - `\b` … ein leerer String am Anfang oder Ende eines Wortes (Gegenteil davon: `\B`), z.B.: `\botto\b` findet `otto`, `otto.`, `(otto)`, `mini otto maxi` aber nicht `ottomaxi` or `otto42`

Zeichenklassen

- `[abc]` … eines der Zeichen a, b oder c
  - `[a-c]` … eines der Zeichen a, b oder c
  - `[-abc]`, `[abc-]` oder `[a\-bc]` … eines der Zeichen a, b, c oder -
  - `[]abc]` oder `[abc\]]` … eines der Zeichen a, b, c oder]
  - `[(+*)]` … eines der Zeichen (, +, * oder)
- `[^abc]` … ein Zeichen aber weder a noch b noch c
  - `[abc^]` … aber: eines der Zeichen a, b, c oder ^
- Weitere Bedeutungen *innerhalb* von `[]`:
  - `[:alnum:]`, `[:alpha:]`, `[:digit:]`,
  - `[:space:]`, `[:upper:]`, `[:lower:]`

Zusammensetzungen

- `r?` … optional (greedy, d.h. gierig)
- `r*` … beliebig (greedy)
- `r+` … mind. ein Mal (greedy)
- `r??, r*?, r+?` … nicht greedy!
- `r{n}` … genau n Mal
- `r{n,}` … mind. n Mal
- `r{,n}` … höchstens n Mal
- `r{m,n}` … m bis n Mal
- `|` … oder
- `( )` … runde Klammern bilden Gruppe
  - `\n` … Zugriff auf Inhalt der n.ten Gruppe
  - `(.+) \1` … z.B.: ”maxi maxi” oder ”42 42”
  - Gruppe ohne ”capture”: `(?:regex)`, z.B.: `(?:42)`
    - nicht in POSIX, grep, GNU

Präfix und Postfix

wird zur Suche herangezogen ist jedoch nicht im Ergebnis enthalten

- Präfix `(?<=regex)`
  - `(?<=[A-Z]+)[0-9]+`
    - `ALPHA42` → true
    - `alpha42` → false
- Postfix `(?=regex)`
  - `[0-9]+(?=[A-Z]+)`
    - `42OMEGA` → true
    - `42omega` → false

Beispiel mittels grep

```
grep -Ein "ko[[:digit:]]" *.c
```

- E … ERE
- i … ignore case
- n … show line numbers
- o … zeigt nur die gefundenen Zeichenketten (nicht ganze Zeilen)

D.h. es werden alle Vorkommnisse der Form `ko1` oder `KO001` in C - Source-Dateien gesucht und die Ergebnisse mit Zeilennummern ausgegeben.

Beispiel in Python

```
import re

result = re.fullmatch(pattern="[A-Z]{2}",
                      string="AT")

if result:
    print("match")
else:
    print("Doesn't match")
```

Beispiele 1

- Menge der Zeichenketten aus Nullen und Einsen, die mindestens ein Paar aufeinanderfolgender Einsen enthält.
- Gesucht sind 5 Wörter des folgenden regulären Ausdruckes: $(c|d)\ (d|e)^*$
- Menge der Zeichenreihen aus Nullen und Einsen, deren zehntes Symbol von rechts eine Eins ist.
- Menge von Zeichenketten über dem Alphabet {a,b,c}, die mindestens ein a und mindestens ein b enthalten.
- ganze Zahlen oder Dezimalzahlen in der üblichen Notation darstellt:
  - Vorzeichen optional
  - danach mindestens eine Ziffer
  - danach kann ein Komma kommen. Wenn Komma dann jedoch mindestens eine Ziffer.

Beispiele 2

- Datumsformat, z.B.: 2005-OKT-06 oder 2005-10-06 (nicht jedoch 2004-10-54).
- Nullen und Einsen derart, dass alle Paare aufeinanderfolgender Nullen vor allen Paaren aufeinanderfolgenden Einsen stehen.
- Menge der Zeichenketten aus Nullen und Einsen, deren Anzahl von Nullen durch 5 teilbar ist.
- Zeichenketten, die mit einer 1 beginnen und danach beliebig viele Zeichen haben können, jedoch immer wechselt sich eine 0 mit einer 1 ab. Also: 1 oder 10 oder 101…
- Nullen und Einsen, die die Teilzeichenkette 101 nicht enthalten.
- Üben und Testen: $\rightarrow$ - `https://regex101.com/`