

Chomsky-Endliche Automaten-Reguläre Ausdrücke

Chomsky:

- Sprache nach dem Modell von Noam Chomsky 1956
- Modell der generativen Grammatik: ein endlicher Automat (Markoff-Prozess) generiert die Sprache nach definierten Regeln

Endliche Automaten:

- Beispiel eines deterministischen endlichen Automaten (DFA): der automatische Türkontrolleur
- Der DFA als Denkmodell
- Der nicht-deterministischer endlicher Automat (NFA)
- Reguläre Sprachen/Reguläre Operationen

Reguläre Ausdrücke:

- Allgemeiner Aufbau von regulären Ausdrücken am Beispiel von *egrep*
- Einfache Beispiele regulärer Ausdrücke
- Unterschiede in der Abarbeitung regulärer Ausdrücke: DFA vs NFA

Anwendungen Regulärer Ausdrücke in der Bioinformatik: Mustersuche in Text

- Mustersuche in Proteinsequenzen: Beispiel Prosite-Datenbank
- Mustersuche in DNA-Sequenzen: Beispielaufgabe: Wieviele möglicherweise polymorphe (CA)_n-Mikrosatelliten befinden sich in den ersten 10 000 bp des kurzen Arms des X-Chromosoms

Noam Chomsky *1928

“From now on I will consider a *language* to be a set (finite or infinite) of sentences, each finite in length and constructed out of a finite set of elements.

All natural languages in their spoken or written form are languages in this sense, since each natural language has a finite number of phonemes (or letters in its alphabet) and each sentence is representable as a finite sequence of these phonemes (or letters), though there are infinitely many sentences.

Similarly, the set of ‘sentences’ of some formalized system of mathematics can be considered a language. (...)

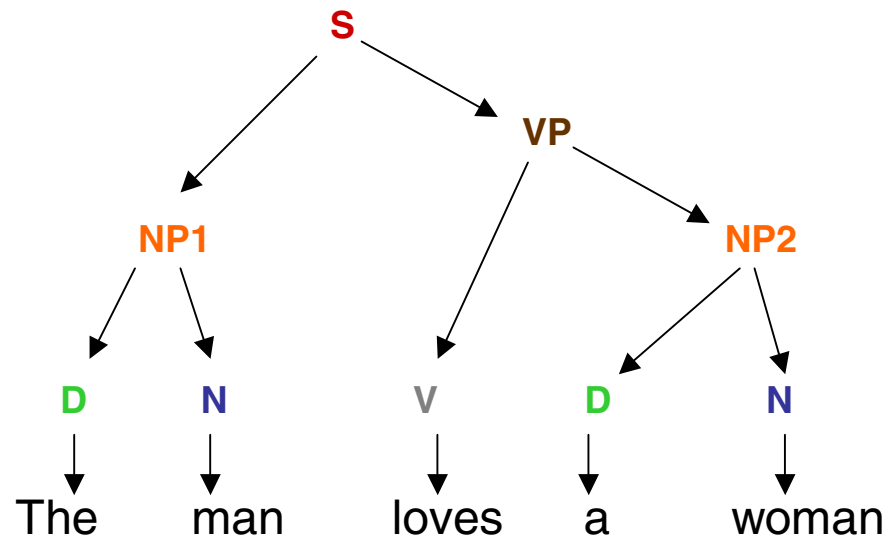
The grammar of L will thus be a device that generates all of the grammatical sequences of L .“

Beschreibende Grammatik

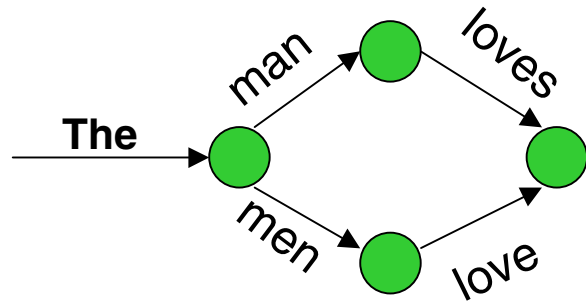
Phrasenstrukturebenen

“The man loves a woman.”

$(((\text{the})_D (\text{man})_N)_{NP1} ((\text{loves})_V ((\text{a})_D (\text{woman})_N)_{NP2})_{VP})_S$

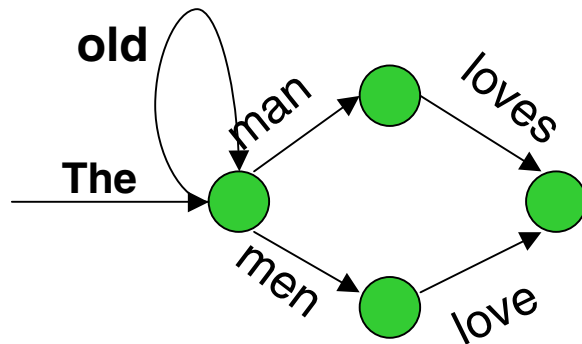


Generative Grammatik



„The man loves“

„The men love“



„The man loves“

„The men love“

„The old man loves“

„The old men love“

„The old old man loves“

„The old old men love“

„...“ ...

„...“ ...

Beschreibende Grammatik bei natürlichen Sprachen:

Chomsky: „Direkte Präsentation der Menge aller grammatikalisch korrekten Sequenzen einer Sprache führt zu einer komplizierten, unübersichtlichen und damit nutzlosen Grammatik...“

Ausweg:

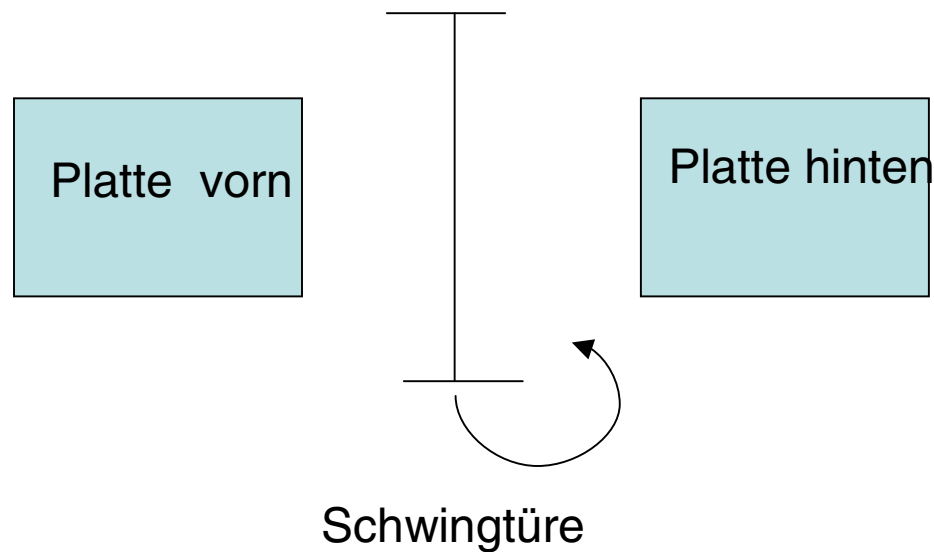
Vorstellung: Der Sprecher einer Sprache ist eine Maschine.

Die Maschine hat: einen **Anfangszustand**, einen **Endzustand** und kann eine **endliche Anzahl interner Zustände** annehmen. Bei jedem Zustandsübergang wird ein Wort produziert. Die Sequenz von produzierten Worten ist ein Satz.

Jede Sprache, die von einer solchen „Sprachmaschine“ produziert werden kann heißt „endliche Zustandssprache“ (**finite state language**). Die Maschine selbst heißt „endliche Zustandsgrammatik“ (**finite state grammar**).

Endlicher Automat

- Modell eines Computers mit extrem limitiertem Speicher
- Beispiel zur Veranschaulichung: automatische Türkontrolle

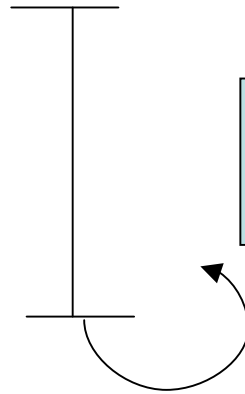


input

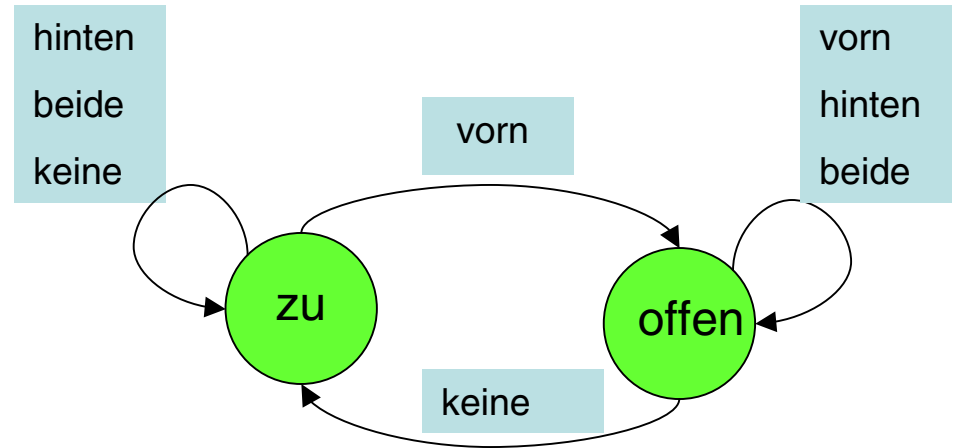
state

	keine Platte	Platte vorn	Platte hinten	Beide Platten
zu	zu	offen	zu	zu
offen	zu	offen	offen	offen

State transition table



Schwingtüre

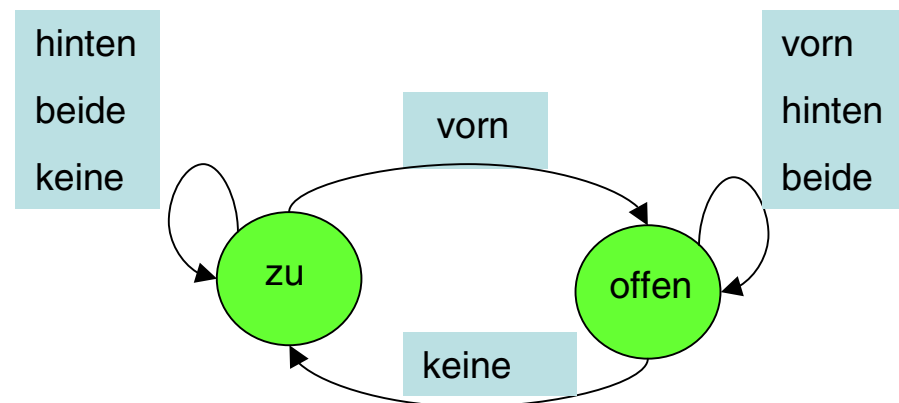


State diagram

Versuch einer formale Beschreibung des endlichen Automaten „Türkontrolleur“

Übergangsfunktion δ

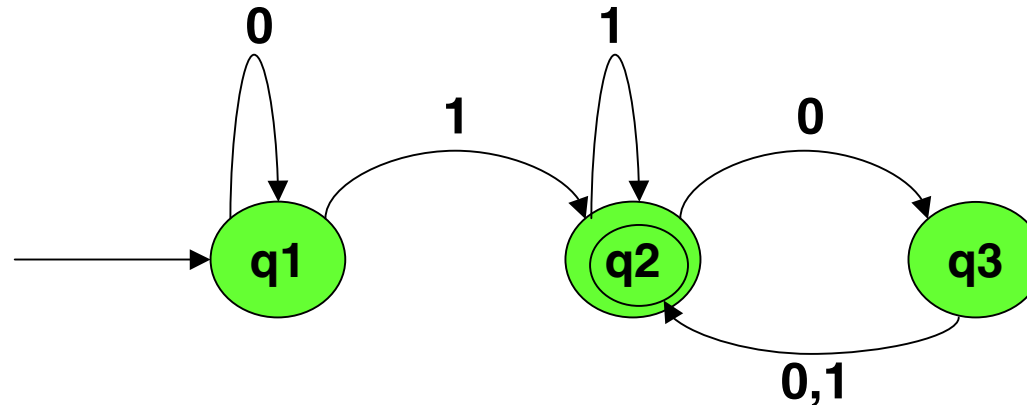
	keine Platte	Platte vorn	Platte hinten	Beide Platten
zu	zu	offen	zu	zu
offen	zu	offen	offen	offen



Zustandsdiagramm

1. endliche Menge von **Zuständen**: $Q = \{zu, auf\}$
2. endliche Menge von Eingabemöglichkeiten (Alphabet): $\Sigma = \{keine PI, beide PI, PI vorne, PI hinten\}$
3. **Übergangsfunktion** $\delta : Q \times \Sigma \rightarrow Q$ (Regeln für den Übergang in den nächsten Zustand)

Endlicher Automat (DFA) als Denkmodell (deterministic) finite automaton)



Formale Beschreibung des Automaten M: $M=(Q, \Sigma, \delta, q_1, F)$

1. **Q** ist eine endliche Menge genannt **Zustände**: $Q=\{q_1, q_2, q_3\}$
2. **Σ** ist eine endliche Menge genannt **Alphabet**: $\Sigma=\{1, 0\}$
3. **Übergangsfunktion**: $\delta : Q \times \Sigma \rightarrow Q$ (Regeln für den Übergang)
4. **Anfangszustand**: $q_1 \in Q$
5. **Menge der Endzustände**: $F \subseteq Q : F=\{q_2\}$

Tabelle der
Übergangsfunktion δ

	0	1
q1	q1	q2
q2	q3	q2
q3	q2	q2

Formale Definition eines DFA

Ein DFA ist ein **5-Tupel**: $(Q, \Sigma, \delta, q_0, F)$ für:

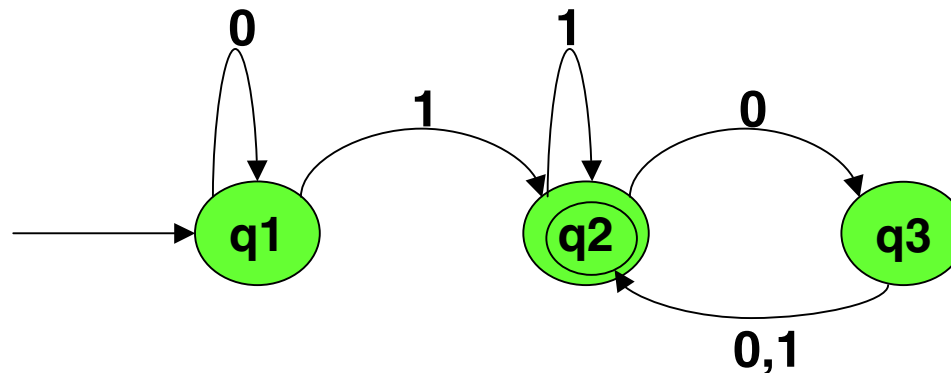
1. Q ist eine endliche Menge genannt **Zustände**.
2. Σ ist eine endliche Menge genannt das **Alphabet**.
3. $\delta : Q \times \Sigma \longrightarrow Q$ ist die **Übergangsfunktion** (*transition function*)
4. $q_0 \in Q$ ist der **Anfangszustand**.
5. $F \subseteq Q$ ist die Menge der **Endzustände** (*accept states*),

„Deterministisch“

Deterministisch: es gibt für jeden Zustand des DFA nur einen möglichen Folgezustand (wegführender Übergangspfeil) für jedes Symbol des Alphabets

Input-Strings: 1, 11, 100, 0100, ... werden angenommen:

Input-Strings: 0, 00, 1000, 10, 1010,werden abgelehnt



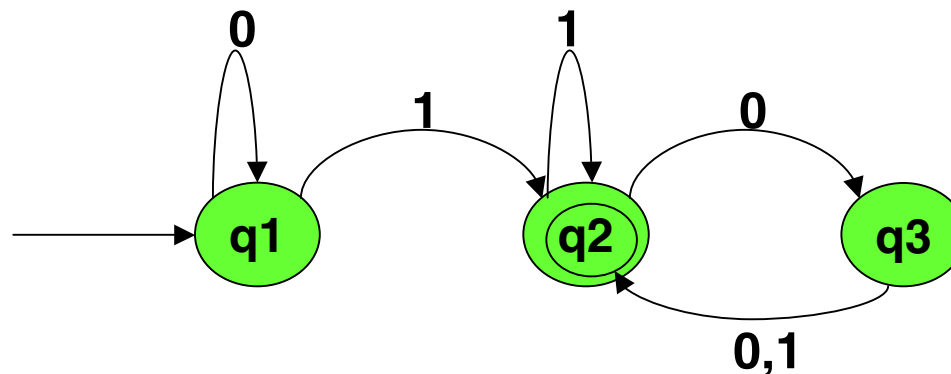
M akzeptiert/erkennt alle Strings des Alphabets $\Sigma=\{0,1\}$, die mit Eins enden oder mit einer geraden Anzahl von Nullen nach der zuletzt eingelesenen Eins.

Die Sprache des endlichen Automaten M

Input-Strings: 1, 11, 100, 0100, ... werden angenommen.

Input-Strings: 0, 00, 1000, 10, 1010,werden abgelehnt.

- M akzeptiert/erkennt alle Strings des Alphabets $\Sigma=\{0,1\}$, die mit Eins enden oder mit einer geraden Anzahl von Nullen nach der zuletzt eingelesenen Eins.



A ist die Menge aller Strings, die von der Maschine M akzeptiert/erkannt werden.

$A=\{\omega \mid \omega \text{ enthält mindestens eine 1 und eine gerade Anzahl von darauf folgenden 0}\}$

man sagt:

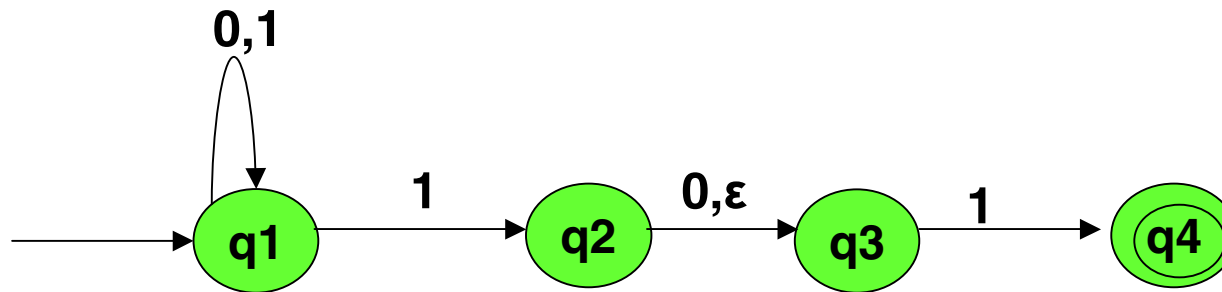
M akzeptiert/erkennt A (und die leere Sprache \emptyset).

A ist die Sprache der Maschine M:

$L(M)=A$.

↓

Nicht deterministischer endlicher Automat



Nicht-deterministisch heisst:

- ein Zustand kann keinen, einen oder viele Übergangspfeile (Transitionen) für jeden Buchstaben des Alphabetes annehmen
- das Alphabet wird um das leer Wort erweitert: $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$
- Die Übergangsfunktion ändert sich entsprechend: $\delta : Q \times \Sigma_\epsilon \rightarrow P(Q)$

Formale Definition eines NFA

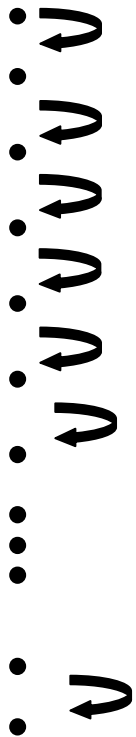
Ein NFA ist ein **5-Tupel**: $(Q, \Sigma, \delta, q_0, F)$ für:

1. Q ist eine endliche Menge genannt **Zustände**.
2. Σ ist eine endliche Menge genannt das **Alphabet**.
3. $\delta : Q \times \Sigma \longrightarrow P(Q)$ ist die **Übergangsfunktion**.
4. $q_0 \in Q$ ist der **Anfangszustand**.
5. $F \subseteq Q$ ist die Menge der **Endzustände**.

Der Eingabestring wird bei einem NFA parallel abgearbeitet

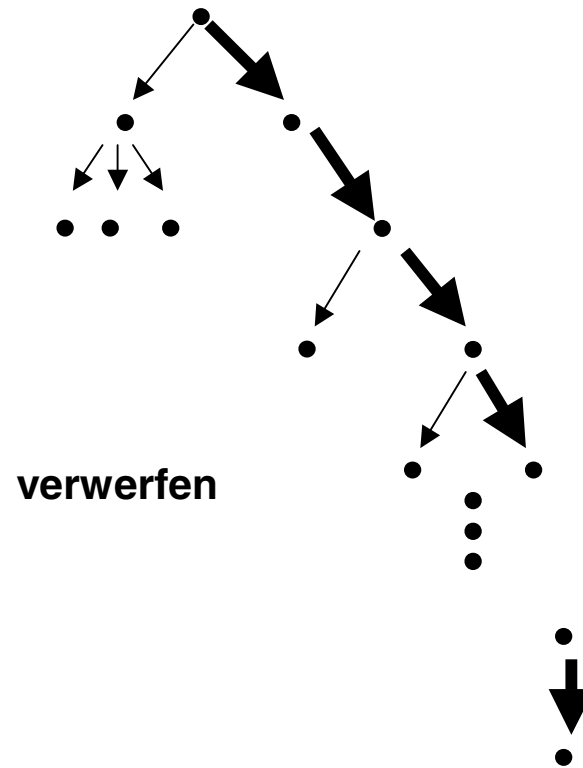
jeder NFA kann prinzipiell in einen äquivalenten DFA umgewandelt werden, wenn alle möglichen Folgezustände als Zustände mit einem Übergang für jeden Buchstaben des Alphabetes dargestellt wird.

Deterministisch



**Endzustand:
akzeptieren/verwerfen**

Nicht-deterministisch



**Endzustand:
akzeptieren**

Reguläre Sprachen/Reguläre Ausdrücke/Reguläre Operationen

Definition: Eine reguläre Sprache (regulärer Ausdruck) ist eine Sprache, die von einem endlichen Automaten erkannt wird.

Reguläre Operationen

**In der Mathematik: Operatoren +; x;
Ausdruck, z.B. (5+3)x4
Wert des Ausdrucks: = 32**

Reguläre Ausdrücke, die Sprachen beschreiben:

Operatoren:

Vereinigung: $A \cup B = \{x \mid x \in A \text{ oder } x \in B\}$

Konkatenation: $A \circ B = \{xy \mid x \in A \text{ und } y \in B\}$

Stern: $A^* = \{x_1, x_2, x_3, \dots, x_k \mid k \geq 0 \text{ und jedes } x_i \in A\}$

Beispiel für reguläre Operationen

- Alphabet Σ sei das gesamte lateinische Alphabet (26 Buchstaben) $\{a, b, \dots, z\}$.
- Sprache $A = \{\text{good}, \text{bad}\}$
- Sprache $B = \{\text{boy}, \text{girl}\}$

$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$,

$A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$

$A^* = \{\epsilon, \text{good}, \text{bad}, \text{goodgood}, \text{goodbad}, \text{badbad}, \text{goodgoodgood}, \text{goodgoodbad}; \dots\}$

Anwendungsgebiet Regulärer Ausdrücke

- Mustersuche in Text:
- Frage: Wird im zu untersuchenden Text (Doktorarbeit, DNA-Sequenz, Proteinsequenz) ein Muster gefunden, das auf das im regulären Ausdruck definierten Muster passt oder passt das Muster nicht
- Ein Regulärer Ausdruck besteht aus Metazeichen und Literalen

Metazeichen bei egrep

Dinge, die auf einzelne Zeichen passen

Metazeichen	Beschreibung	Passt auf
.	Punkt	Irgendein Zeichen
[...]	Zeichenklasse	Eines der Zeichen aus dieser Liste
[^...]	Negierte Zeichenklasse	Irgendein Zeichen nicht aus dieser Klasse
\Zeichen	Escape	Wenn ein Zeichen ein Metazeichen ist oder wenn die Kombination mit Backslash ein neues Metazeichen ergibt: das Literal <i>Zeichen</i>

Dinge, die „zählen“; Quantoren

?	Fragezeichen	Einmal erlaubt, aber optional
*	Stern	Jede Anzahl erlaubt, auch null
+	Plus	Mindestens eins, mehr erlaubt
{min,max}	Expliziter Bereich	<i>min</i> gefordert, bis und mit <i>max</i> erlaubt

Metazeichen bei egrep

Dinge, die auf Positionen passen

<code>^</code>	Zirkumflex	Passt auf Zeilenanfang
<code>\$</code>	Dollar	Passt auf Zeilenende
<code>\<</code>	Wortgrenze	Passt auf die Position am Wortanfang
<code>\></code>	Wortgrenze	Passt auf die Position am Wortende

Anderes

<code> </code>	Alternation	Passt auf mindestens eine der Alternativen
<code>(...)</code>	Klammern	Beschränkt den Geltungsbereich von Alternationen, gruppiert Dinge für nachfolgenden Quantoren, „merkt sich Text“ für Rückwärtsreferenzen
<code>\1,\2</code>	Rückwärtsreferenz	Passt auf Text, der im ersten, zweiten usw. Paar von Klammern vorkam

Einfache Beispiele von regulären Ausdrücken

Was wird durch folgende regulären Ausdrücke erkannt?

Marl?en

q[[^]u]

^

^ding\$

Einfache Beispiele von regulären Ausdrücken

Was wird durch folgende regulären Ausdrücke erkannt?

Marl?en

z.B. Maren, Marlen

q[[^]u]

z.B. Iraqi, miqra, qintar, zaqqum

^

„passt immer am Zeilenanfang“ → alle Zeilen

^ding\$

passt wenn die Zeile einen Anfang hat, gefolgt von d-i-n-g
und wenn dann unmittelbar das Zeilenende folgt
passt also, wenn eine Zeile nur ding enthalten, nichts
anderes;

Auflösung der Aufgabe

