

# Einführung in HTML

Computerlinguistik der Universität Erlangen  
zusammengestellt von Martin Lorenz  
und Jörg Kapfer

15. Januar 2005

## Inhaltsverzeichnis

<b>1 Einführung</b>	<b>1</b>
1.1 Was sind reguläre Ausdrücke? . . . . .	1
1.2 Wozu sind reguläre Ausdrücke gut? . . . . .	1
<b>2 Wie sieht ein regulärer Ausdruck aus?</b>	<b>2</b>
2.1 Die Zeichenklasse . . . . .	2
2.2 Die Postfix-Operatoren . . . . .	2
2.3 Die Alternative . . . . .	3
2.4 Die Klammerung . . . . .	3
2.5 Zeilenanfang und -ende . . . . .	3
<b>3 Übergangsdigramme</b>	<b>4</b>
<b>4 Übungsaufgaben</b>	<b>6</b>

## 1 Einführung

### 1.1 Was sind reguläre Ausdrücke?

Ein regulärer Ausdruck (engl. *regular expression*) ist eine “Schablone”, die für eine oder mehrere Zeichenketten steht. Hier ein Beispiel: Der reguläre Ausdruck

`H.llo`

steht für alle Zeichenketten, die mit “H” anfangen und mit “llo” aufhören; dazwischen darf ein beliebiges Zeichen stehen. “H.llo” steht also für Zeichenketten wie “Hallo”, “Hello”, “Hxllo”, aber auch für “H-llo”, “H llo” und andere. Der Punkt “.” hat in einem regulären Ausdruck eine besondere Bedeutung: er steht für ein beliebiges Zeichen. Es gibt noch etliche andere Symbole mit spezieller Bedeutung (mehr dazu später).

## 1.2 Wozu sind reguläre Ausdrücke gut?

Reguläre Ausdrücke sind sehr gut dazu geeignet, Texte nach bestimmten Zeichenketten zu durchsuchen, wenn man gleichzeitig nach mehr als einer Zeichenkette sucht, weil man sich z.B. nicht über die Groß-/Kleinschreibung klar ist. Unix stellt dafür einen Befehl zur Verfügung: **grep**. Er wird folgendermaßen aufgerufen:

```
grep -E "regulärer Ausdruck" Datei1 Datei2 ...
```

Die Text-Dateien *Datei<sub>1</sub>*, *Datei<sub>2</sub>* ... werden nach Zeichenketten durchsucht, die *regulärer Ausdruck* entsprechen, und jede Zeile, die eine solche Zeichenkette enthält, wird auf dem Bildschirm ausgegeben.

Der Editor **Emacs** kennt ebenfalls reguläre Ausdrücke: Man kann mit den Befehlen `M-x search-forward-regexp` und `M-x search-backward-regexp` nach regulären Ausdrücken im aktiven Buffer suchen. Mit dem Befehl `C-M-s` kann man die inkrementelle Suche aufrufen, nur daß man diesmal, statt einer gewöhnlichen Zeichenkette, einen regulären Ausdruck eingibt.

## 2 Wie sieht ein regulärer Ausdruck aus?

Zwei Prinzipien haben wir bereits kennengelernt:

1. Der Punkt "." steht für ein beliebiges Zeichen.
2. Zeichen, die in der Zeichenkette vorkommen sollen, werden an die entsprechende Stelle des regulären Ausdrucks geschrieben.

Für das letzte Prinzip gibt es ein paar Ausnahmen: Die Zeichen "\ () [] \*+?- . ^ \$" müssen durch ein vorangestelltes "\" gekennzeichnet werden, weil sie spezielle Bedeutungen bei der Bildung von regulären Ausdrücken haben. Der Ausdruck für die Zeichenkette "\$o." ist beispielsweise "\\$o\."

### 2.1 Die Zeichenklasse

Manchmal ist das Symbol "." zu allgemein, weil man zwar an einer bestimmten Stelle des regulären Ausdrucks mehrere Zeichen erlauben will, aber nicht *alle* Zeichen. Dann kann man sich mit einer *Zeichenklasse* behelfen: Man schreibt jedes Zeichen, das an dieser Stelle des Ausdrucks stehen darf, in eckige Klammern "[ ]", ohne irgend einen Zwischenraum zwischen den einzelnen Zeichen. Der reguläre Ausdruck "H[ae]llo" etwa steht *nur* für die beiden Zeichenketten "Hallo" und "Hello". Wichtig ist, sich zu merken, daß eine Zeichenklasse immer nur für *ein* einzelnes Zeichen einer Zeichenkette steht und nur sagt, welche Zeichen hier zulässig sind.

Wenn man alle Buchstaben oder alle Ziffern in eine Zeichenklasse aufnehmen will, ist es sehr mühsam, sie alle einzeln aufzuzählen. Deswegen kann man sich die Arbeit abkürzen und schreibt z.B. nur "[a-z]" für alle Kleinbuchstaben von a bis z. Diese Abkürzung, sie heißt *Bereich* sollte man aber nur innerhalb der Großbuchstaben, innerhalb der Kleinbuchstaben oder innerhalb der Ziffern verwenden, weil Bereiche auf den Computer-internen Codierungen der Zeichen basieren, und die dürften die wenigsten auswendig kennen. Die Zeichenklasse "[A-z]" etwa steht nicht nur für jeden Groß- und Kleinbuchstaben, sondern auch für einige Sonderzeichen.

In einer Zeichenklasse dürfen auch mehrere Bereiche hintereinander stehen (wieder ohne Zwischenraum), und sie dürfen mit einzelnen Zeichen gemischt werden. Beispiel: Die Zeichenklasse “[A-Za-zßÄÖÜäöü]” steht für jeden Groß- und Kleinbuchstaben, das “ß” und jeden Umlaut (Die Umlaute sind nicht im Bereich “A-Z” bzw. “a-z” enthalten).

Wenn die Zeichenklasse sehr groß ist, kann es ökonomischer sein, alle Zeichen aufzuzählen, die *nicht* in der Zeichenklasse enthalten sein sollen. Dazu fügt man direkt hinter das “[” ein Circonflex “^” ein. Die Zeichenklasse “[^A-ZÄÖÜ]” steht für jedes Zeichen *außer* den Großbuchstaben und den großen Umlauten.

## 2.2 Die Postfix-Operatoren

Bis jetzt müssen Zeichenketten, die einem regulären Ausdruck entsprechen, immer die gleiche Länge haben. Um flexibler zu werden, kann man mit speziellen Operatoren angeben, wie oft das Zeichen, das davorsteht, in der Zeichenkette vorkommen darf. Diese Operatoren heißen Postfix-Operatoren, weil sie *hinter* dem Zeichen stehen, das sie beeinflussen. Es gibt drei Postfix-Operatoren:

“?” - **die Option.** Der Ausdruck “Pfanne?kuchen” steht für zwei Zeichenketten: “Pfannkuchen” und “Pfannekuchen”. Das Zeichen, das vor dem “?” steht, darf in der Zeichenkette an der entsprechenden Stelle vorkommen (“Pfannekuchen”), aber auch fehlen (“Pfannkuchen”).

“+” - **die Wiederholung.** Der Ausdruck “Toll+11” steht für die Zeichenketten: “Toll”, “Tooll”, “Toooll” usw. Das Zeichen, das vor dem “+” steht, muß in der Zeichenkette an der entsprechenden Stelle mindestens einmal vorkommen (“Toll”), darf aber beliebig oft wiederholt werden — nach oben sind keine Grenzen gesetzt.

“\*” - **der Kleene-Operator.** Der Ausdruck “He!\*” steht für die Zeichenketten: “He”, “He!”, “He!!”, “He!!!” usw. Das Zeichen, das vor dem “\*” steht, muß in der Zeichenkette an der entsprechenden Stelle nicht unbedingt vorkommen (“He”), darf aber auch einmal (“He!”) vorkommen oder beliebig oft wiederholt werden.

Die Postfix-Operatoren dürfen auch hinter Zeichenklassen und dem “.” stehen. Wenn eine Zeichenklasse mehrfach wiederholt wird, dann darf bei jeder Wiederholung ein neues Zeichen aus der Zeichenklasse in der Zeichenkette stehen, unabhängig davon, welches Zeichen bei der letzten Wiederholung stand. Der reguläre Ausdruck “[au]+!” steht also nicht nur für Zeichenketten wie “aaa!” und “uuuuu!”, sondern auch für solche wie “auuuu!” oder “auauauaa!”. Das gleiche gilt für das Symbol “.”: Der Ausdruck “.\*” etwa steht für beliebig viele beliebige Zeichen, also für *jede* Zeichenkette.

## 2.3 Die Alternative

Wie kann man einen regulären Ausdruck schreiben, der für zwei verschiedene Zeichenketten steht, etwa “Fahrstuhl” und “Aufzug”? Man schreibt einfach die Ausdrücke für die beiden Zeichenketten hintereinander, durch einen senkrechten Strich “|” getrennt: “Fahrstuhl|Aufzug”. Der Ausdruck ist eine *Alternative* zwischen “Fahrstuhl” und “Aufzug”. Man kann auch Alternativen zwischen drei oder noch mehr einzelnen regulären Ausdrücken bilden, wie “Fahrstuhl|Aufzug|Lift”. Die einzelnen Ausdrücke dürfen Zeichenklassen, den Punkt und Postfix-Operatoren enthalten.

## 2.4 Die Klammerung

Der Wirkungsbereich mancher Operatoren ist für manche Fälle zu klein oder zu groß. Wenn man etwa einen regulären Ausdruck für “wirksam” und “wirklich” schreiben will, hat “wirksam|lich” nicht die gewünschte Wirkung. Man muß den Teil der Zeichenkette klammern, für den die Alternative gültig sein soll: “wirk(sam|lich)”. Ein weiteres Beispiel: “la+” ist nicht der richtige Ausdruck für Zeichenketten wie “la”, “lala”, “lalala” usw., weil er auch für Zeichenketten wie “laa” und “laaaa” steht. Der richtige reguläre Ausdruck ist in diesem Fall  $(la)^+$ .

Reguläre Ausdrücke, die in **Emacs** eingegeben werden, müssen mit “\ ( \)” statt mit “( )” geklammert werden.

## 2.5 Zeilenanfang und -ende

Die auf regulären Ausdrücken basierenden Suchbefehle in **Emacs** und **grep** suchen in jeder Zeile eines Buffers (oder einer Datei) nach Teil-Zeichenketten, die dem regulären Ausdruck entsprechen. Das Symbol “^” am Anfang des Ausdrucks bedeutet, daß die gesuchte Zeichenkette am *Anfang* einer Zeile stehen muß; das Symbol “\$” am Ende des Ausdrucks bedeutet, daß die Zeichenkette am *Ende* einer Zeile stehen muß. Der reguläre Ausdruck “^Anfang” etwa steht für alle Zeichenketten “Anfang”, die am Anfang einer Zeile stehen.

## 3 Übergangsdigramme

Eine anschauliche Darstellung für reguläre Ausdrücke bieten sogenannte Übergangsdigramme. Diese bestehen aus beschrifteten Knoten, die durch Pfeile (*gerichtete Kanten*) miteinander verbunden sind. Der Graph wird von links aus durchlaufen, wobei man — die vorgegebenen Pfeile benutzend — von einem Knoten zum nächsten springt. Geendet wird dann, wenn man ganz rechts angekommen ist.

Im Zusammenhang mit den regulären Ausdrücken bedeutet das: Man läuft (von links aus) über die Pfeile von Knoten zu Knoten und merkt sich die Beschriftungen der durchlaufenen Knoten. Ist man am rechten Ende angekommen, sollten die aneinandergereihten Knotenbeschriftungen eine Zeichenkette bilden, die auf den zugrundeliegenden regulären Ausdruck paßt. Ist das nicht der Fall, hat man das Diagramm falsch konstruiert. Außerdem muß es zu jeder durch einen regulären Ausdruck beschriebenen Zeichenkette einen Weg durch das Diagramm geben.

Abb. 1 zeigt Diagramme zu einigen einfachen regulären Ausdrücken. Für alle Beispiele wurde die Normalform gewählt, d.h. in jedem Knoten steht genau ein Buchstabe. Diese Regel ist eigentlich nicht in allen Fällen notwendig (z.B. könnte man im Diagramm zu *der* auch nur einen Knoten verwenden, in dem alle drei Buchstaben enthalten sind), sollte aber zur besseren Übersichtlichkeit eingehalten werden.

Für die unterschiedlichen Varianten von regulären Ausdrücken gilt dabei folgendes:

**normale Zeichenkette:** Für jeden Buchstaben wählt man einen Knoten und fügt einen Pfeil von jedem Knoten zu dem Knoten des nachfolgenden Buchstabens ein (vgl. *der*).

**Alternative (|):** Die einzelnen Alternativen werden im Diagramm als parallele Knotenfolgen beschrieben (vgl. *de(r|nn|m)*).

**Wiederholung (+):** Die zu wiederholenden Zeichen werden wie eine normale Zeichenkette in

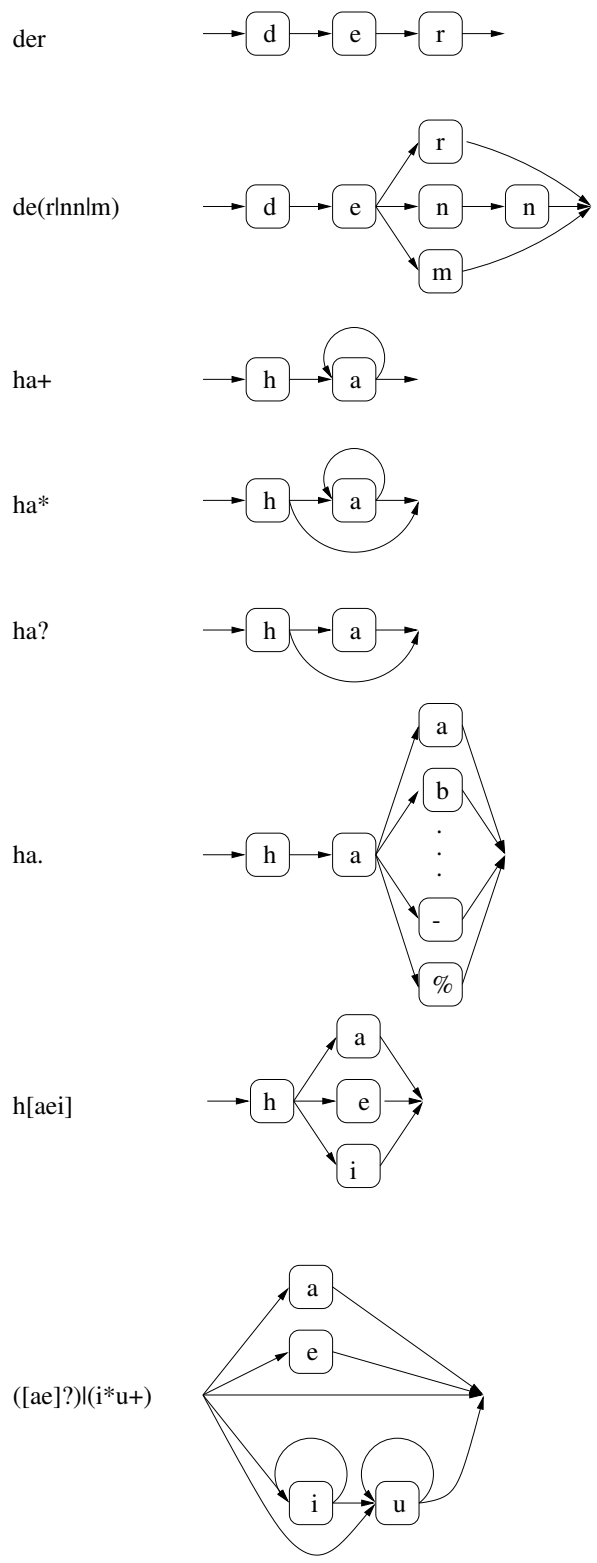


Abbildung 1: Einfache Beispiele für Übergangsdigramme

einen Graphen übertragen. Zusätzlich wird noch ein Pfeil vom Knoten des letzten Zeichens zum Knoten des ersten Zeichens ergänzt (vgl.  $ha+$ ).

**Kleene-Operator (\*):** Man verfährt wie bei der Wiederholung, fügt aber zusätzlich noch einen Pfeil ein, der genau die Knoten der zu wiederholenden Zeichenkette überspringt (vgl.  $ha^*$ ).

**Option (?):** Man verfährt wie bei einer normalen Zeichenkette und fügt danach noch einen Pfeil ein, der die optionalen Zeichen überspringt (vgl.  $ha?$ ).

**beliebiges Zeichen (.):** Dies ist etwas komplizierter, da man alle möglichen Zeichen, für die der Punkt steht, im Diagramm parallel aufzeichnen müßte. Als Vereinfachung kann man sich aber auch mit ein paar Möglichkeiten begnügen und den Rest nur andeuten (vgl.  $ha.$ ).

**komplexere Ausdrücke:** Kommen mehrere der bisher genannten Möglichkeiten gleichzeitig in einem regulären Ausdruck vor, muß man sich das Gesamtdiagramm nach den obigen Mechanismen aus den einzelnen Bestandteilen zusammenbauen (vgl.  $([ae]?)(i^*u+)$ ).

## 4 Übungsaufgaben

1. Für welche Zeichenketten stehen folgende regulären Ausdrücke?

- |                           |                      |
|---------------------------|----------------------|
| (a) “wahnsinn+”           | (g) “wahn(sinn)+”    |
| (b) “(wahn)+(sinn)+”      | (h) “[Rr]ot”         |
| (c) “von [A-Z]”           | (i) “Durch Unfall”   |
| (d) “rosa?m(ari u)n(de)?” | (j) “(Durch Unfall)” |
| (e) “.*(e em en es)”      | (k) “.*a.*”          |
| (f) “An+gst”              | (l) “[A-Z]bung”      |

2. Finden sie für folgende Zeichenketten jeweils einen regulären Ausdruck:

- alle Zeichenketten, die mit einem Vokal anfangen und nicht mit einem Vokal aufhören;
- alle Zeichenketten, die aus einer geraden Anzahl von “a” bestehen;
- alle Zeichenketten, die (neben beliebigen anderen Zeichen) mindestens zweimal “-hicks-” enthalten;
- alle Zeichenketten, die aus einer Folge von “dumm” (mindestens eins) oder aus einer Folge von “di” (mindestens eins) bestehen.
- alle nichtleeren Zeichenketten, die aus beliebigen Folgen von “dumm” und “di” bestehen (wie “dumm”, “di”, “dummdi”, “didummdidi”, “dummdidumm”);
- alle Zeichenketten, die (neben beliebigen anderen Zeichen) entweder
  - genau zwei “m”
  - genau zwei “f”,
  - ein “m” und ein “f”
  - oder weder “m” noch “f”enthalten;
- alle Zeichenketten, die (neben beliebigen anderen Zeichen) das Zeichen “p” mindestens einmal und höchstens dreimal enthalten;
- alle Zeichenketten, die “dumm” nicht enthalten (sehr schwer!).