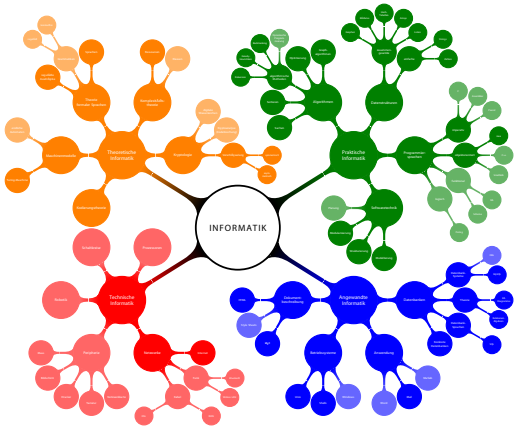
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Kapitel 5

Reguläre Ausdrücke

Pattern Matching

Vorlesung Einführung in die Informatik 1 vom 7. November 2013 von Till Tantau



Lernziele von Kapitel 5

1. Konzept des regulären Ausdrucks verstehen
2. Probleme als reguläre Ausdrücke beschreiben können
3. Mächtigkeit regulärer Ausdrücke kennen

Gliederung von Kapitel 5

Worum geht es bei der Mustersuche?

Das Problem, ein *Muster* innerhalb eines Textes zu finden, taucht in vielen Kontexten auf:

- Man sucht in einem elektronischen Dokument nach dem Vorkommen eines bestimmten Wortes; beispielsweise das Wort »Energiewende« in einem Koalitionsvertrag.
- Eine Suchmaschine möchte anzeigen, wo überall auf einer Webseite das Wort »flauschig« vorkommt.
- Man sucht in einer DNA-Sequenz nach allen Vorkommen einer bestimmten Basensequenz.

Das gesuchte Muster ist aber in der Regel *kein einzelnes Wort*:

- Statt »flauschig« würde man auch »Flauschig« finden wollen. Eventuell auch »FLAUSCHIG«.
- Sucht man die Basensequenz eines Gens, so würde man an Polymorphismus-Stellen auch zulassen, dass dort im Text vom Muster abgewichen wird.

Was ist ein Muster?

Muster und Muster-Suche

- Ein *Muster* (englisch *pattern*) *beschreibt* eine *Menge von möglichen Worten*.
- Bei der *Muster-Suche* (englisch *pattern search*) sind ein Muster und ein (längerer) Text gegeben.
  Ziel ist es, *Stellen im Text zu finden*, an denen ein Wort steht, das vom Muster beschrieben wird.

Beispiel

Wir werden bald definieren, dass das Muster `ab[cd]` die beiden Strings `abc` und `abd` beschreibt.
Würde man nach `ab[cd]` in `Ich werd' das abchecken.` suchen, so gäbe es genau einen Treffer, nämlich am Anfang von `abchecken`.

Was im Folgenden passiert.

- Es werden *reguläre Ausdrücke* eingeführt; dies ist eine Art, Muster aufzuschreiben.
- Es wird erklärt, welche Mengen an Wörtern ein regulärer Ausdruck beschreibt.
- Es wird *nicht* erklärt, wie die Algorithmen funktionieren, die nach Ausdrücken in Texten suchen.
  Es sei lediglich erwähnt, dass es *sehr schnelle* Algorithmen gibt und dass man *mehrere Kapitel über Theoretische Informatik* braucht, um diese wirklich zu verstehen.

Wörter im Sinne der Theorie

Definition

1. Ein *Alphabet* ist eine nichtleere, endliche Menge.
2. Die Elemente eines Alphabets nennt man *Buchstaben* oder auch *Symbole*.
3. Eine *endliche Folge* von Buchstaben nennt man ein *Wort*.

Beachte:

- Alphabete werden häufig mit griechischen Großbuchstaben bezeichnet, also Γ oder Σ.
  Praktische Beispiele sind die Menge $\{0, 1\}$ (bei Informatikern beliebt), die Menge $\{A, C, G, T\}$ (bei Biologen beliebt) und die Zeichenmenge des UNICODE.
- In »Wörter« können Leerzeichen (!) als Symbole auftauchen.
- Es gibt auch ein *leeres Wort*, abgekürzt ε oder λ, das Länge Null hat.

Die vornehme Bezeichnung für Mengen von Wörtern: Sprachen.

Definition

Eine *Sprache* ist eine Menge von Wörtern über einem Alphabet.

- Sprachen müssen weder sinnvoll noch interessant sein,
- sie können endlich sein oder auch unendlich.
- Ein Beispiel einer endlichen Sprache ist $\{AAA, AAC, AAT\}$;
- ein Beispiel einer unendlichen Sprache die Menge aller Basensequenzen, die `TATA` genau zweimal enthalten.

Muster-Suche, noch einmal etwas formaler.

- Ein *regulärer Ausdruck* wird ein *einzelnes Wort* sein, wie `ab[cd]` oder `x|y|z*`.
- Wir werden für jeden regulären Ausdruck R eine bestimmte Sprache $L(R)$ angeben, von der wir sagen werden, dass *R sie beschreibt*.
  Beispielsweise wird $L(\texttt{ab[cd]}) = \{abc, abd\}$ gelten.
- Das *Muster-Such-Problem* ist dann, zu einem gegeben regulären Ausdruck R und einem längeren Wort w ein Teilwort u in w zu finden mit $u \in L(R)$.

Das Programm Grep.

- Das Programm Grep löst das Muster-Such-Problem,
- wobei GREP für »Global search for REgular Patterns« steht.
- Parameter sind
  1. ein regulärer Ausdruck und
  2. der Name einer Datei, in der nach Vorkommen des Musters gesucht werden soll.
- Findet Grep das Muster in einer Datei, so gibt es die Zeile aus, in der das Muster vorkommt;
  mit der Option `-n` kann man sich die Zeilennummer auch anzeigen lassen.
- Aus historischen Gründen sollte man auch immer die Option `-E` angeben, damit die »moderne« Syntax für reguläre Ausdrücke benutzt wird.

Beispiel

```
maus:media tantau$ grep -n -E "Philosophie" faust.txt
536:Habe nun, ach! Philosophie,
maus:media tantau$
```

Die einfachsten regulären Ausdrücke sind einfache Wörter.

Definition (Reguläre Ausdrücke 1 – Atomare Ausdrücke)

Sei Σ ein Alphabet, in dem bestimmte Sonderzeichen nicht vorkommen. Dann ist jedes Wort w über diesem Alphabet ein regulärer Ausdruck, im Folgenden mit dem Buchstaben R abgekürzt.
Die von R beschriebene Sprache $L(R)$ ist einfach $\{w\}$, sie enthält also einfach nur w.

- Für diese einfachen Ausdrücke ist der formale Apparat mit Wörtern und Sprachen und $L(R)$ natürlich Overkill.
- Trotzdem ist schon hier das *Muster-Such-Problem* interessant: Es ist einfach das Problem, in einem Text ein ganz bestimmtes Wort zu finden.

Beispiele


```
maus:media tantau$ grep -n -E "heilig" faust.txt
1799:Knurre nicht Pudel! Zu den heiligen Tönen,
1823:Das heilige Original
1951:Ich versenge dich mit heiliger Lohe!
4430:Und hier mit heilig reinem Weben
4527:           Und warf den heiligen Becher
4595:Ob das Ding heilig ist oder profan;
4808:Bey'm heiligen Antonius,
5979:Ihr selig machend ist, sich heilig quäle,
7902:Was will der an dem heiligen Ort?
7917:Ihr Engel! Ihr heiligen Schaaren,
maus:media tantau$
```

Definition (Reguläre Ausdrücke 2 – Alternativen)

Sind R_1 und R_2 beides reguläre Ausdrücke, so auch $R_1 \mid R_2$.
Die von $R_1 \mid R_2$ beschriebene Sprache $L(R_1 \mid R_2)$ ist die Vereinigung der Sprachen $L(R_1)$ und $L(R_2)$:

$$L(R_1 \mid R_2) = L(R_1) \cup L(R_2)$$

- Ein senkrechter Strich kann also genutzt werden, um *alternativ* nach den einen Wörtern oder nach den anderen Wörtern zu suchen.
- Der senkrechte Strich gehört zu den *Sonderzeichen*. Will man tatsächlich nach einem senkrechten Strich suchen, so muss man ihm einen Backslash voranstellen.

Beispiele


```
maus:media tantau$ grep -n -E
    "Philosophie|Theologie|Medicin" faust.txt
536:Habe nun, ach! Philosophie,
538:Juristerey und Medicin,
539:Und leider auch Theologie!
2996:Fast möcht' ich nun Theologie studiren.
3029:Wollt ihr mir von der Medicin
3042:Der Geist der Medicin ist leicht zu fassen;
maus:media tantau$
```

Erst dies, dann das. . .

Definition (Reguläre Ausdrücke 3 – Konkatenation)

Sind R_1 und R_2 beides reguläre Ausdrücke, so auch R_1R_2.
Die von R_1R_2 beschriebene Sprache $L(R_1R_2)$ ist die so genannte *Verkettung* der Sprachen $L(R_1)$ und $L(R_2)$:

$$L(R_1R_2) = \{uv \mid u \in L(R_1), v \in L(R_2)\}.$$

- Die Sprache $L(R_1R_2)$ enthält alle Wörter w, die man irgendwie in zwei Teile $w = w_1w_2$ trennen kann, so dass der *Präfix* in $L(R_1)$ und der *Suffix* in $L(R_2)$ liegt.
- Man kann *runde Klammern* benutzen, wenn Verkettungen und Alternativen gemischt werden, um die Reihenfolge der Auswertung vorzugeben.
- Wieder kann man auch nach Klammern selbst suchen, wenn man einen Backslash voranstellt.

Beispiele


```
maus:media tantau$ grep -n -E "(Phi|Theo)lo(soph|g)ie"
    faust.txt
536:Habe nun, ach! Philosophie,
539:Und leider auch Theologie!
2996:Fast möcht' ich nun Theologie studiren.
maus:media tantau$
```

Zur Übung

Wie lautet $L(R)$ jeweils für folgende Ausdrücke R?

1. `(0|1|2)(3|4|5)`
2. `0|((1|2|3)3(4|5))(6|7)`
3. `(a||b)(c|d)`
4. `\||(\||\()(\)|\\|\|)`
   (Keine Angst, so etwas kommt zwar in Computerprogrammen, aber nicht in Klausuren vor.)

Dies oder das oder dies oder das. . .

Definition (Reguläre Ausdrücke 4 – Wiederholungen)

Ist R ein regulärer Ausdruck, so auch R^*.
Die von R^* beschriebene Sprache $L(R)$ ist der so genannte *Kleene-Stern* von $L(R)$:

$$L(R^*) = \{u_1 \dots u_n \mid u_i \in L(R)\}$$

- Die Sprache $L(R^*)$ enthält alle Wörter, die beliebige Aneinanderreihungen von *Wörtern in $L(R)$* sind.
- In der Aneinanderreihungen kann ein Wort beliebig oft vorkommen, muss es aber nicht.
- Auch das leere Wort ε liegt in $L(R^*)$.
- Die Sprache $L(R^*)$ ist *unendlich*.

Beispiele

```
maus:media tantau$ grep -n -E "as*e" faust.txt
...
6465:Solvet saeclum in favilla.
6602:Und die langen Felsennasen,
6604:Wie sie schnarchen, wie sie blasen!
6606:     Durch die Steine, durch den Rasen
6627:Aus belebten, derben Masern
6628:Strecken sie Polypenfasern
6648:Fasse wacker meinen Zipfel!
...
maus:media tantau$
```

Zur Übung

1. Welche Wörter der Länge ≤ 6 gehören zu $L(\texttt{(ab|cab)*})$?
2. Beschreiben Sie $L(R)$ für folgenden regulären Ausdruck R:
   `aug((a|u|g|c)(a|u|g|c)(a|u|g|c))*(uag|uaa|uga)`
3. Geben Sie drei Wörter an, die vom Ausdruck `0(1*0|2*0)*3` erzeugt werden.
4. Finden Sie einen regulären Ausdruck, der alle Worte über $\{a, b, c\}$ beschreibt, die mit a oder b beginnen und deren drittletzter Buchstabe ein c ist.

Analogie von regulären und arithmetischen Ausdrücken.

Syntaktische Analogien

Arithmetische Ausdrücke

Bestandteile:

1. Zahlen
2. Unäre Operationen wie Wurzel.
3. Binäre Operationen wie
   - Addition
   - Multiplikation
   - Potenz
4. Klammern

Reguläre Ausdrücke

Bestandteile:

1. Ein-Wort-Sprachen
2. Unäre Operationen wie Kleene-Stern.
3. Binäre Operationen
   - Vereinigung
   - Verkettung
4. Klammern

Analogie von regulären und arithmetischen Ausdrücken.

Semantische Analogien

Arithmetische Ausdrücke

1. Ein arithmetischer *Ausdruck* und sein *Wert* sind verschiedene Dinge:
   Der *Ausdruck*
   $(5 + 6) \cdot 2$
   hat den *Wert*
   22.
2. Eine Zeichenfolge wie 22 kann man sowohl als Ausdruck als auch als Wert lesen.

Reguläre Ausdrücke

1. Ein regulärer *Ausdruck* und sein *Wert* sind verschiedene Dinge:
   Der *Ausdruck*
   `(a|b)*a`
   hat den *Wert*
   $\{w \mid w$ besteht aus a's und b's und endet auf a$\}$.
2. Der Ausdruck `hallo` hat als Wert gerade die Ein-Wort-Sprache $\{$hallo$\}$.

Einsatzgebiete von regulären Ausdrücken.

- Innerhalb von Programmen werden reguläre Ausdrücke zur *Eingabekontrolle* benutzt.
- Innerhalb von Programmen werden reguläre Ausdrücke zum *Parsen von Eingaben* benutzt.
  Beispiel: Zerlegung einer URL in ihre Bestandteile.
- Menschen können reguläre Ausdrücke für *Suchanfragen* benutzen.
  - In der Datenbanksprache »SQL« kann man statt `like` auch *`regexp`* benutzen, um nach Attributen zu filtern.
  - In Microsoft Word kann man im Suchen-Dialog einen *Mustervergleich* einschalten.
  - Das Programm `grep` sucht nach einem Vorkommen eines regulären Ausdrucks in einer Datei.

Probleme beim Einsatz von regulären Ausdrücken.

Notationsprobleme:

- Will man reguläre Ausdrücke aufschreiben, so muss man irgendwie alles mit ASCII-Zeichen aufschreiben.
- Leider hat sich keinerlei Standard zum Aufschreiben von regulären Ausdrücken durchgesetzt.

Effizienzprobleme:

- Manche Arten regulärer Ausdrücke lassen sich *effizienter behandeln* als andere.
- Deshalb ist es manchmal sinnvoll, nur *bestimmte Arten von regulären Ausdrücken zuzulassen*.
- Einige Programme vereinfachen deshalb die Notation, sind dann aber inkompatibel mit anderen Notationen.

Typische Notationen für reguläre Ausdrücke.

Auch wenn die Notation für reguläre Ausdrücke nicht einheitlich ist, so gilt wenigstens *oft*:

- Statt `a|b|c|d|e|x|y|z` kann man auch schreiben `[abcdexyz]` oder auch kürzer `[a-exyz]`.
- Statt einer Aufzählung aller möglichen Zeichen kann man einfach einen Punkt schreiben, er steht für ein beliebiges Zeichen.
- Statt $(|A)$, was »gar kein Vorkommen von A oder ein Vorkommen von A« bedeutet, kann man schreiben $A?$.
- Statt $AA*$, was »mindestens ein Vorkommen von A« bedeutet, kann man schreiben $A+$.
- Schreibt man $A\{n\}$, wobei n eine Zahl ist, so bedeutet dies »genau n Vorkommen von A«.
- Schreibt man $A\{n, m\}$, so bedeutet dies »mindestens n und höchstens m Vorkommen von A«.
- Sucht man tatsächlich nach einem Sonderzeichen, so kann man ihm einen Backslash voranstellen.

Zusammenfassung

Worte und Sprachen

- Ein *Wort* ist eine endliche Folge von Symbolen über einem *Alphabet*.
- Eine *Sprache* ist eine beliebige (!) Menge von Worten über einem festen Alphabet.

Regulärer Ausdruck

- Ein *regulärer Ausdruck R* ist selbst ein Wort über einem speziellen Alphabet, das einige Sonderzeichen enthält.
- Er *beschreibt* eine Sprache $L(R)$ nach den in diesem Kapitel beschriebenen Regeln.

Das Muster-Such-Problem

Beim Muster-Such-Problem hat man einen regulären Ausdruck R gegeben und einen Text und man *sucht* nach Vorkommen von Worten aus $L(R)$ in dem Text. Das Problem ist auch bei *Eingabekontrollen* wichtig.