Stringverarbeitung I Mustersuche mittels »Regular Expressions«

Christian Gollwitzer

Experimentalphysik V, Universität Bayreuth, D-95440 Bayreuth

28. April 2006

Wortbedeutung

- „Regulärer Ausdruck" ist üblich, aber Fehlübersetzung
- Treffende Übersetzung: „Aus Regeln aufgebautes Suchmuster"
- Beispiel: *Drei Ziffern und ein großer Buchstabe*

Wortbedeutung

- „Regulärer Ausdruck" ist üblich, aber Fehlübersetzung
- Treffende Übersetzung: „Aus Regeln aufgebautes Suchmuster"
- Beispiel: *Drei Ziffern und ein großer Buchstabe*
- Wirkung auf den Laien: Wirrer Zeichensalat:

```
\b([\w'-]+)(\s+\1)+\b
```

```
/\\ref[ \t]*\*?{[ \t]*([^}]*)}/\1/
```

- Ohne Zusatzinformation praktisch unlesbar

Einordnung

- Zunächst bei UNIX-Texteditoren: `ed` und im Compilerbau
- Parsen (Zerlegen) von maschinellen Sprachen
- Effizient im Analysieren von Strings (s. Informatik: Finite Automaten)
- Später in Skriptsprachen komplett integriert: `sed`, `awk`, `perl`
- Zwei Haupt-Geschmacksrichtungen: *POSIX* und *Shell*

Einführendes Beispiel

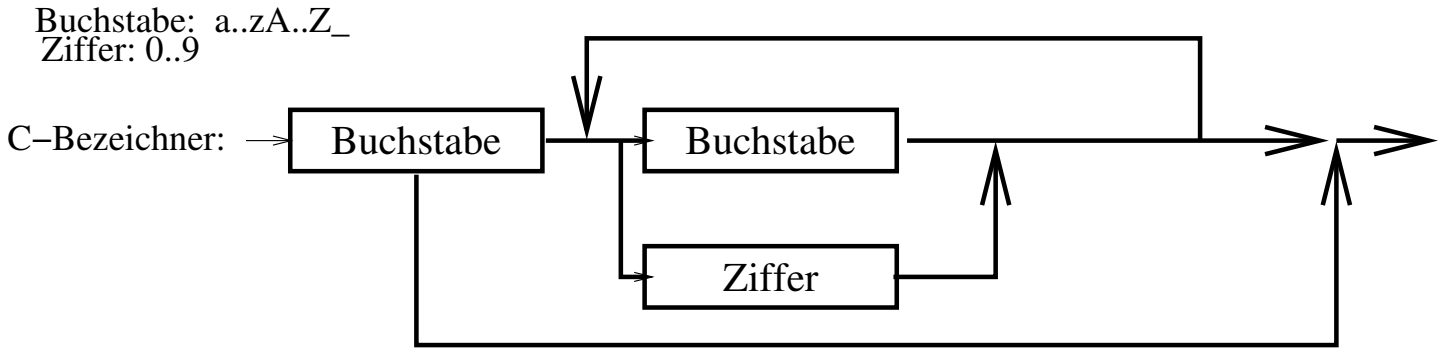
- `egrep 'Ausdruck'Datei` zeigt alle Zeilen einer Datei, die auf einen Regex passen,
  `egrep -o 'Ausdruck'Datei` alle Treffer
- Die meisten Zeichen passen auf sich selbst:

  ```
  egrep 'Hallo' Datei
  ```
- Ein beliebiges Zeichen ist der Punkt:

  ```
  egrep 'H.llo' Datei
  ```

  zeigt auch englische Begrüßungen
- `egrep '[0-9]'`
  Eine Ziffer
- `egrep '[a-g]'`
  Ein a,b,c,d,e,f oder g
- `egrep '[0-9]+'`
  mindestens eine Ziffer

Beispiel: C-Bezeichner (POSIX)



Ein C-Bezeichner...

- ...darf Ziffern und Buchstaben und Unterstriche enthalten:

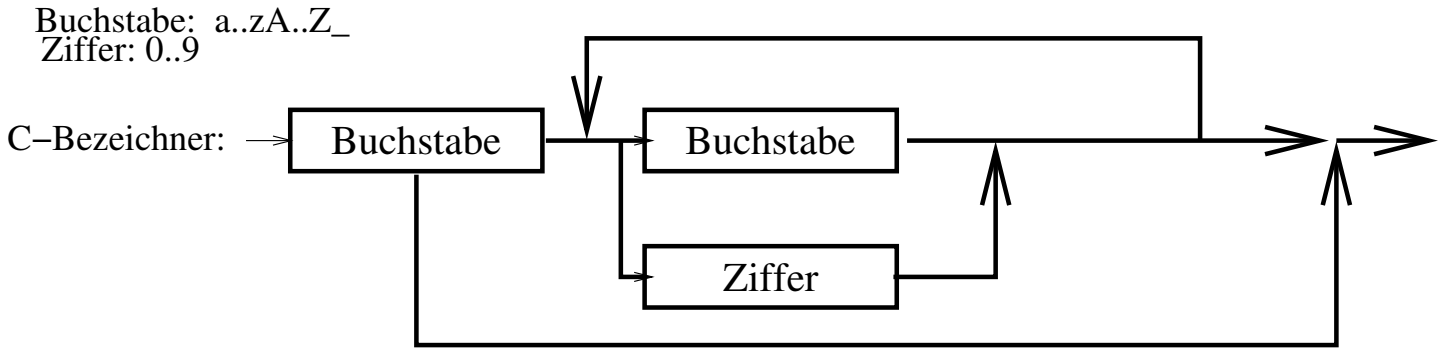
  `[a-zA-Z0-9_]`

- ...muss mit einem Buchstaben oder einem Unterstrich beginnen:

  `[a-zA-Z_]`

- ...muss mindestens ein Zeichen lang sein

Beispiel: C-Bezeichner (POSIX)



Ein C-Bezeichner...

- ...darf Ziffern und Buchstaben und Unterstriche enthalten:

  `[a-zA-Z0-9_]`

- ...muss mit einem Buchstaben oder einem Unterstrich beginnen:

  `[a-zA-Z_]`

- ...muss mindestens ein Zeichen lang sein
- `egrep -o '[a-zA-Z_][a-zA-Z_0-9]*'`
- `egrep -o '[[:alpha:]_][[:alpha:][:digit:]_]*'`

Beispiel: Uhrzeiten (POSIX)

Eine Uhrzeit...

- ...hat die Form hh:mm:ss
- ...Stunden ≥ 30 sind suspekt, genauso Minuten und Sekunden ≥ 60

Beispiel: Uhrzeiten (POSIX)

Eine Uhrzeit...

- ...hat die Form hh:mm:ss
- ...Stunden ≥ 30 sind suspekt, genauso Minuten und Sekunden ≥ 60
- `egrep -o '[012]?[0-9]:[0-5]?[0-9]:[0-5]?[0-9]'`

Beispiel: Uhrzeiten (POSIX)

Eine Uhrzeit...

- ...hat die Form hh:mm:ss
- ...Stunden ≥ 30 sind suspekt, genauso Minuten und Sekunden ≥ 60
- `egrep -o '[012]?[0-9]:[0-5]?[0-9]:[0-5]?[0-9]'`
- Verankerung am Ende der Zeile:

  `egrep '[012]?[0-9]:[0-5]?[0-9]:[0-5]?[0-9]$'`

Beispiel: Reelle Zahl (POSIX)

Eine reelle Zahl...

- ...kann ein Vorzeichen haben +23
- ...dann mind eine Ziffer 7
- ...dann evtl. ein Punkt und Nachkommastellen -37.53
- ...dann einen Exponenten: e oder E, bis zu dreistllige Zahl mit Vorzeichen -278.53e+200

Beispiel: Reelle Zahl (POSIX)

Eine reelle Zahl...

- ...kann ein Vorzeichen haben +23
- ...dann mind eine Ziffer 7
- ...dann evtl. ein Punkt und Nachkommastellen -37.53
- ...dann einen Exponenten: e oder E, bis zu dreistllige Zahl mit Vorzeichen -278.53e+200
- `'[+-]?[0-9]+(\.[0-9]+)?([eE][+-]?[0-9]{1,3})?'`

Beispiel: Dvips-Logfileanalyse

- Latex-File mit eingebundenen Bildern soll weitergegeben werden
- Problem: Kopiere alle Bilder in ein Unterverzeichnis
- Vorhanden: Dvips-Logfile

```
          <texps.pro><special.pro>. <cmbx9.pfb>
[<realspace_3d.eps><farbcastle.eps>] [6]
 [7<voronoi50_bw.eps>
```

Beispiel: Dvips-Logfileanalyse

- Latex-File mit eingebundenen Bildern soll weitergegeben werden
- Problem: Kopiere alle Bilder in ein Unterverzeichnis
- Vorhanden: Dvips-Logfile

```
          <texps.pro><special.pro>. <cmbx9.pfb>
 [<realspace_3d.eps><farbcastle.eps>] [6]
  [7<voronoi50_bw.eps>
```

- `egrep -o '[a-zA-Z_0-9]+\.eps' paper.log`
- eigentliches Kopieren kommt erst später; bash:

  `cp $(...) bilder`

Beispiel: Dreifachbuchstaben – Backreferences

Gegeben eine Wortliste. Suche alle Wörter mit Dreifachkonsonanten

- Konsonant gegeben: Einfach

```
egrep 'sss' wortliste.txt
```

Beispiel: Dreifachbuchstaben – Backreferences

Gegeben eine Wortliste. Suche alle Wörter mit Dreifachkonsonanten

- Konsonant gegeben: Einfach

  ```
  egrep 'sss' wortliste.txt
  ```

- Buchstabe unbekannt: Problem

  ```
  egrep 'sss|ttt|fff|...'
  ```

  funktioniert, aber unendliche Ausdrücke

Beispiel: Dreifachbuchstaben – Backreferences

Gegeben eine Wortliste. Suche alle Wörter mit Dreifachkonsonanten

- Konsonant gegeben: Einfach

  ```
  egrep 'sss' wortliste.txt
  ```

- Buchstabe unbekannt: Problem

  ```
  egrep 'sss|ttt|fff|...'
  ```

  funktioniert, aber unendliche Ausdrücke

- Lösung: Backreference

  ```
  egrep '(.)\1\1'
  ```

Beispiel: Dreifachbuchstaben – Backreferences

Gegeben eine Wortliste. Suche alle Wörter mit Dreifachkonsonanten

- Konsonant gegeben: Einfach

  ```
  egrep 'sss' wortliste.txt
  ```

- Buchstabe unbekannt: Problem

  ```
  egrep 'sss|ttt|fff|...'
  ```

  funktioniert, aber unendliche Ausdrücke

- Lösung: Backreference

  ```
  egrep '(.)\1\1'
  ```

- Missverstanden: Wort mit drei gleichen Buchstaben („Pfeiffer")

  ```
  egrep '(.).*\1.*\1'
  ```

Shell-Pattern

- Weniger leistungsfähig; standardmäßig nur Unterstützung für

  `? ≈ ., * ≈ .*, [chars]`
- Match von Dateinamen muss vollständig sein ≈ Anker vorn und hinten
- Weitere Operatoren in besseren Shells, bash & ksh:

  `?(), +(), *(), @(), !()`

  mit `shopt -s extglob`
- Shell-Pattern werden durch alle Dateinamen ersetzt, die matchen
- Test mit

  `echo pa*ttern`

Beispiel: Shell-Pattern

- Gegeben: Dateien mit Namen `Bild0.bmp...Bild30.bmp`
- Gesucht: Bilder mit Nummern ≤ 10

Beispiel: Shell-Pattern

- Gegeben: Dateien mit Namen `Bild0.bmp...Bild30.bmp`
- Gesucht: Bilder mit Nummern ≤ 10
- `echo Bild?.bmp Bild10.bmp`
- Gesucht: Bilder mit Nummern ≥ 20

Beispiel: Shell-Pattern

- Gegeben: Dateien mit Namen `Bild0.bmp...Bild30.bmp`
- Gesucht: Bilder mit Nummern ≤ 10
- `echo Bild?.bmp Bild10.bmp`
- Gesucht: Bilder mit Nummern ≥ 20
- `echo Bild[23]?.bmp`

Schluss und Ausblick: Grenzen von RegExps

- Regexps sind effektives Werkzeug zur Stringanalyse
- Realisierung als endlicher Automat
- $\Rightarrow$ höhere Sprachen nicht realisierbar, z.B. Klammerung:

  `a( a( b ) c ( d ) )`

  Ist die Klammerung korrekt? $\Rightarrow$ Stackautomat
- „kontext-freie" Grammatiken (z.B. C) nicht realisierbar $\Rightarrow$ Yacc
- Keine Logik-Prüfung möglich $\Rightarrow$ Kombination mit höherer Sprache, z.B. in `awk` $\Rightarrow$ nächste Woche

Literatur

- `http://www.regular-expression.info/`
- Manuals der jeweiligen Programme
- Mastering Regular Expressions, O'Reilly
  `http://www.oreilly.com/catalog/regex/`