


Blatt 01: Reguläre Sprache

A 1.1: Sprachen von regulären Ausdrücken

Der Ausdruck besteht aus zwei Teilen, welche durch ein Pluszeichen (+) verbunden sind.

Das (+) steht hierbei für ein oder.

Der erste Teil besteht dabei aus dem einzelnen a, welches dabei für das einzelne Zeichen a steht.

Der zweite Teil ($a(a+b)^*a$), auch wieder beginnend mit a, mit darauf folgendem $(a+b)^*$ bedeutet, dass in dem Wort beliebig viele a oder b vorkommen können. Am Ende des Ausdrucks kommt wieder ein a, was bedeutet, dass das Wort wieder mit a enden muss.

Die Sprache besteht nur aus Wörtern, die mit a beginnen und enden und dazwischen nur a und b enthalten.

A 1.2: Bezeichner in Programmiersprachen

1) Regulärer Ausdruck

Ziel:

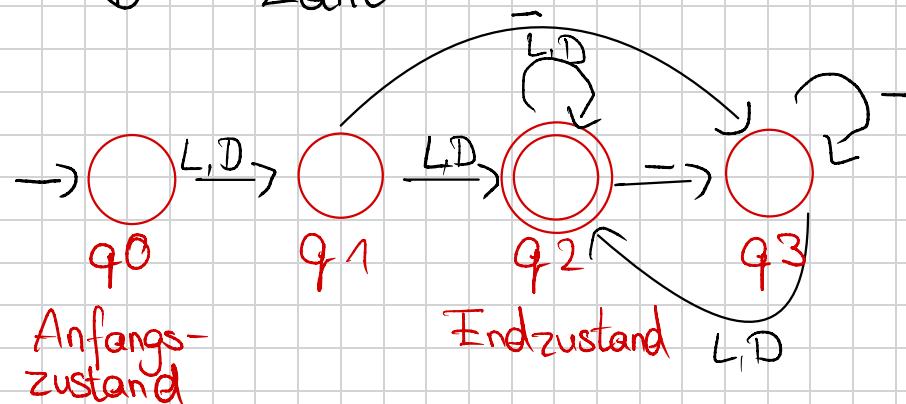
- Start mit Buchstabe (a-z, A-Z)
- Danach Buchstaben / Ziffern / Unterstrich
- Darf nicht mit Unterstrich enden
- Mindestens 2 Zeichen
- Sonderfälle (V/r = Variablen; p/P = Parameter)

Regex:

$^ [A-Za-z][A-Za-z0-9_]*[A-Za-z0-9] \$$

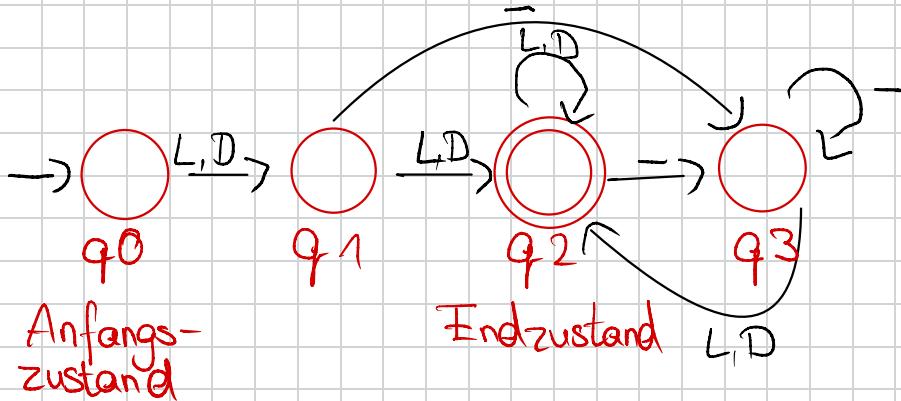
Alphabet

- L = Buchstabe - - = Unterstrich
- D = Zahl



Beispiel 1 Vcount 1

gelesenes Zeichen	Zustand vorher	Neuer Zustand	
-	q_0 (Start)		
V	q_0	q_1	1. Buchstabe
C	q_1	q_2	2. Zeichen ✓
O	q_2	q_2	✓
U	q_2	q_2	✓
n	q_2	q_2	✓
t	q_2	q_2	✓
1	q_2	q_2	Zahl
Ende	q_2	akzeptiert	q_2 gültig



Beispiel 2 p-value 2

gelesenes Zeichen	Zustand vorher	Neuer Zustand	
-	q_0 (Start)		Startzustand
p	q_0	q_1	1. Buchstabe
-	q_1	q_3	Unterstrich
v	q_3	q_2	endet nicht mit -
a	q_2	q_2	✓
l	q_2	q_2	✓
u	q_2	q_2	~
e	q_2	q_2	✓
2	q_2	q_2	akzeptiert

Ableitung aus DFA

DFA Zustand	Bedeutung	Neues Symbol
q0	Startzustand	S
q1	erstes Zeichen gelesen	A
q2	gültig	B
q3	endet auf -	C

Terminale

L = Buchstabe [A-Z, a-z]

D = Ziffer [0-9]

_ = Unterstrich

Regeln

1. Startzustand ($S \rightarrow q0$)

$S \rightarrow L A$

2. Zustand q1 (A)

Nach dem ersten Buchstaben kann Buchstabe, Zahl, -

$A \rightarrow LB \mid DB \mid - C$

3. Zustand q_2 (B) gültig

Darf sich selbst wiederholen, aber nur wenn - nicht am Ende

$$B \rightarrow LB \mid DB \mid -C \mid \epsilon$$

$\rightarrow \epsilon$ beschreibt akzeptierend (gültig)

4. Zustand q_3 (C)

Nach einem - darf wieder ein Buchstabe/Zahl kommen
oder nochmal C (-)

$$C \rightarrow LB \mid DB \mid -C$$

Beispiel Vcount 1

S

$\rightarrow L A \quad (L = V)$

$\rightarrow L B \quad (L = C)$

$\rightarrow L B \quad (L = \emptyset)$

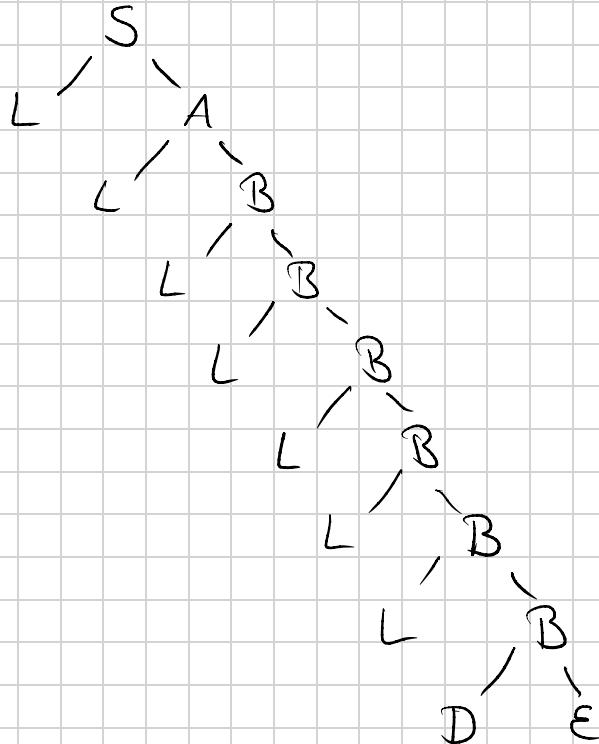
$\rightarrow LB \quad (L = u)$

$\rightarrow LB \quad (L = n)$

$\rightarrow LB \quad (L = t)$

$\rightarrow DB \quad (D = 1)$

$\rightarrow \epsilon$



Beispiel 2: p-value 2

S

→ L A

→ - C

→ L B

→ L B

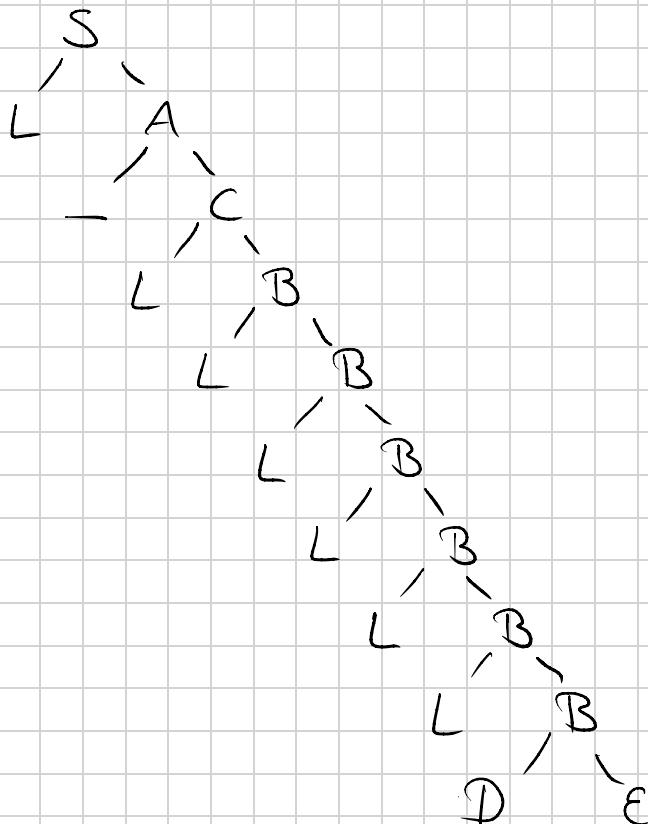
→ L B

→ L B

→ L B

→ D B

- ε



A.1.3: Gleitkommazahlen in Programmiersprachen

Gleitkommazahlen in Python

- Ziffern (0-9)
- Dezimalpunkt (.) → 1.5
- Nach dem Dezimalpunkt muss nichts stehen (1.)
- Unterstrich _ als Trenner
- Exponent

Gleitkommazahlen in Java

- Ziffern (0-9)
- Dezimalpunkt
- Exponent
- Suffix → f oder F = Float, d oder D = Double
- Hexadezimale Gleitkommazahlen

Reguläre Ausdrücke

Python

$^1[0-9]*\cdot ?[0-9]+([eE][+-]?[0-9]+)?\$$

- $[0-9]^*$ → beliebig viele Ziffern

- \cdot → optionaler Punkt

- $[0-9]^+$ → mindestens eine Ziffer danach

- $([eE][+-]?[0-9]+)?$ → optionaler Exponent/Kürzeichen

Java

$^1[0-9]^*\cdot ?[0-9]+([eE][+-]?[0-9]+)?ffdd?\$$

- $[ffdd]$ → optionales Suffix für float/double

A1.4: Mailadressen?

Der folgende Regex ist ungeeignet, weil

- (a-z) ist laut Aufgabe nicht erlaubt (soll ausgeschlossen werden)
- (a-z) deckt nur Kleinbuchstaben ab
- es werden keine Ziffern abgedeckt
- der . heißt normalerweise beliebiges Zeichen,
!. müsste für einen Punkt verwendet werden
- der Ausdruck erlaubt nur eine Domain

Verbesserung:

$^[[A-Za-zA-Z\.-]+@[A-Za-z0-9\.-]+(\?.:\!.[A-Za-z0-9\.-]+)+\$$

T

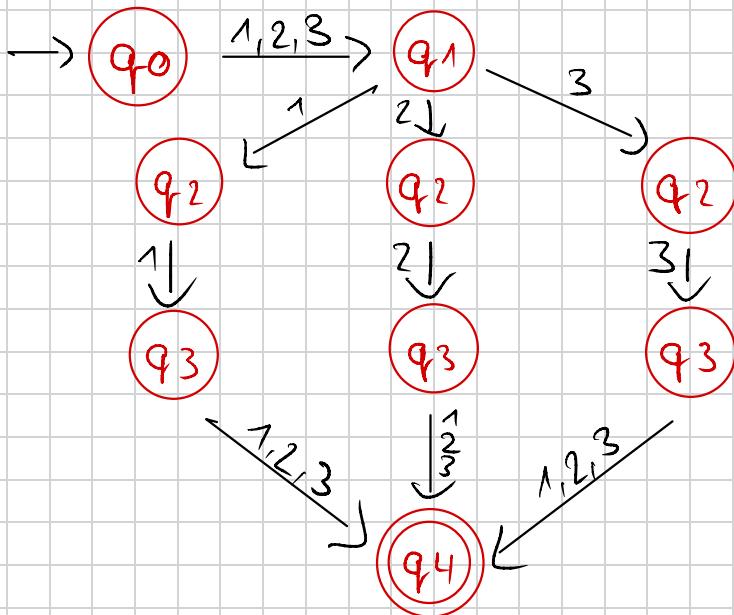
RFC-Lite

Lokaler Teil mindesten 1 Zeichen

Domain Teil ^ Anfang des Strings

Label \$ Ende des Strings

A1.5 : Der zweitletzte Buchstabe



A 1.6: Sprache einer regulären Grammatik

- Das Wort beginnt mit a ($S \rightarrow aA$)
- In A können beliebig viele b oder c ($bA \mid cA$)
- Irgendwann kommt ein d ($A \rightarrow dB$)
- In B gibt es drei Möglichkeiten
 - (aC) Wort endet mit a
 - (bC) Wort endet mit b
 - (cA) c wird angehangen und wir gehen zurück zu A

$$\Sigma = \{a, b, c, d\}$$

Regulärer Ausdruck

$$a((b \mid c)^*dc)^*(b \mid c)^*d(a \mid b)$$

Zustände: S, A, B, C, \emptyset

Zustand	a	b	c	d
S	A	\emptyset	\emptyset	\emptyset
A	\emptyset	A	A	B
B	C	C	A	\emptyset
C	\emptyset	\emptyset	\emptyset	\emptyset
\emptyset	\emptyset	\emptyset	\emptyset	\emptyset

