Grundfragen der Informatik, KV

Turingmaschine

Überblick

Algorithmus

Turingmaschine

Turingvollständigkeit

Halteproblem/Entscheidungsproblem

Turing Machine

Definition: Algorithmus

Algorithmen kennen gelernt: Subtraktion von Binärzahlen, UTF-8, Wahrheitstabelle, Parsetree, KNF erzeugen …

“Ein Algorithmus ist eine detaillierte und explizite Vorschrift zur schrittweisen Lösung eines Problems” [GUMM]

- Ausführung in einzelnen Schritten
- Jeder Schritt besteht aus einer einfachen Grundaktion
- Eindeutigkeit: es muss zu jeder Zeit klar sein welcher Schritt folgt.

“Eine Berechnungsvorschrift zur Lösung eines Problems heißt genau dann Algorithmus, wenn eine zu dieser Berechnungsvorschrift äquivalente ***Turingmaschine*** *existiert, die für jede Eingabe, die eine Lösung besitzt, stoppt.”*
[https://de.wikipedia.org/wiki/Algorithmus#Definition]

→ **alles was eine Turingmaschine kann, kann ein Computer!**

Turingmaschine

- “*is a universal computing model*”
- Kann jedes algorithmische Problem lösen.
- (theoretisch) unendlichen Band, eingeteilt in Felder
- pro Feld genau ein Zeichen
- Schreib- und Lesekopf
- internen Zuständen

Definition

Programme der Turingmaschine bestehen aus Tripel der Form: **Ausgabe**, **Bewegung**, **Zustand**

Ausgabe: Das Symbol, das vom Schreib- / Lesekopf auf das Band geschrieben wird
0,1,2,3 etc.

Bewegung: Die Richtung, in die sich der Schreib- / Lesekopf bewegt.

L	oder	**<**	(ein Schritt nach links)
R	oder	**>**	(ein Schritt nach rechts)
S	oder	**-**	(stehenbleiben)

Zustand: Der Zustand, in den nach Ausgabe und Bewegung gewechselt wird.
S (Start), **Z0**, **Z1**, etc.

HALT (terminiert die Turingmaschine)

Funktionsweise

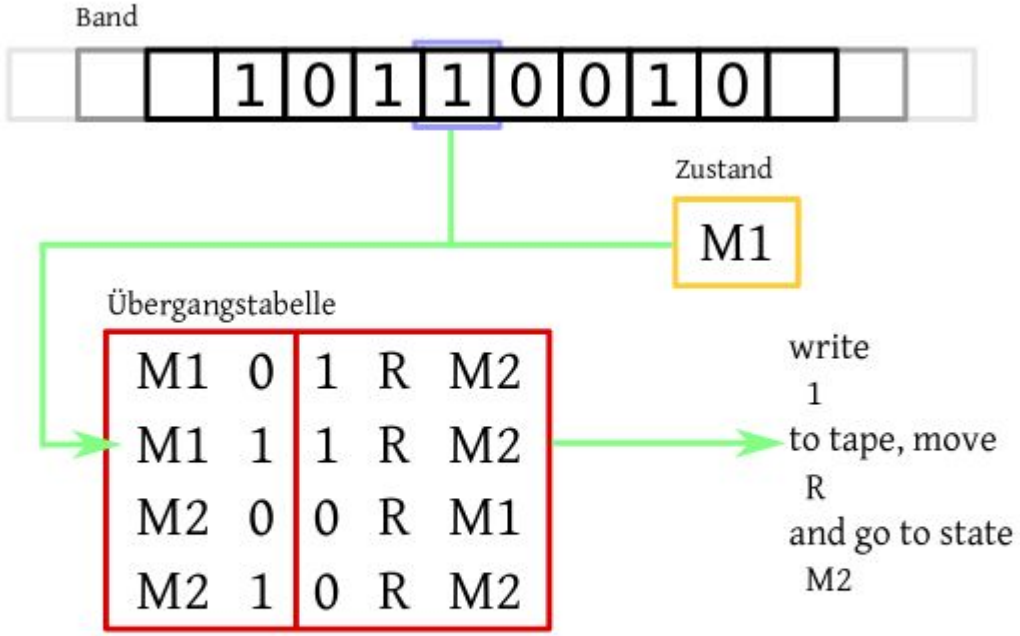


Abbildung 5.1: Die Funktionsweise einer Turingmaschine visualisiert.

http://gdi.ist.tugraz.at/skriptum.pdf

Beispiel: Gerade Zahlen von 0’en

Gegeben sei eine Turingmaschine:

- Auf dem Band befindet sich eine beliebige Sequenz von Nullen (0)
- Links und rechts umgeben von 1ern.
- Der Cursor befindet sich auf der ersten 1 links der ersten 0: 11**1**0001111
- Gehe in den Endzustand „Zgerade“ wenn sich eine gerade Anzahl an 0 auf dem Band befinden.
- Gehe in den Endzustand „Zungerade“ wenn sich eine ungerade Anzahl an Ziffern auf dem Band befinden.
- Das Band muss im Endzustand gleich wie am Anfang aussehen.
- Schreibe das Programm der Turingmaschine, um dieses Problem zu entscheiden.

Zustände:	{S,Z0, Z1, Zgerade, Zungerade}	**Accept:** 100001
Arbeitsalphabet:	{1,0}	**Reject:** 10001
Überführungsfunktion:	Zustandstabelle	
Startzustand:	{S}	
Haltezustand:	{Zgerade}	

Ablauf

```
Input:  11000011

S:      11000011
Z0:     11000011
Z1:     11000011
Z0:     11000011
Z1:     11000011
Z0:     11000011
Zgerade: accept
```

```
Input:  1100011

S:      1100011
Z0:     1100011
Z1:     1100011
Z0:     1100011
Z1:     1100011
Zungerade: reject
```

Input

S: ...1100001 1...

Z0: 100001

Z1: 100001

Z0: 100001

Z0: 100001

Z1: 100001

Z0: 100001

Output

1100001 1

S, 1	1, R, Z0
Z0, 0	0, R, Z1
Z1, 0	0, R, Z0
Z0, 1	1, -, Zgerade
Z1, 1	1, -, Zungerade

https://turingmachinesimulator.com

```
name: Even amount of zeros
init: S
accept: Zgerade

S,1
Z0,1,>

Z0,0
Z1,0,>

Z1,0
Z0,0,>

Z0,1
Zgerade,1,-

Z1,1
Zungerade,1,-
```

current	read		new	write	direction	comment
S	1	→	Z0	1	R	Start und beginne bei Z0
Z0	0	→	Z1	0	R	Geh nach rechts
Z1	0	→	Z0	0	R	Geh nach rechts
Z0	1	→	Zgerade	1	S	HALT, es muss eine gerade Zahl sein
Z1	1	→	Zungerade	1	S	Reject, es muss eine ungerade Zahl sein

Zustandstabelle

Turing-Vollständigkeit (= turingmächtig)

Turing-Vollständigkeit bezeichnet in der Berechenbarkeitstheorie die Eigenschaft einer Programmiersprache sämtliche Funktionen berechnen zu können, die eine universelle Turingmaschine berechnen kann.

[https://de.wikipedia.org/wiki/Turing-Vollst%C3%A4ndigkeit]

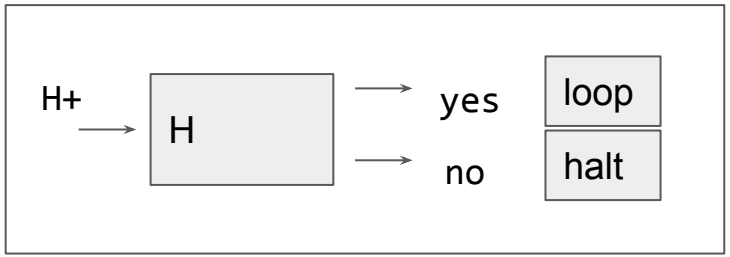
vollständig:	Java, Python, C, XSLT, etc.
nicht vollständig:	HTML, Reguläre Ausdrücke, etc.

Halteproblem/Entscheidungsproblem

Es gibt **keinen** Algorithmus, der über einen anderen Algorithmus aussagen kann, ob dieser in endlicher Zeit abbricht oder für immer in einer Schleife bleibt.

[https://de.wikipedia.org/wiki/Halteproblem]

? … ein beliebiges Programm
H … ein Programm, das das Halteproblem löst
(dieses Programm gibt es nicht)



H+

Wenn yes, dann loop
Wenn no, dann halt… ein Paradoxon

Beispiel: Verdoppelung von 1er

- Aus „1100000“ wird „11011“
- Aus „11110000000“ wird „111101111“
- Nach den 1ern folgen unendlich 0en

Gegeben ist folgende Turingmaschine:

Zustände: {Z1, Z2, Z3, Z4, Z5, Z6}
Arbeitsalphabet: {1,0}
Überführungsfunktion: Zustandstabelle
Startzustand: {Z1}
Haltezustand: {Z6}

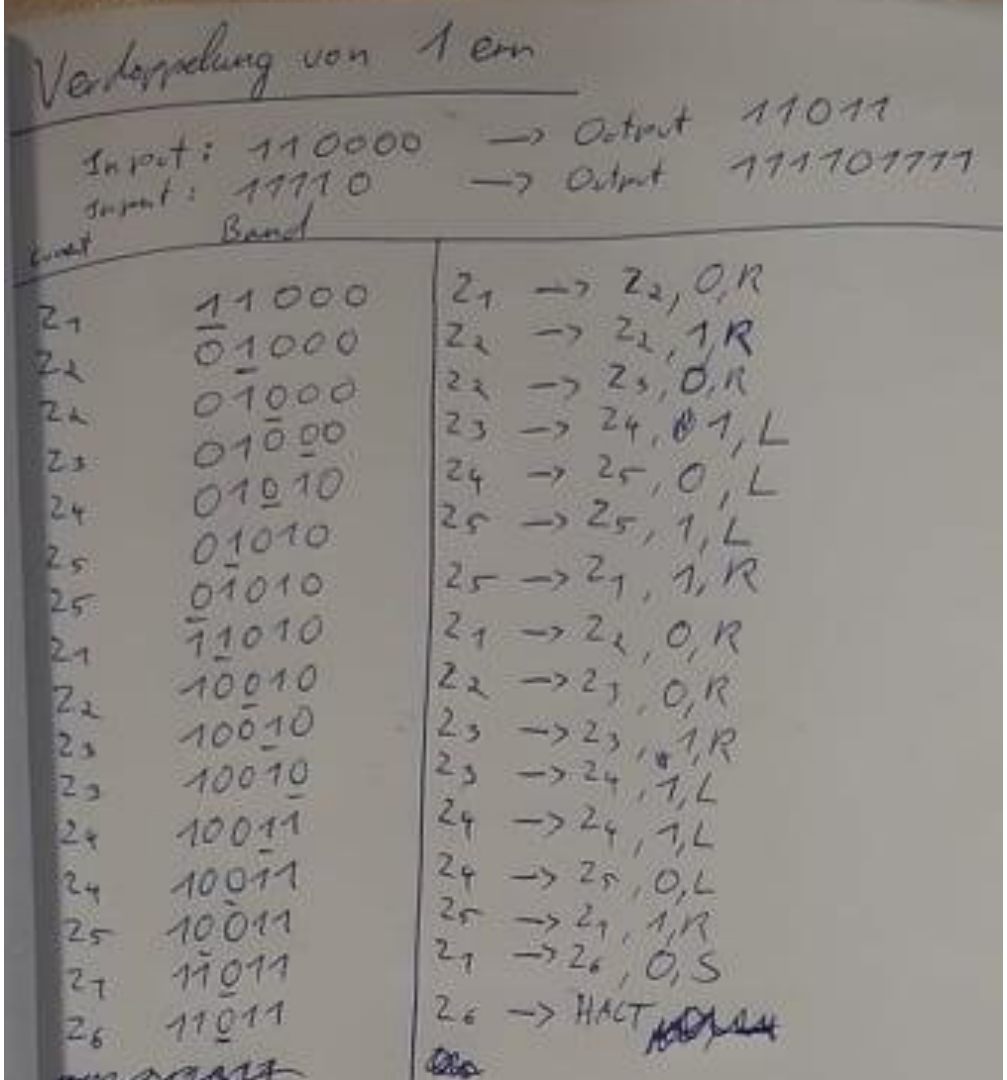
Start: **1**1000
Ende: 11**0**11

current	read		new	write	direction	comment
Z1	1	→	Z2	0	R	nimm 1 und geh nach rechts
Z1	0	→	Z6	0	S	HALT
Z2	1	→	Z2	1	R	geh nach rechts
Z2	0	→	Z3	0	R	geh nach rechts, erste 0
Z3	1	→	Z3	1	R	schreibe weitere verdoppelte 1
Z3	0	→	Z4	1	L	schreibe verdoppelte 1
Z4	1	→	Z4	1	L	hole die nächste 1, Fall: 111
Z4	0	→	Z5	0	L	hole die nächste 1, Fall: 100
Z5	1	→	Z5	1	L	hole die nächste 1
Z5	0	→	Z1	1	R	hole die nächste 1

Zustandstabelle und Formalisierung für https://turingmachinesimulator.com

current	read		new	write	direction	comment
Z1	1	→	Z2	0	R	nimm 1 und geh nach rechts
Z1	0	→	Z6	0	S	HALT
Z2	1	→	Z2	1	R	geh nach rechts
Z2	0	→	Z3	0	R	geh nach rechts, erste 0
Z3	1	→	Z3	1	R	schreibe weitere verdoppelte 1
Z3	0	→	Z4	1	L	schreibe verdoppelte 1
Z4	1	→	Z4	1	L	hole die nächste 1, Fall: 111
Z4	0	→	Z5	0	L	hole die nächste 1, Fall: 100
Z5	1	→	Z5	1	L	hole die nächste 1
Z5	0	→	Z1	1	R	hole die nächste 1

```
name: 1er verdoppeln
init: Z1
accept: Z6

Z1,1
Z2,0,>

Z1,0
Z6,0,-

Z2,1
Z2,1,>

Z2,0
Z3,0,>

Z3,1
Z3,1,>

Z3,0
Z4,1,<

Z4,1
Z4,1,<

Z4,0
Z5,0,<

Z5,1
Z5,1,<

Z5,0
Z1,1,>
```



Verdoppelung von 1 en

Input: 110000 → Output 11011
Input: 1111 0 → Output 111101111

Zustand	Band	
z_1	11000	$z_1 \to z_2, 0, R$
z_2	01000	$z_2 \to z_2, 1, R$
z_2	01000	$z_2 \to z_3, 0, R$
z_3	01000	$z_3 \to z_4, 1, L$
z_4	01010	$z_4 \to z_5, 0, L$
z_5	01010	$z_5 \to z_5, 1, L$
z_5	01010	$z_5 \to z_1, 1, R$
z_1	11010	$z_1 \to z_2, 0, R$
z_2	10010	$z_2 \to z_3, 0, R$
z_3	10010	$z_3 \to z_3, 1, R$
z_3	10010	$z_3 \to z_4, 1, L$
z_4	10011	$z_4 \to z_4, 1, L$
z_4	10011	$z_4 \to z_5, 0, L$
z_5	10011	$z_5 \to z_1, 1, R$
z_1	11011	$z_1 \to z_6, 0, S$
z_6	11011	$z_6 \to$ HALT

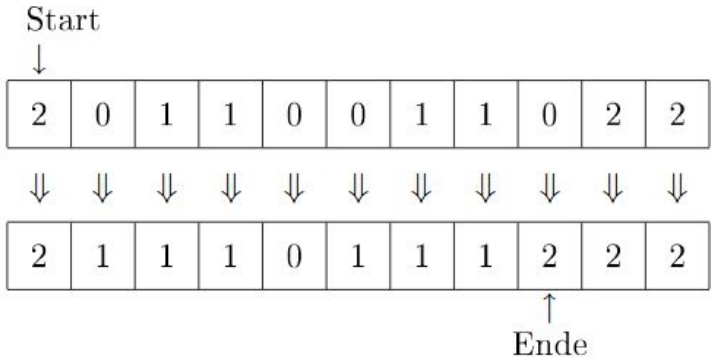
Übung

Gegeben ist folgende Turingmaschine:

Zustände:	{Z1, Z2, Z3, Z4, Z5}
Arbeitsalphabet:	{1,0,2}
Überführungsfunktion:	
Startzustand:	{Z1}
Haltezustand:	{Z5}

Die Turingmaschine ersetzt alle Nullen mit dem Zeichen, das sich vor Ausführung der Turingmaschine rechts neben der jeweiligen Null befindet. Nachdem sie alle Ersetzungen durchgeführt hat, geht sie in den Haltezustand Z5, sobald sie eine Zwei als Eingabezeichen findet.
Das Eingabeband ist am Anfang mit Nullen und Einsen gefüllt, die links und rechts von unendlich vielen Zweiern umgeben sind.

Band vor dem Programmablauf



Band nach dem Programmablauf

Was fehlt?

aktuell	read		neuer	write	Richtung	Kommentar
Z1	1	→	Z2	2	R	
Z1	2	→	Z2	2		
Z2	0	→	Z3	1	R	
Z2	1	→				
Z2	2	→	Z5	2	L	
Z3	0	→			L	
Z3	1	→	Z2	1	L	
Z3	2	→	Z1	2	L	
Z4		→	Z2	0	R	

https://turingmachinesimulator.com

aktuell	read		neuer	write	Richtung	Kommentar
Z1	1	→	Z2	2	R	
Z1	2	→	Z2	2	**R**	Beginn
Z2	0	→	Z3	1	R	Ersetze 0 durch 1
Z2	1	→	**Z2**	**1**	**R**	1 passt, geh weiter
Z2	2	→	Z5	2	L	
Z3	0	→	**Z4**	**0**	L	Wenn 0, geh zurück überprüfen, ist keine 1, neuer Zustand Z4 um 0 ersetzen zu schaffen
Z3	1	→	Z2	1	L	Wenn 1, dann geh einmal zurück und überprüf ob 1
Z3	2	→	Z1	2	L	Fall, wenn 2, wie bei 0, schreib und geh zum nächsten und überprüfe.
Z4	**1**	→	Z2	0	R	
Z5			HALT			

name: 1er verdoppeln
init: Z1
accept: Z5

#input= 20110011022
#output= 21110111222

Z1, 1
Z2, 2, >

Z1, 2
Z2, 2, >

Z2, 0
Z3, 1, >

Z2, 1
Z2, 1, >

Z2, 2
Z5, 2, <

Z3, 0
Z4, 0, <

Z3, 1
Z2, 1, <

Z3, 2
Z1, 2, <

Z4, 1
Z2, 0, >

Assignment 4 https://moodle.uni-graz.at/mod/assign/view.php?id=162580

Gegeben ist die Turingmaschine mit dem Eingabeband und dem Programm aus folgender Tabelle. Die drei rechten Spalten definieren das Verhalten, das für die Eingabe (erste zwei Spalten) gilt. Die Bewegungsrichtungen sind rechts (R) und links (L).

EINGABEBAND
Fett ist jeweils Start und Haltezustand.

*Start: **0**122112210*
*Ende: 0222122**0**0*

Zustände: (Z1, Z2, Z3, Z4 und HALT)
Arbeitsalphabet: 0,1,2
Startzustand: Z1
Haltezustand: HALT

Verhalten der Turingmaschine

Die Turingmaschine ersetzt alle Einser mit dem Zeichen, das sich vor Ausführung der Turingmaschine **rechts** neben der jeweiligen Eins befindet. Nachdem sie alle Ersetzungen durchgeführt hat, geht sie in den Zustand HALT, sobald sie eine Null als Eingabezeichen findet. Das Eingabeband ist am Anfang mit Einsen und Zweiern gefüllt, die links und rechts von unendlich vielen Nullen umgeben sind.

Assignment 4

https://moodle.uni-graz.at/mod/assign/view.php?id=162580

1) Vervollständige die fehlenden Werte in der Zustandstabelle (unten). Schreibe den ganzen Ablauf (per Hand) der Turingmaschine mit gegebenen Start und Endzustand auf. [4 Punkte]

2) Gib weiters an, wie oft sich die Turingmaschine bei der Ausführung mit dem gegebenen Eingabeband im Zustand Z3 befindet. [2 Punkte]

(Hier gibt es zwei Lösungen je nachdem, wie die Zustandstabelle aussieht.)

3) Schreibe einen Code für https://turingmachinesimulator.com, der bei gegebenem Input gegebenen Output ausgibt. [4 Punkte]
(Code im turingmaschinesimulator ausführen)

Zustandstabelle:

Z1	2	Z3	0	R
Z1	0	Z3	0	R
Z2	2	Z3	1	R
Z3	1	Z4		R
Z3	2	Z3		
Z3	0	HALT	0	
Z4	1	Z2	1	L
Z4	2	Z3	2	
Z4		Z1		L

Bonus: Zeichnen einen endlichen Automaten und stelle fest, ob folgende Terme auf gegeben RegEx matchen:
`a*b+a(b+a|aa+b)*ab`

ababaab, bbbaaaaabab, aabbbab, baab, babaaabbaab, abaaaabbbaab

Ass 3

1) Konstruiere einen RegEx der auf folgende Terme matcht: **abbca, abbbcca, ac, accca, acc** **Marie S.**

2) Gib an welche der folgenden Terme auf den angegebenen Regulären Ausdruck matchen: **ac?c+a?a*ca?** **Helena P.**

- *acaaacccaaa*
- *aaaaaaca*
- *acabca*
- *acc*

3) Gib an welche der folgenden Terme auf den angegebenen Regulären Ausdruck matchen. Hier werden Zeichenklassen etc. verwendet (die könnten von Tool/Programmiersprache leicht unterschiedlich sein.) : **\d+.(\d{2}|M\w+).\d{2}[5|6]0 Markus K.**

- *01.02.1950*
- *1.02.1906*
- *100.00.1960*
- *01.Mai.1950*
- *01.Maerz.1950*

4) Gib an welche der folgenden Terme auf den angegebenen Regulären Ausdruck matchen: **(ac(c+a?)?|(a+(c|c+)?(caac)?ca)+(ca)?)+ Stefan T.**

- accaaccaacaaccaca
- acaaac
- accaccca
- cacaaa

5) Zeichne einen endlichen Automaten, der genau die Wörter akzeptiert, die durch folgenden RegEx gematcht werden. Versuche ihn so vereinfacht wie möglich zu zeichnen. Gib alle Endzustände an: **(ac+a*)|(c+a*c)? Sabine Z.**

6) Definiere ein Alphabet **Σ** für die formale Sprache der Addition, Subtraktion, Multiplikation und Division von Rationalen Zahlen (+, -, Komma). Gib 2 gültige und zwei ungültige Wörter dieser Sprache an. Beschreibe die Regeln dieser formale Sprachen in Worten**.** **Stefanie S.**

7) Zeichne eine Syntaxdiagramm für die formale Sprache der Addition, Subtraktion, Multiplikation und Division von Rationalen Zahlen. **Stefanie S.**

8) Schreibe einen RegEx der auf ISBN-10 und ISBN-13 Nummern matcht inklusive dem Text "ISBN-10:" oder "ISBN-13:". Verwende dafür explizit Zeichenklassen. Du kannst z.B: dieses Web-Tool verwenden: https://regexr. **Jakov G.**

ISBN-13: 978-3-86680-192-9, ISBN-10: 3-86631-007-2, ISBN-10: 111-3-86631-007-2 ISBN-13: 978-3-86631-007-0

ISBN-12: 973-3-86684-192-9, 973-3-86684-192-9, 973-3a-86684-192-9, ISBN-10: 11-33-86631-007-2

[https://de.wikipedia.org/wiki/Internationale_Standardbuchnummer#ISBN-13]

9) Verwende den Befehl *egrep* im Terminal, um aus den zwei Dateien *uk-500.csv* und *us-500.csv* (auf Moodle zum Herunterladen) mittels eines RegEx alle Email-Adressen auszulesen, die nach dem @ ein „*gmail.com.*“ beinhalten. **Sina K.**

10) Verwende den Befehl *egrep* im Terminal, um aus den zwei Dateien *uk-500.csv* und *us-500.csv* mittels eines RegEx alle HTML-Adressen auszulesen. Schreibe das Ergebnis in eine neue Datei, direkt in der Konsole. Der Befehl dafür schaut so aus. Erkläre auch ungefähr diesen Befehlsaufruf in der Konsole:

egrep -i -oh HIER STEHT DEIN REGEX uk-500.csv > mails.txt **Sina K.**

https://www.computerhope.com/unix/uegrep.htm

Bonus: **Vera C.**

Zeichne einen Parse Tree für folgende Formel und verwende dieses Modell: p = F, q = T, r = F. Ist diese Formel erfüllbar? Erstelle auch eine Wahrheitstabelle und die KNF für diese Formel: **((q -> ¬p) v r) -> (q ⋀ (r -> p))**

RegEx

```
(ac(c+a?)?|(a+(c|c+)?(caac)?ca)+(ca)?)+
```

accaccca matcht nicht im https://regexr.com/ aber mit dem egrep Befehl

```
egrep '(ac(c+a?)?|(a+(c|c+)?(caac)?ca)+(ca)?)+' regex.txt
```

```
B\w+?
```

b *be bee beer beers*

- lazy/greedy RegEx
- Spezifikation wie RegEx umgesetzt werden