

Grundfragen der Informatik, KV

Formale Sprachen,
Reguläre Ausdrücke und
Automaten



Überblick

Formale Sprachen

Syntaxdiagramm

Endliche Automaten (finite-state automaton)

Reguläre Ausdrücke (regular expressions)

REGULAR EXPRESSIONS



Es gibt unterschiedliche Sprachen

Natürliche Sprachen

hallo

$$\Sigma = \{a \dots z\}$$

DNA

$$\Sigma = \{a, c, g, t\}$$

Morse

-- --- .- . . . / -.- --- -.- .

$$\Sigma = \{-, \cdot, /\}$$

Programmiersprachen

```
if temperature > 70: print('Wear shorts.')  
else: print('Wear long pants.')
```

$$\Sigma = \{\text{if}, >, \text{print} \dots\}$$

Binär

$$\Sigma = \{0, 1\}$$

Formale Sprachen

Sprache	...	beschreibt eine beliebige Menge von Wörtern
Alphabet (Σ)	...	ist eine endliche, nicht leere Menge. Jedes $a \in \Sigma$ heißt Buchstabe oder Terminalsymbol.
Wort (w)	...	besteht aus mindestens einem Terminalsymbol eines Alphabets, endliche Folge aus einem Alphabet
leeres Wort (ϵ)	...	ist ein Wort, $\{\epsilon\}$, $ \epsilon = 0$, $ \text{hallo} = 5$

- Formale Sprachen können leer, endlich oder unendlich sein.
- Nicht Kommunikation, sondern mathematische Verwendung im Vordergrund.
- Es gibt keine Information über Bedeutung!

Eine einfache Sprache

Wortproblem:

“Gegeben sei eine Sprache S . Ermittle, ob ein Wort W Teil von S ist oder nicht.”

Sprache lässt sich als Menge repräsentieren.

$S = \{“a”\}$

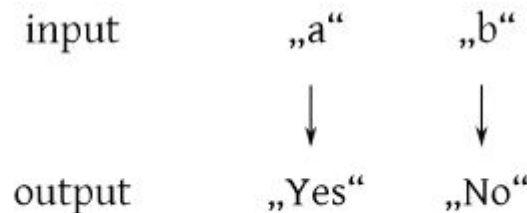


Abbildung 11.1: Zwei Beispiele für die Entscheidung des Wortproblems für die Sprache „a“.

Formale Sprachen

Sie können über eine mathematische Bedingung an ihre Wörter definiert sein: „Die Sprache ... ist die Menge aller Wörter, für die gilt ...“.

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

$$S1 = \{aa, bb, cc, aab, cab, a\}$$

Sprache S1 hat 6 Wörter

$$S2 = \{a^n, b^n, c^n \mid n \in \mathbb{N}\}$$

abc, aabbcc, aaabbbccc

...

Zwei grundlegende Möglichkeiten formale Sprachen zu beschreiben

Über die Definition einer Grammatik

... um eindeutig festzulegen, ob ein Wort Element einer Sprache ist und zum anderen, um Eigenschaften dieser formalen Sprachen zu untersuchen bzw. zu beweisen.

Über Automaten

... die ein Wort Zeichen für Zeichen analysieren und akzeptieren, wenn es einer Sprache zugehörig ist.

(→ Compiler)

Formale Sprache der Gleitkommazahlen

Erstellen wir ein **Syntaxdiagramm** (abstrakte Darstellung eines Automaten) für alle erlaubten Wörter der Gleitkommazahlen mit wissenschaftlicher Notation.

Ein **Syntaxdiagramm** ist ein Weg um kontextfreie Sprache (kontextfreie Grammtiken) zu repräsentieren.

Folgende Wörter sind **erlaubt** in dieser Sprache:

1.0e3, -42. , +0.43, 156E+10., .73, 2.3e-5, 2.3e-5.2, ...

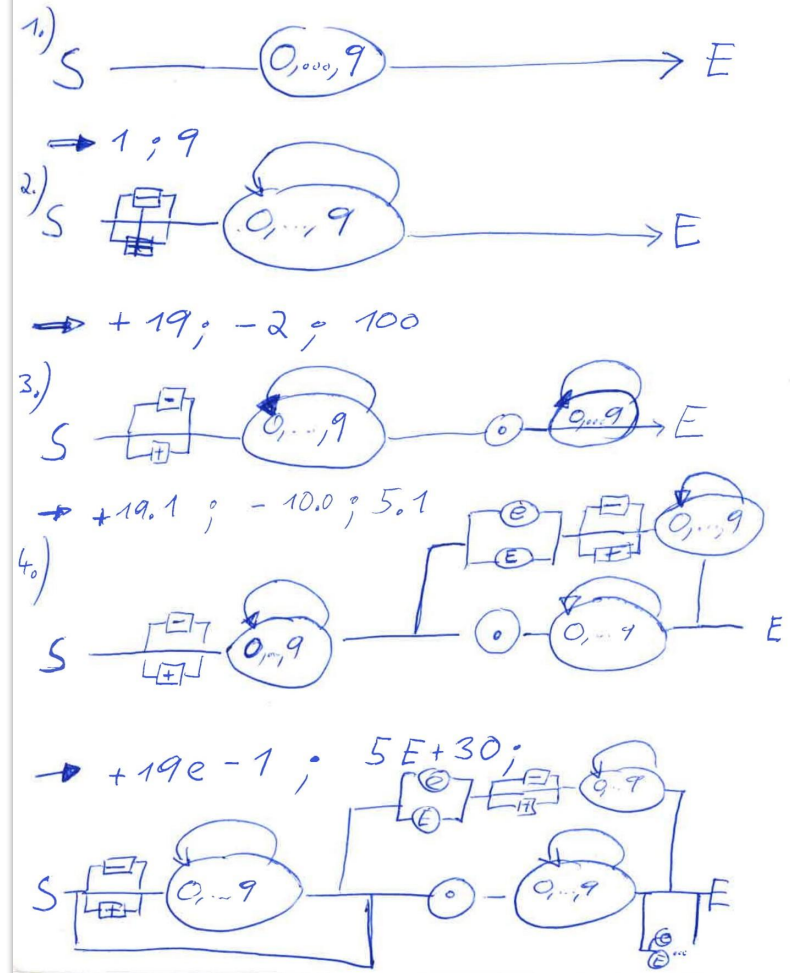
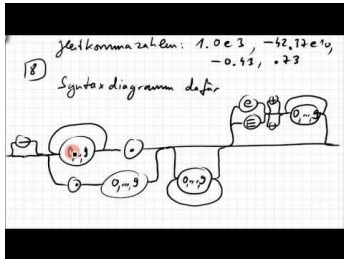
[2.3e-5 = $2.3 * 10^{-5} = 0.000023$]

Folgende Wörter sind **verboten** in dieser Sprache: -e.45, EE123, +++, ...

Gleitkommazahlen

Folgende Wörter sind **erlaubt** in unserer Sprache:

1.0e3, -42. , +0.43, 156E+10., .73,
2.3e-5

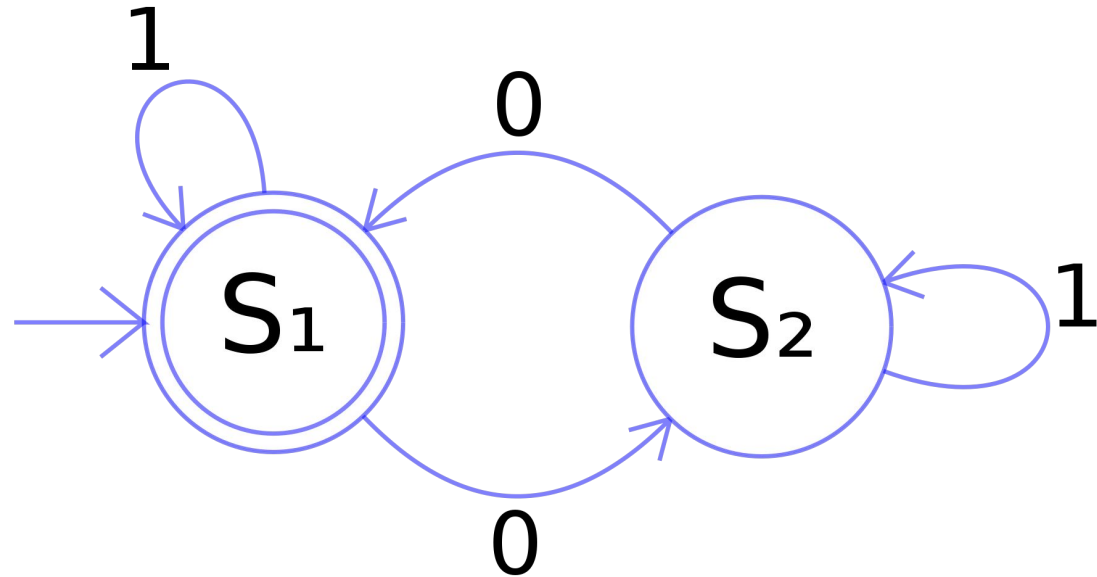


Endlicher Automat

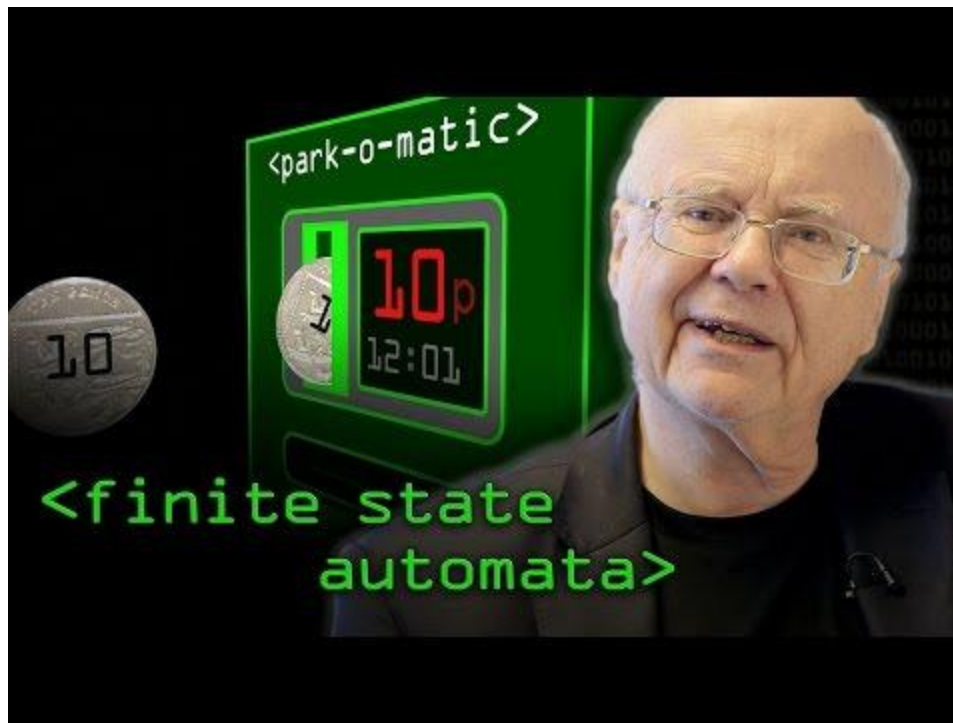
Dieser Endliche Automat überprüft ob eine Binärzahl eine gerade oder ungerade Anzahl von 0en hat:

Input: 10001
Not accepted

Input: 1001
Accepted

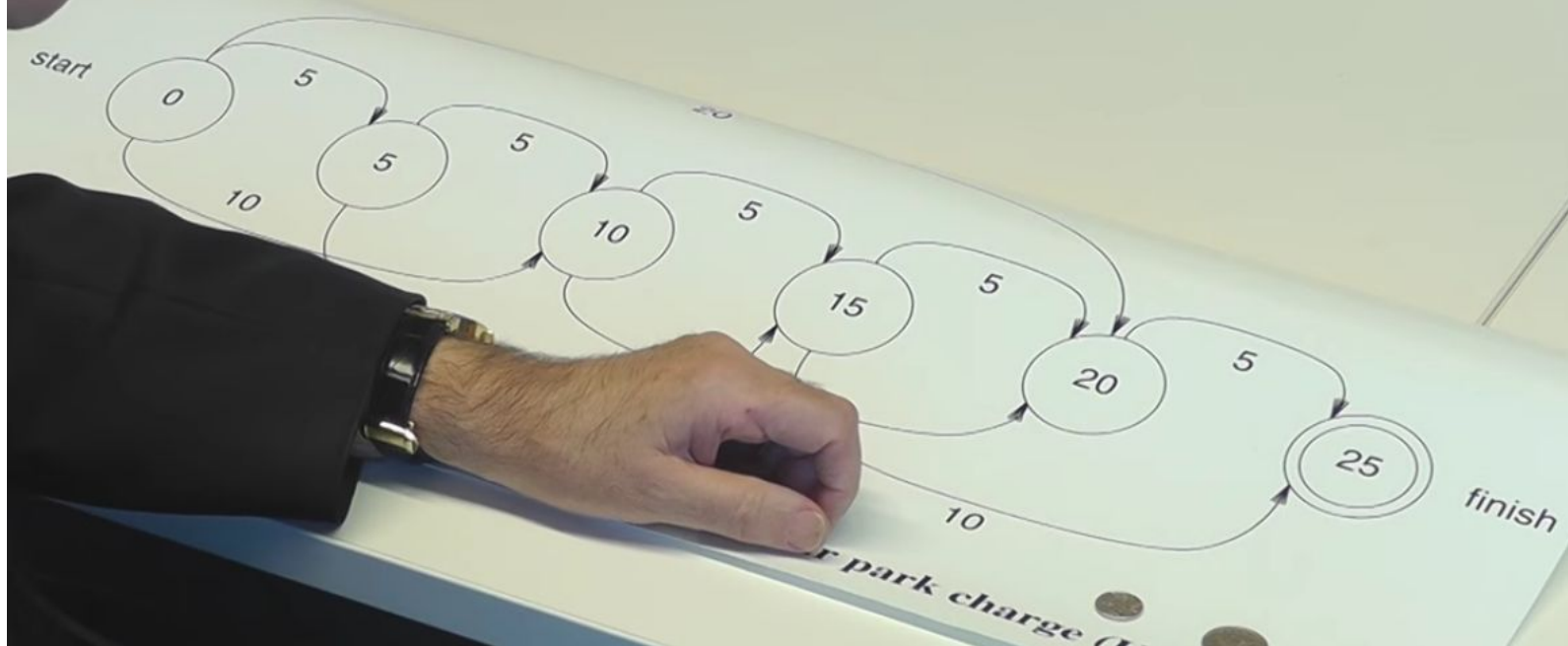


- Startzustand
- Gerichtet
- Endzustände



Computers Without Memory - Computerphile

https://en.wikipedia.org/wiki/Finite-state_machine



Ein **Endlicher Automat** analysiert ein Wort einer Sprache Zeichen für Zeichen und entscheidet, ob sie zur Sprache gehört oder nicht. Kein Speicher ist dafür notwendig. Wird beispielsweise zum Überprüfen von Variablennamen verwendet.

Alphabet	...	5,10, 20
Wörter	...	5 5 5 5 5, 5 10 10, 20 5, 10 5 5 5...
Zustand	...	0, 5, 10, 15, 20
Endzustand	...	25

Reguläre Ausdrücke: RegEx (regular expressions)

- Werden durch reguläre Grammatiken erzeugt.
- Zu jedem regulären Ausdruck existiert ein endlicher Automat, der die vom Ausdruck spezifizierte Sprache akzeptiert.
- z.B. die Sprache aller Wörter, die aus beliebig vielen a oder beliebig vielen b und der leeren Menge besteht:

Alphabet: {a,b}

Wörter: ab, a, b, aaaaa, bbbbbb, ababab, ε

RegEx: (a*b*)*, a+, b?, ab, a, b

Verwendung von RegEx

R1R2 wenn R1, R2 sind RegEx

(R1|R2) ODER

(R1)*

Konkateniert

Reguläre Ausdrücke $\Sigma = \{a, b, \epsilon\}$
 z.B. a^*bca^* bc aabca aaaaa
 $a(b|c|d)a$ abca acba
 $x \in \Sigma$ a
 ϵ



Quantoren

Beziehen sich immer auf voranstehende Ausdrücke (bzw. Klammern)

- ?** ist optional, kommt null- oder einmal
- +** mindestens einmal vorkommen, ein- oder mehrfach
- *** beliebig oft, kein- oder mehrfach
- ()** Zusammenfassen von Termen
- |** Alternative

RegEx

a+ aaaa, a, aaa

$ab?$ a, ab

(ab)? ab, ε

abab^* ababbbbbbb, aba, ababb

$$(aab+)^* \quad aab, aabbbb, aabaab, aabbaabbbbaab, \varepsilon$$

$(a+b+ \mid (bba)^+)^+$ aaabbb, bbabbabba, abbbbaabbbbabba

RegEx als endlichen Automaten

Automaten für:

a^+

$ab^?$

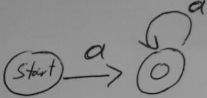
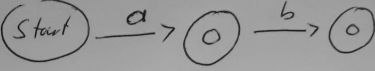
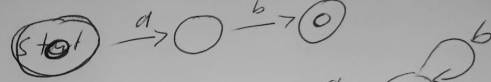
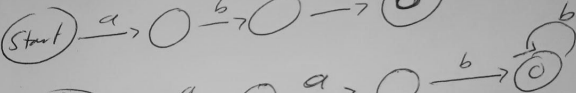
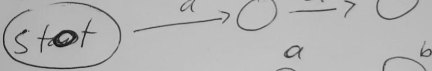
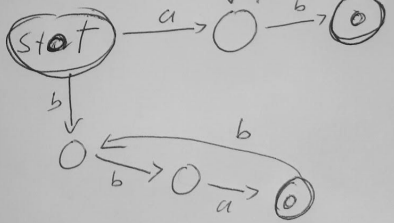
$(ab)^?$

$abab^*$

$(aab^+)^*$

$a^+b^+|(bba)^+$

RegEx - Endliche Automaten

RegEx	endlicher Automat
a^+	
$ab^?$	
$(ab)^?$	
$abab^*$	
$a(ab^+)^*$	
$a^+b^+ (bba)^+$	

Automaten für:

a^+

$ab^?$

$(ab)^?$

$abab^*$

$(aab+)^*$

$(a+b+|(bba)+)$

Übung RegEx matcht?

Gib an welche der folgenden Terme auf den angegebenen Regulären Ausdruck matchen.

Regulärer Ausdruck: **((ac)+ | (bb)?d)+**

Term	matcht?
------	---------

acacacacac	
------------	--

acbbd	
-------	--

d	
---	--

acb	
-----	--

RegEx konstruieren

Konstruiere einen RegEx der auf folgende Terme matcht:

ε , eeee, de, deee, deef

Lösung:

$(d?e+f?)?$ oder $d^*e^*f^*$ oder...

RegEx als Automaten zeichnen

Zeichne einen endlichen Automaten für **$(d^*e^+)f?$**

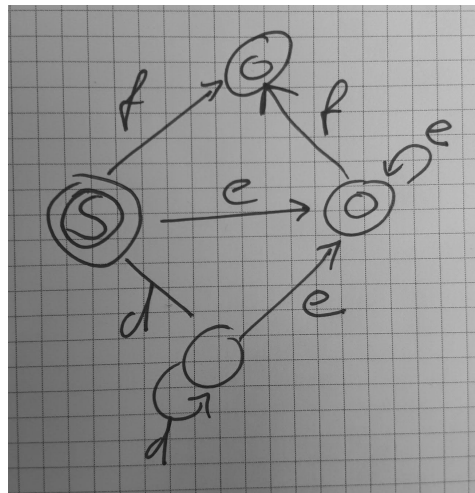
Füge sowohl Startzustand (S), als auch alle Endzustände (Doppelkreis) an.

RegEx als Automaten zeichnen

Zeichne einen endlichen Automaten für

$(d^*e^+)?f?$

Füge sowohl Startzustand (S), als auch alle Endzustände (Doppelkreis) an.



Weitere RegEx Operatoren

Vordefinierte Zeichenklassen:

\d	digit	eine Ziffer, [0-9] (und evtl. auch weitere Zahlzeichen in Unicode)
\D	no digit	ein Zeichen, das keine Ziffer ist, also [^\d]
\w	word character	ein Buchstabe, eine Ziffer oder der Unterstrich, also [a-zA-Z_0-9]
\W	no word character	ein Zeichen, das weder Buchstabe noch Zahl noch Unterstrich ist, also [^\w]
\s	whitespace	meist zumindest das Leerzeichen und die Klasse der Steuerzeichen \n, \r, \t
\S	no whitespace	ein Zeichen, das kein Whitespace ist, also [^\s]

[https://de.wikipedia.org/wiki/Regul%C3%A4rer_Ausdruck]

Online RegEx Check-Tool

<https://regexr.com>

folgende Dinge:

- Alle Zeichen
- Alle Zahlen
- Alle Wörter, alle Wörter die mit 'B' beginnen
- Alle URI's
- Alle Datumsangaben [01.01.1991]
- Alle Wörter
- Alle Zahlen

{ } ... so viele Zeichen

(a|b) ... a oder b

[a-g] ... Zeichen zwischen a und g

^ ... not

Lösungen

- `\w`
- `\d`
- `\w+`, `\bb\w+`
- `http:\V(\w|\d|\.|\\b|\\V)+`
(bis zu ?, ? oft ein Parameter)
- `\\b\\d{2}\\.|\\d{2}\\.|\\d{4}\\b`

Übungen

Ass 3



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

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

- *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+(\{2\}|M\w+)\.d{2}[5|6]0**

- *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)?+**

- *accaaccaacaaccaca*
- *acaaac*
- *accacca*
- *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: $(ac+a^*)(c+a^*c)^?$
- 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.
- 7) Zeichne eine Syntaxdiagramm für die formale Sprache der Addition, Subtraktion, Multiplikation und Division von Rationalen Zahlen.
- 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>.

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.

- 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`

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

Bonus:

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

- $((q \rightarrow \neg p) \vee r) \rightarrow (q \wedge (r \rightarrow p))$

Assignment 2

1.) Formalisiere jede der folgenden Aussagen durch eine aussagenlogische Formel:

- 1.1) *Wenn du die drei Fragen beantwortet hast, dann kannst du die Brücke des Todes überqueren.*
- 1.2) *Die Ritter der Kokosnuss gibt es nur zusammen mit Pferden und Kokosnüssen.*
- 1.3) *Entweder wird ein Gebüsch gekauft oder es wird kein Gebüsch gekauft.*

2.) Formuliere die Regel, "wenn der Hahn auf dem Misthaufen kräht, dann ändert sich das Wetter oder es bleibt so, wie es ist" als Formel der Aussagenlogik. Zeige, dass die Formel eine Tautologie ist (Wahrheitstabelle) Suzana Sagadin

3.) Begründe ob folgende Formeln erfüllbar, gültig oder unerfüllbar sind. Erfüllbar bist ist eine Formel, wenn mindestens einmal "wahr" in der Wahrheitstabelle auftaucht. Gültig wenn alle "wahr" sind und unerfüllbar wenn alle "falsch" sind.

- 3.1) $a \rightarrow a \wedge b$
- 3.2) $(a \rightarrow b) \rightarrow (b \rightarrow a)$
- 3.3) $(a \wedge b) \rightarrow (\neg a \vee \neg b)$
- 3.4) $(\perp \rightarrow p) \wedge \neg(q \longleftrightarrow p) \wedge \neg((\neg p \wedge \neg q) \rightarrow q)$

4.) Zeichne einen Parse Tree für folgende Formel und verwende den Tree, um zu überprüfen ob die Zuordnung $p = F, q = T, r = F$ die Formel true oder false werden lässt. Finde ein Modell, welches das Gegenteil erreicht.

$(\neg(r \longleftrightarrow q) \rightarrow \neg r) \wedge (\neg(r \rightarrow q) \vee (p \rightarrow q))$

5.) Gib folgende Befehle in deiner Konsole ein. Erkläre was jeweils passiert und um welche unterschiedlichen Operatoren es sich handelt.

<https://www.linux.com/tutorials/logical-ampersand-bash/>

- `echo $((2 & 3))`
- `echo $((2 && 3))`
- `echo $((5 | 5))`
- `echo $((5 || 5))`

Was ist die Konjunktive Normalform (KNF) des folgenden logischen Ausdrucks:
 $(p \rightarrow q) \wedge (\neg q \vee p)$

Lösungen Ass 2

1.1) $p \text{ implies } q$

1.2) $p \text{ impllies } (q \text{ and } r)$

1.3) $p \text{ xor } q$

1.4) $p \text{ euqivalent } p$

2.) $p \text{ implies } (q \text{ or } \text{not}(q))$

3.1) $a \text{ implies } (a \text{ and } b)$

3.2) $(a \text{ implies } b) \text{ implies } (b \text{ implies } a)$

3.3) $(a \text{ and } b) \text{ implies } (\text{not}(a) \text{ or } \text{not}(b))$

3.4) $(p \text{ and } \text{not}(p) \text{ implies } p) \text{ and } \text{not}(q \text{ EQUIVALENT } p) \text{ and } (\text{not}((\text{not}(p) \text{ and } \text{not}(q)) \text{ imples } q))$

4.) $(\text{not}(r \text{ EQUIVALENT } q) \text{ IMPLIES } \text{not}(r)) \text{ and } (\text{not}(r \text{ implies } q) \text{ or } (p \text{ implies } q))$