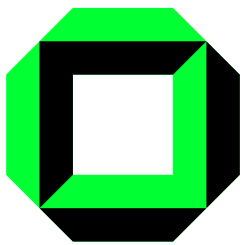
Info I – Übungsblatt 9

Joachim Breitner

mit Aufgaben von Martin Kiefel und Christian Maier

http://www.joachim-breitner.de/wiki/Infotut

16. Januar 2006



Unser Programm heute

1. Übungsblatt 8
2. Endliche Automaten
3. Reguläre Ausdrücke
4. OOP

Übungsblatt-Rückblick

Statistik

- Schnitt: 39 von 50 Punkten

Hinweise

- Bei Chomsky-Einschätung immer zeigen, warum weder die nächst-größere noch die nächst-kleinere Klasse ist.
- Bei Semi-Thue und Markov macht ein ε nur alleine Sinn. $\varepsilon a = a = a\varepsilon$! Es ist falsch, anzunehmen, dass ein ε den Rand des Wortes markiert.
- Beim Angeben von Grammatiken zu einer Sprache, vergesst nicht auf das leere Wort zu achten: ε rückwärts ist auch ε.

Wiederholung: Die Leeren der Leerer

Was ist der Unterschied zwischen dem leeren Wort und der leeren Sprache? Hier ein Beispiel:

- Sei L_i die Menge der gültigen Passwörter zum Login des Benutzers i
- Alan ist hat ganz normal ein Passwort, „gott“. Also: $L_{\text{Alan}} = \{\text{gott}\}$.
- Gast ist für den öffentlichen Zugang eingerichtet, hier muss man kein Passwort eingeben, um sich anzumelden: $L_{\text{Gast}} = \{\varepsilon\}$ (ε heißt auch das leere Wort)
- Karl hat zu viel gehackt und sein Zugang wurde gesperrt. Egal, was er eingibt, es ist nicht gültig: $L_{\text{Karl}} = \{\} = \emptyset$ (die leere Sprache)

Wiederholung: Die Leeren der Leerer

Was ist der Unterschied zwischen dem leeren Wort und der leeren Sprache? Hier ein Beispiel:

- Sei L_i die Menge der gültigen Passwörter zum Login des Benutzers i
- Alan ist hat ganz normal ein Passwort, „gott“. Also: $L_{\text{Alan}} = \{\text{gott}\}$.
- Gast ist für den öffentlichen Zugang eingerichtet, hier muss man kein Passwort eingeben, um sich anzumelden: $L_{\text{Gast}} = \{\varepsilon\}$ (ε heißt auch das leere Wort)
- Karl hat zu viel gehackt und sein Zugang wurde gesperrt. Egal, was er eingibt, es ist nicht gültig: $L_{\text{Karl}} = \{\} = \emptyset$ (die leere Sprache)

Wiederholung: Die Leeren der Leerer

Was ist der Unterschied zwischen dem leeren Wort und der leeren Sprache? Hier ein Beispiel:

- Sei L_i die Menge der gültigen Passwörter zum Login des Benutzers i
- Alan ist hat ganz normal ein Passwort, „gott“. Also: $L_{\text{Alan}} = \{\text{gott}\}$.
- Gast ist für den öffentlichen Zugang eingerichtet, hier muss man kein Passwort eingeben, um sich anzumelden: $L_{\text{Gast}} = \{\varepsilon\}$ (ε heißt auch das leere Wort)
- Karl hat zu viel gehackt und sein Zugang wurde gesperrt. Egal, was er eingibt, es ist nicht gültig: $L_{\text{Karl}} = \{\} = \emptyset$ (die leere Sprache)

Wiederholung: Die Leeren der Leerer

Was ist der Unterschied zwischen dem leeren Wort und der leeren Sprache? Hier ein Beispiel:

- Sei L_i die Menge der gültigen Passwörter zum Login des Benutzers i
- Alan ist hat ganz normal ein Passwort, „gott“. Also: $L_{\text{Alan}} = \{\text{gott}\}$.
- Gast ist für den öffentlichen Zugang eingerichtet, hier muss man kein Passwort eingeben, um sich anzumelden: $L_{\text{Gast}} = \{\varepsilon\}$ (ε heißt auch das leere Wort)
- Karl hat zu viel gehackt und sein Zugang wurde gesperrt. Egal, was er eingibt, es ist nicht gültig: $L_{\text{Karl}} = \{\} = \emptyset$ (die leere Sprache)

Chomsky-Hierarchie

ganz einfache Computer

Eine Maschine soll zu einer eingegebenen Zeichenreihe feststellen, ob diese Zeichenreihe zur einer Sprache gehört und ggf. ob nicht. Es besteht ein enger Zusammenhang zwischen den durch Grammatiken erzeugten Sprachen und den Maschinen.

- CH-0 Turing-Maschine
- CH-1 Linear beschränkter Automat
- CH-2 Kellerautomat
- CH-3 Endlicher Automat

Eine vertiefte Behandlung des Zusammenhangs zwischen Sprachen und Maschinen ist Teil der Berechenbarkeitstheorie.

Endlich ein Automat!

Wozu?

Ein endlicher Automat ist gerade mächtig genug, um ein Sprache aus CH-3 zu entscheiden. Der Vorteil von endlichen Automaten ist, dass sie sehr einfach zu implementieren sind.

Was braucht man?

- endliche Menge Q von Zuständen
- ein Anfangszustand $q_0 \in Q$
- Zeichenvorrat Σ

Endlich ein Automat!

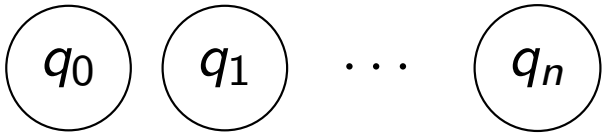
Wie arbeitet er?

Das Lesen eines Zeichens $a \in \Sigma$ führt zu einem Zustandsübergang vom aktuellen Zustand $q \in Q$ in einen neuen Zustand $q' \in Q$

- Notation: $qa \longrightarrow q'$
- Der Zustand läßt sich als ein Gedächtnis über die Vorgeschichte, also die bisher eingegebenen Zeichen, auffassen.
  Dieses ist leider nur endlich!

Darstellung von endlichen Automaten als Graphen

Zustände $Q = \{q_0, q_1, \ldots, q_n\}$ des endlichen Automaten lassen sich als Ecken eines Graphen auffassen

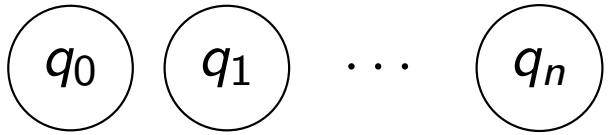


Zustandsübergänge $q_i a \longrightarrow q_j$ mit $a \in \Sigma$ entsprechen markierte gerichtete Kanten

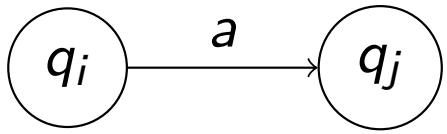
Ein im endlichen Automaten erreichter Zustand q_k ist durch den Anfangszustand q_0 und die bisher eingegebene Zeichenreihe $x = x_1 \ldots x_i$ bestimmt
Die Graph-Notation ist hierbei: $q_0 \Rightarrow^+ q_k$ bzw. $q_0 \Rightarrow^* q_k$

Darstellung von endlichen Automaten als Graphen

Zustände $Q = \{q_0, q_1, \ldots, q_n\}$ des endlichen Automaten lassen sich als Ecken eines Graphen auffassen



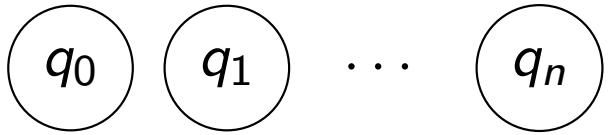
Zustandsübergänge $q_i a \longrightarrow q_j$ mit $a \in \Sigma$ entsprechen markierte gerichtete Kanten



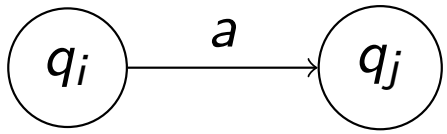
Ein im endlichen Automaten erreichter Zustand q_k ist durch den Anfangszustand q_0 und die bisher eingegebene Zeichenreihe $x = x_1 \ldots x_i$ bestimmt
Die Graph-Notation ist hierbei: $q_0 \Rightarrow^+ q_k$ bzw. $q_0 \Rightarrow^* q_k$

Darstellung von endlichen Automaten als Graphen

Zustände $Q = \{q_0, q_1, \ldots, q_n\}$ des endlichen Automaten lassen sich als Ecken eines Graphen auffassen



Zustandsübergänge $q_i a \longrightarrow q_j$ mit $a \in \Sigma$ entsprechen markierte gerichtete Kanten



Ein im endlichen Automaten erreichter Zustand q_k ist durch den Anfangszustand q_0 und die bisher eingegebene Zeichenreihe $x = x_1 \ldots x_i$ bestimmt
Die Graph-Notation ist hierbei: $q_0 \Rightarrow^+ q_k$ bzw. $q_0 \Rightarrow^* q_k$

Arten von Automarken

Es gibt zwei Arten, wie ein Automat eine Ausgabe haben kann. Wir unterscheiden dabei:

Mealy-Automat
- Erzeugung einer Ausgabe bei jedem Zustandsübergang
- Markieren der Kanten mit a/t_i

Moore-Automat
- Erzeugung einer Ausgabe bei Erreichen eines Zustands

In beiden Fällen ist die Ausgabe ein Wort $t = t_0 \dots t_{n-1}$ über einem Ausgabezeichenvorrat T. Die Ausgabe kann offensichtlich nicht länger sein als das Eingabewort.

Ein spezieller Moore-Automat

Akzeptor

- ist der häufigste Spezialfall eines Moore-Automaten
- Eine Ausgabe findet nicht bei allen Zuständen statt
- Die Zustände $F \subseteq Q$, bei denen eine Ausgabe erfolgt, heißen Endzustände
- Das Ausgegebenes Wort $t \in T^n$ hängt nur vom erreichten Endzustand $q \in F$ ab

Akzeptor unter dem Lupe

- Ein Akzeptor heißt vollständig, wenn für jeden Zustand und jede Eingabe ein neuer Zustand Definiert ist. Die kann immer durch die Einführung eines Fehlerzustandes geschehen.
- Ein endlicher Akzeptor lässt sich als Quintupel (Σ, Q, q_0, F, P) auffassen:

  Σ Zeichenvorrat
  Q nichtleere endliche Zustandsmenge
  q_0 Anfangszustand aus Q
  F nichtleere Menge von Endzuständen aus Q
  P Übergänge $qa \rightarrow q'$ mit $q, q' \in Q$, $a \in \Sigma$

- Die Sprache, die der Akzeptor akzeptiert: $L(A) = \{x | x \in \Sigma^*, q_0 x \Rightarrow^* q_e, q_e \in F\}$

(Nicht)Determinismus

Determinismus

Ein endlicher Automat, der die Eigenschaft besitzt, dass die Übergänge zusammen genommen eine (Übergangs-) Funktion bilden, heißt deterministisch
Die Übergangsfunktion:

$$\delta : Q \times \Sigma \rightarrow Q$$

Nichtdeteminismus

Im allgemeinen Fall wird Nichtdeterminismus zugelassen. Dann fasst man P als Semi-Thue-System mit Eingabezeichenvorrat Σ und syntaktischen Hilfszeichen $q \in Q$ auf. Die Beschreibung der Übergänge ist hier eine Relation

$$\delta \subseteq Q \times \Sigma \times Q$$

(Nicht)Determinismus

Determinismus

Ein endlicher Automat, der die Eigenschaft besitzt, dass die Übergänge zusammen genommen eine (Übergangs-) Funktion bilden, heißt deterministisch
Die Übergangsfunktion:

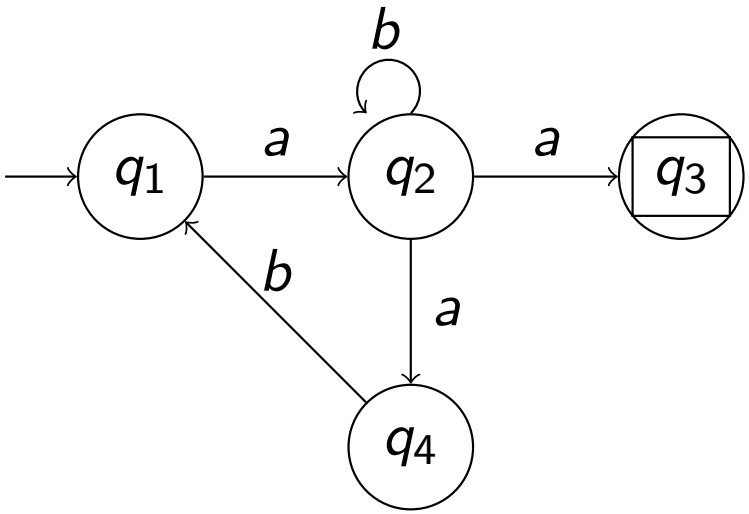
$$\delta : Q \times \Sigma \rightarrow Q$$

Nichtdeteminismus

Im allgemeinen Fall wird Nichtdeterminismus zugelassen. Dann fasst man P als Semi-Thue-System mit Eingabezeichenvorrat Σ und syntaktischen Hilfszeichen $q \in Q$ auf. Die Beschreibung der Übergänge ist hier eine Relation

$$\delta \subseteq Q \times \Sigma \times Q$$

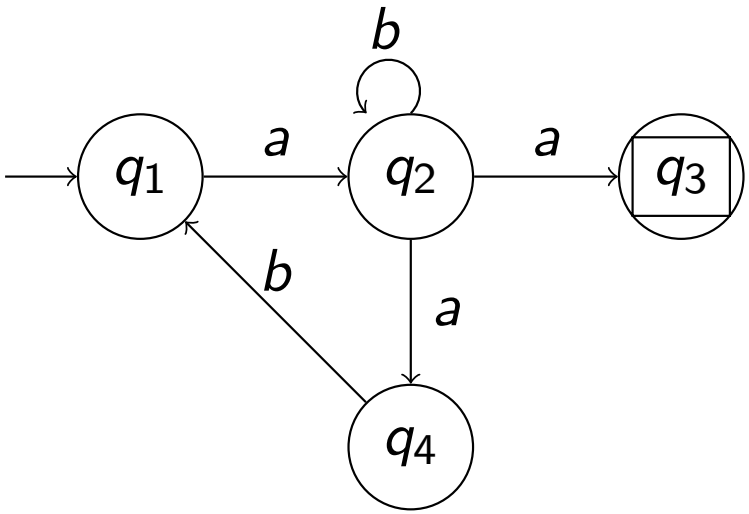
Aufgabe 1



a)

Gib die formale Beschreibung des Graphen an.

Aufgabe 1



a)

Gib die formale Beschreibung des Graphen an.

$A = (\Sigma, Q, q_1, F, P)$,

$\Sigma = \{a, b\}, F = \{q_3\}$,

$P = \{$

$q_1 a \rightarrow q_2$,

$q_2 a \rightarrow q_3$,

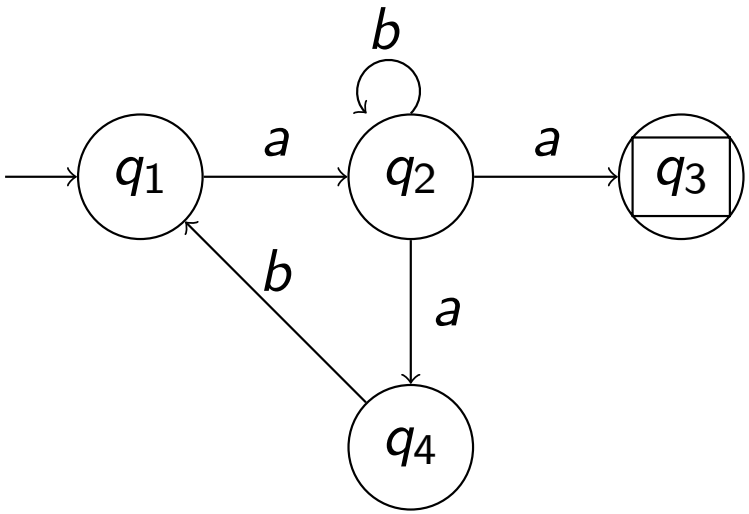
$q_2 a \rightarrow q_4$,

$q_2 b \rightarrow q_2$,

$q_4 b \rightarrow q_1$

$\}$

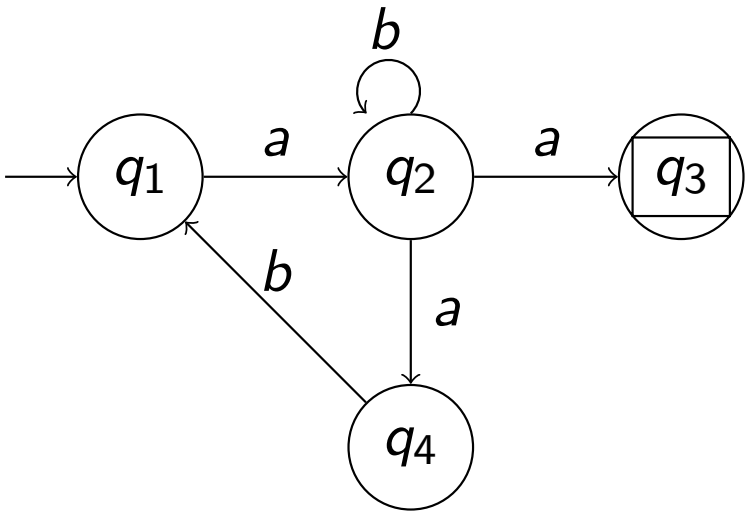
Aufgabe 1



b)

Handelt es sich bei dem Automaten um einen Mealy- oder einen Moore-Automaten?

Aufgabe 1

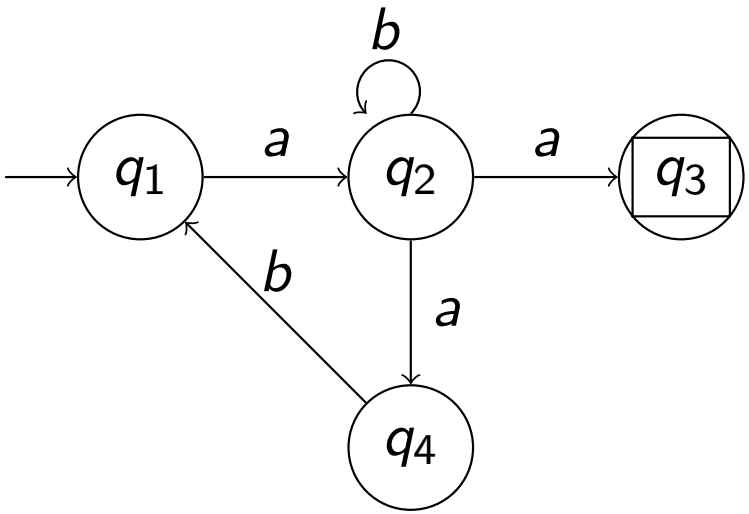


b)

Handelt es sich bei dem Automaten um einen Mealy- oder einen Moore-Automaten?

Es handelt sich um den häufigsten Spezialfall eines Moore-Automaten, den Akzeptor, da keine Ausgaben bei den Zustandsübergängen (Mealy) vermerkt sind. Die Ausgabe des Automaten erfolgt dabei nicht bei allen Zuständen sondern nur bei den Endzuständen.

Aufgabe 1



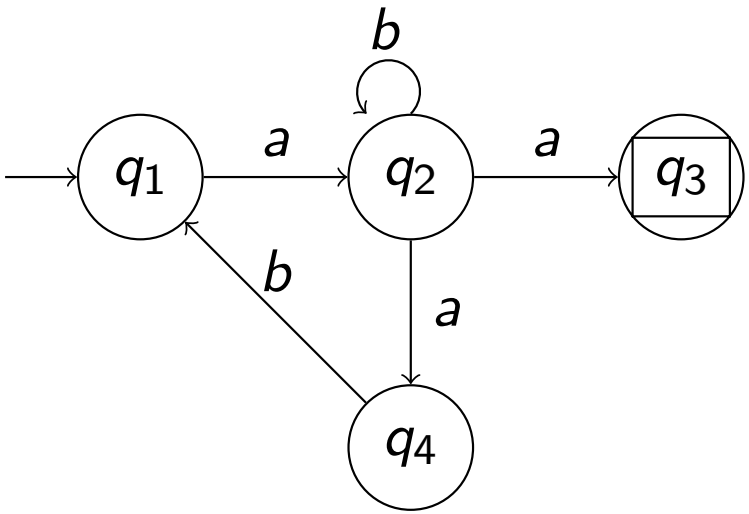
c)

Ist der Automat vollständig?

d)

Ist der Automat deterministisch?

Aufgabe 1



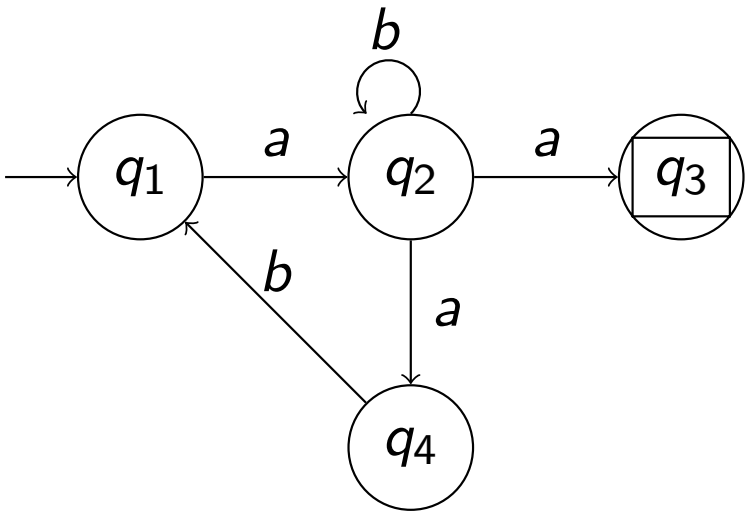
c)

Ist der Automat vollständig?
Nein, beispielsweise fehlt im Zustand q_4 ein Verhalten für die Eingabe a.

d)

Ist der Automat deterministisch?

Aufgabe 1



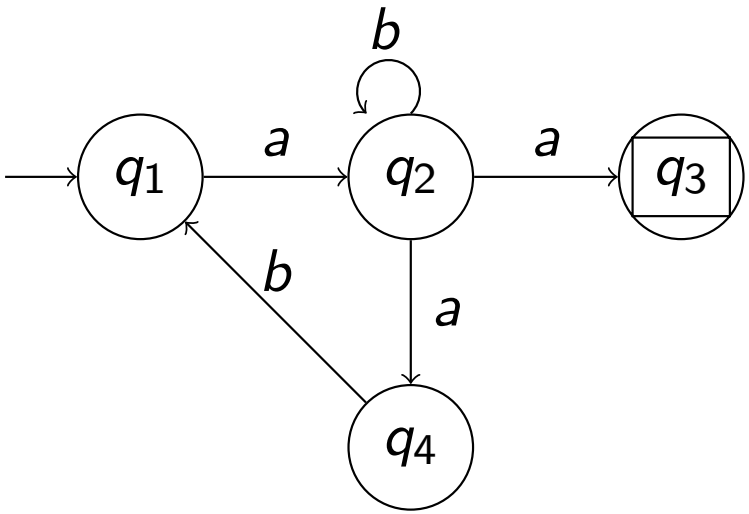
c)

Ist der Automat vollständig?
Nein, beispielsweise fehlt im Zustand q_4 ein Verhalten für die Eingabe a.

d)

Ist der Automat deterministisch?
Nein, im Zustand q_2 ist das Verhalten bei der Eingabe von a nicht eindeutig.

Aufgabe 1



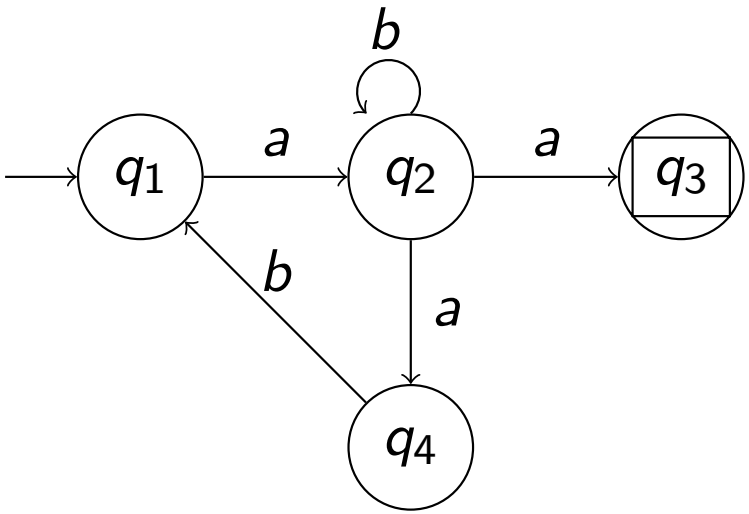
e)

Welche Sprache akzeptiert der Automat?

f)

Welchen Chomsky-Typ hat die Sprache?

Aufgabe 1



e)

Welche Sprache akzeptiert der Automat?

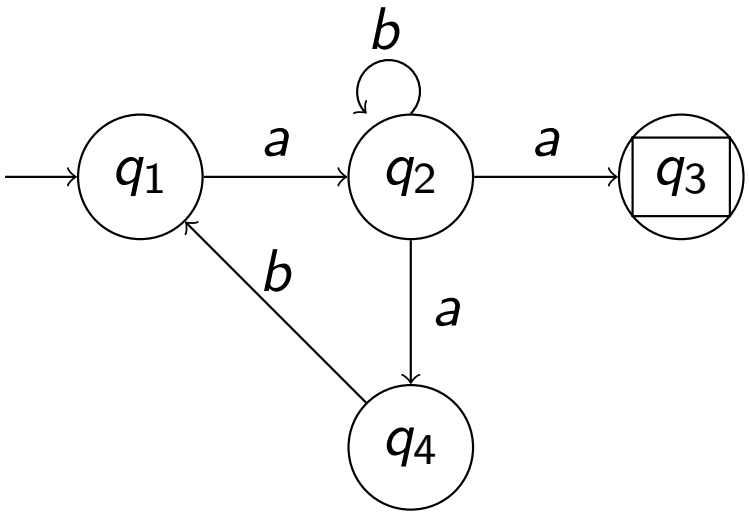
$L = \{a\beta^n a | n \in \mathbb{N}_0, \beta \in \{b, aba\}\}$

(Regulärer Ausdruck: $a(b + aba)^* a$)

f)

Welchen Chomsky-Typ hat die Sprache?

Aufgabe 1



e)

Welche Sprache akzeptiert der Automat?

$L = \{a\beta^n a | n \in \mathbb{N}_0, \beta \in \{b, aba\}\}$

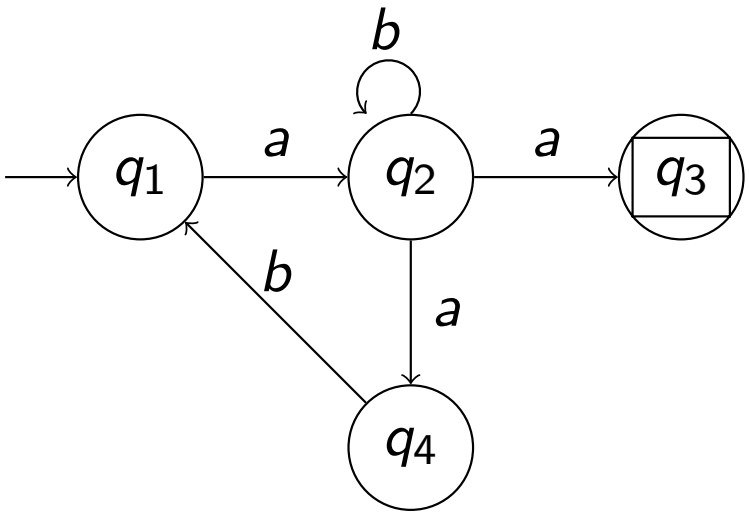
(Regulärer Ausdruck:
$a(b + aba)^* a$)

f)

Welchen Chomsky-Typ hat die Sprache?

CH3, da sie von einem endlichen Automaten akzeptiert wird.

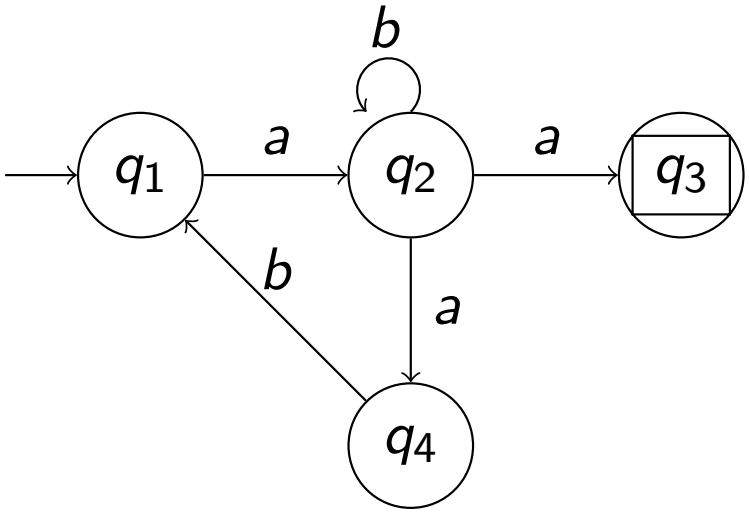
Aufgabe 1



g)

Geben Sie die Grammatik zu der akzeptierten Sprache an.

Aufgabe 1



g)

Geben Sie die Grammatik zu der akzeptierten Sprache an.

$G = (\Sigma, N, P, A)$,
$\Sigma = \{a, b\}, N = \{A, B, C, D\}$
$P = \{$
$A \rightarrow aB$,
$B \rightarrow bB|aC|a$,
$C \rightarrow bD$,
$D \rightarrow aB$
$\}$

Aufgabe 2

Akzeptor gesucht

Gib den Graphen des Akzeptors für die durch G erzeugte Sprache an.

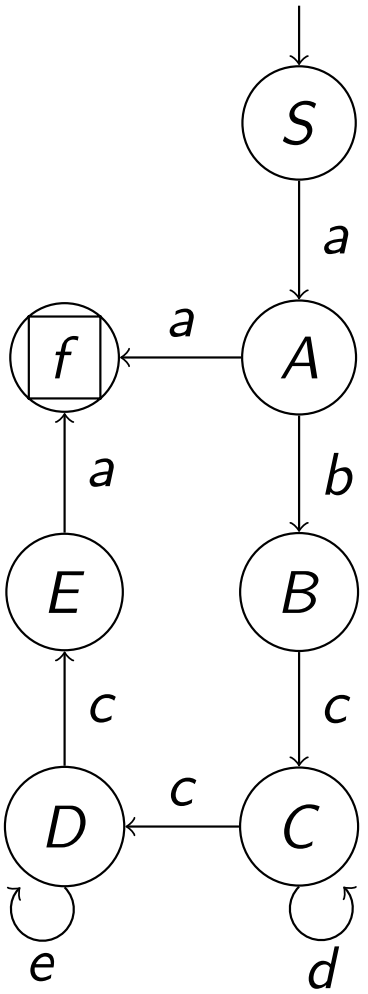
$G = (\Sigma, N, P, S)$,
$\Sigma = \{a, b, c, d, e\}$,
$N = \{S, A, B, C, D, E\}$
$P = \{$
$S \rightarrow aA$,
$A \rightarrow bB|a$,
$B \rightarrow cC$,
$C \rightarrow dC|cD$,
$D \rightarrow eD|cE$,
$E \rightarrow a$
$\}$

Aufgabe 2

Akzeptor gesucht

Gib den Graphen des Akzeptors für die durch G erzeugte Sprache an.

$G = (\Sigma, N, P, S)$,
$\Sigma = \{a, b, c, d, e\}$,
$N = \{S, A, B, C, D, E\}$
$P = \{$
$S \rightarrow aA$,
$A \rightarrow bB|a$,
$B \rightarrow cC$,
$C \rightarrow dC|cD$,
$D \rightarrow eD|cE$,
$E \rightarrow a$
$\}$

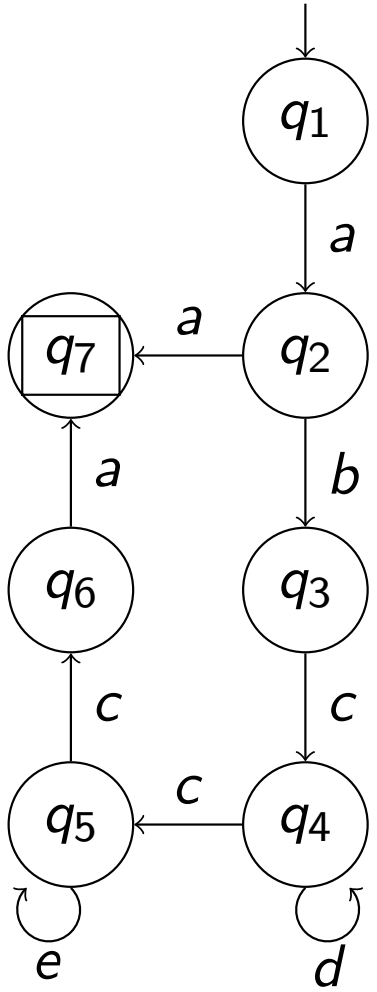


Aufgabe 2

Akzeptor gesucht

Gib den Graphen des Akzeptors für die durch G erzeugte Sprache an.

$G = (\Sigma, N, P, S)$,
$\Sigma = \{a, b, c, d, e\}$,
$N = \{S, A, B, C, D, E\}$
$P = \{$
$S \rightarrow aA$,
$A \rightarrow bB|a$,
$B \rightarrow cC$,
$C \rightarrow dC|cD$,
$D \rightarrow eD|cE$,
$E \rightarrow a$
$\}$



Definition

Reguläre Ausdrücke sind eine verbreitete und geeignete Notation, um reguläre Sprachen (CH-3) zu beschreiben. Ein regulärer Ausdruck über einem Zeichenvorrat C ist wird rekursiv definiert:

1. Für jedes $c \in C$ ist c ein regulärer Ausdruck.
2. ε ist ein regulärer Ausdruck.
3. Kein Zeichen („()“) ist auch ein regulärer Ausdruck, und beschreibt die leere Sprache.
4. Ist R ein regulärer Ausdruck, dann auch $(R)^*$.
5. Sind R und S reguläre Ausdrücke, so sind auch (RS) und $(R + S)$ reguläre Ausdrücke.
6. Klammern darf man ggf. weglassen:
   * bindet stärker als Verkettung bindet stärker als $+$

Bespiele für reguläre Ausdrücke

Beispiele

Bezeichner	$b(b+z)^*$
Ganze Zahlen	zz^*
Dezimalbrüche	zz^*/zz^*

Reguläre Aufgabe

Gegeben ist folgende Klasse von Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$:

$L_n = \{$Das Wort enthält genau einmal
eine Folge von a der Länge n, die nicht Teil
einer Folge von a mit einer Länge $> n$ ist$\}$

Gebt a) einen regulären Ausdruck für L_4 sowie b) einen vollständigen Akzeptor für L_3 an!

Reguläre Aufgabe

Gegeben ist folgende Klasse von Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$:

$$L_n = \{\text{Das Wort enthält genau einmal eine Folge von } a \text{ der Länge } n, \text{ die nicht Teil einer Folge von } a \text{ mit einer Länge} > n \text{ ist}\}$$

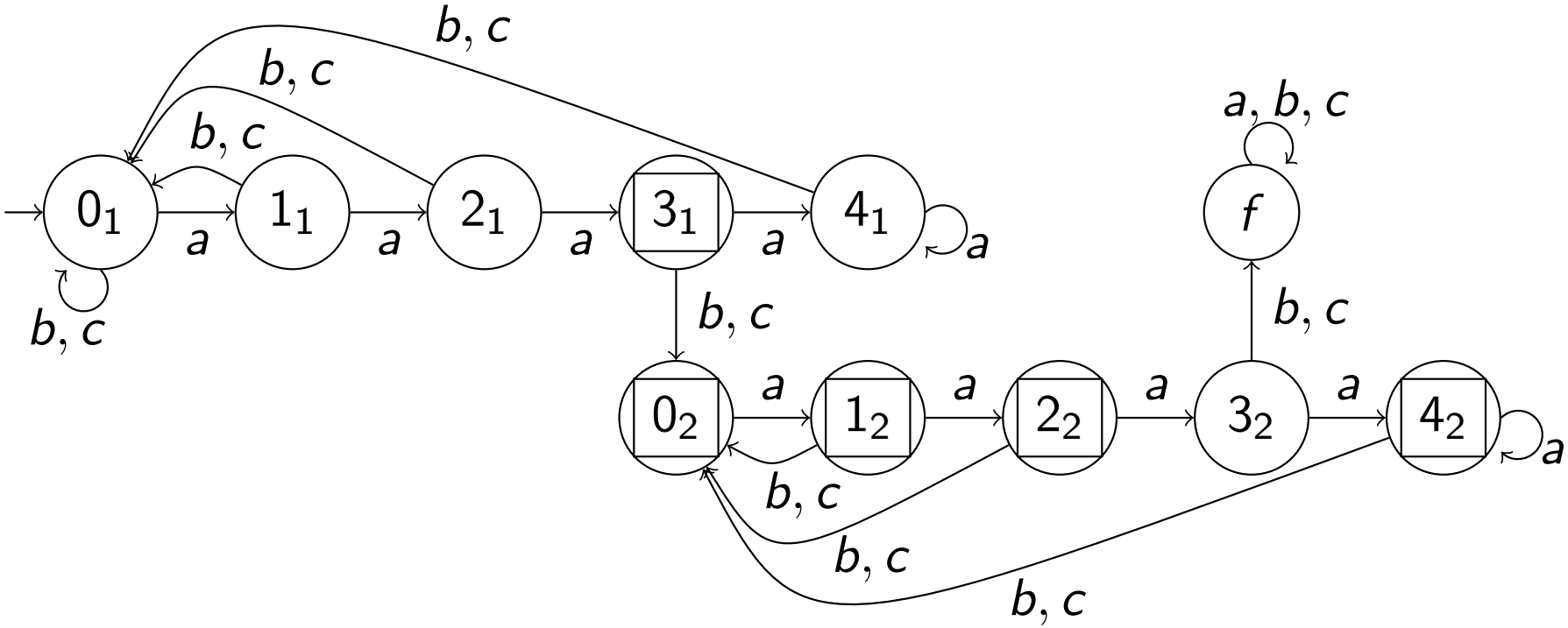
Gebt a) einen regulären Ausdruck für L_4 sowie b) einen vollständigen Akzeptor für L_3 an!

Lösung a)

Wir stellen sicher, dass die vier a genau einmal vorkommen, und sonst nur 1,2,3 oder mehr als 4 am Stück.

$$L_4 = ((b+c)^*(a+aa+aaa+aaaaaa^*)(b+c)(b+c)^*)^* \\ aaaa((b+c)(b+c)^*(a+aa+aaa+aaaaaa^*)(b+c)^*)^*$$

Lösung b)


b, c
b, c
b, c
a, b, c
0_1
1_1
2_1
3_1
4_1
f
a
a
a
a
a
b, c
b, c
b, c
0_2
1_2
2_2
3_2
4_2
a
a
a
a
a
b, c
b, c
b, c

Regulär Einkaufen

Gebt einen regulären Ausdruck an, der das Einkaufen in einem Supermarkt beschreibt. Beschreibt den Einkauf mit diesen Zeichen:

- b Laden betreten
- v Laden verlassen
- s Einkaufswagen verschieben
- p Produkt in den Einkaufswagen legen
- z Einkäufe bezahlen

Den Einkaufswagen erhält man am Eingang beim Betreten des Ladens und gibt ihn beim Verlassen am Ausgang zurück. Die Produkte sind im ganzen Laden verteilt und nicht in Reichweite des Ein- oder Ausgangs – aber an der Kasse gibt es Süßes! Beachtet, dass nichts zu bezahlen ist, wenn keine Waren im Einkaufswagen liegt (genau dann, ihr Schelme!).
Zielloses Rumstöbern ist erlaubt!

Einkaufslösung

Eine mögliche Lösung ist:

$$bss^*(\varepsilon + p(p+s)^*zs)v$$

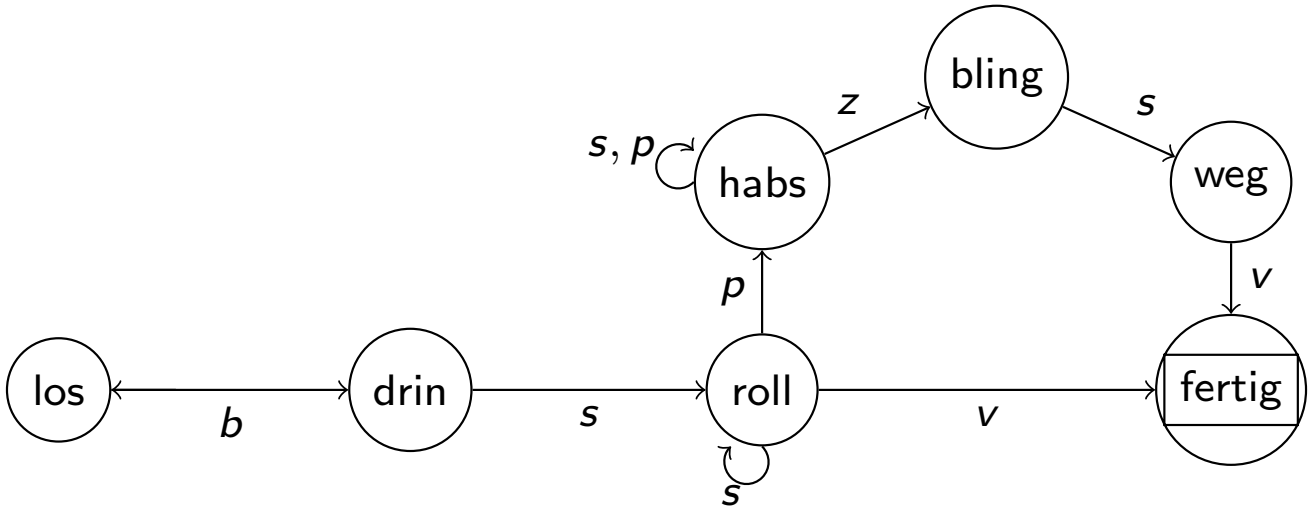
Und wie immer:
Wie sieht der Akzeptor dazu aus (muss nicht vollständig sein)?

Einkaufslösung

Eine mögliche Lösung ist:

$$bss^*(\varepsilon + p(p+s)^*zs)v$$

Und wie immer:
Wie sieht der Akzeptor dazu aus (muss nicht vollständig sein)?

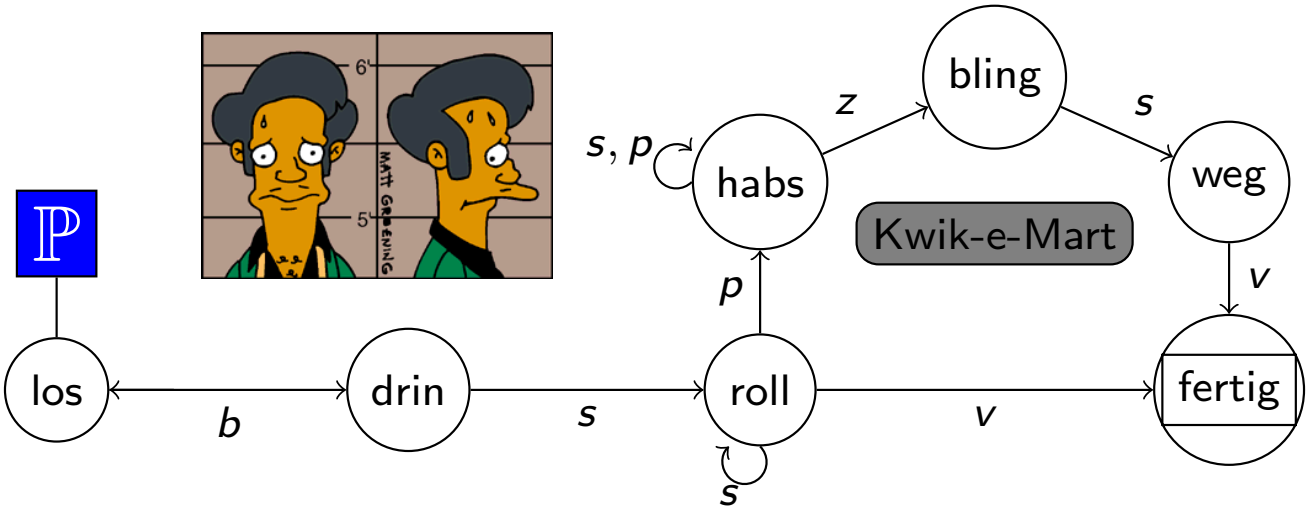


Einkaufslösung

Eine mögliche Lösung ist:

$$bss^*(\varepsilon + p(p+s)^*zs)v$$

Und wie immer:
Wie sieht der Akzeptor dazu aus (muss nicht vollständig sein)?



Klasse

Was ist das?

Eine Klasse ist eine Art Prototyp, so etwas wie eine Bauanleitung, für die Objekte, die man später von dieser Klasse ableitet.
Jede Klasse kann folgende Dinge haben:

Konstruktor Das ist eine Funktion die aus dem Prototyp das Objekt zusammenbau. Es gibt der Klasse also nochmal einen Feinschliff und gibt dann das fertige Objekt an den Aufrufer zurück. Wird kein Konstruktor angegeben, so erstellt Java einen.

Klasse (Fortsetzung)

Was ist das? (Fortsetzung)

Variable Jede Klasse kann Variablen haben, die entweder von allen Objekten gleichzeitig genutzt werden (→ `static`) oder jedem Objekt selbst gehören (default).

Methoden Jede Klasse kann auch Funktionen besitzen, die normalerweise mit dieser Klasse direkt arbeiten und logisch mit ihr in Verbindung stehen. Diese heißen dann Methoden. Auch hier kann man zwischen Funktionen der Klasse und der Objekte unterscheiden (→ `static`).

Objekt

Was ist das?

Ein Objekt ist ein Abbild einer Klasse. Von ihnen kann es viele geben und sie werden dann auch Instanzen der Klasse, von der sie stammen, genannt.

Im Code einer Methode kann man auf das Objekt, mit dem aufgerufen wurde, mit der Variable `this` zugreifen. Diese Variable wird von Java bereitgestellt.

Um von außerhalb eine Methode oder eine Variable eines Objektes oder eine Klasse zuzugreifen, benutzt man den „.“.

Ein Beispiel für einen Methodenaufruf ist: `phone.dial(...);`

Es gehört in Java zum guten Stil, nicht direkt auf die Variablen eines Objektes zuzugreifen. Damit man sie trozdem verändern kann, stellt man meist get- und set-Methoden bereit.

Objekt

Was ist das?

Ein Objekt ist ein Abbild einer Klasse. Von ihnen kann es viele geben und sie werden dann auch Instanzen der Klasse, von der sie stammen, genannt.
Im Code einer Methode kann man auf das Objekt, mit dem aufgerufen wurde, mit der Variable `this` zugreifen. Diese Variable wird von Java bereitgestellt.
Um von außerhalb eine Methode oder eine Variable eines Objektes oder eine Klasse zuzugreifen, benutzt man den „.“.
Ein Beispiel für einen Methodenaufruf ist: `phone.dial(...);`
Es gehört in Java zum guten Stil, nicht direkt auf die Variablen eines Objektes zuzugreifen. Damit man sie trozdem verändern kann, stellt man meist get- und set-Methoden bereit.

Objekt

Was ist das?

Ein Objekt ist ein Abbild einer Klasse. Von ihnen kann es viele geben und sie werden dann auch Instanzen der Klasse, von der sie stammen, genannt.
Im Code einer Methode kann man auf das Objekt, mit dem aufgerufen wurde, mit der Variable `this` zugreifen. Diese Variable wird von `Java` bereitgestellt.
Um von außerhalb eine Methode oder eine Variable eines Objektes oder eine Klasse zuzugreifen, benutzt man den „`.`“.
Ein Beispiel für einen Methodenaufruf ist: `phone.dial(...);`
Es gehört in Java zum guten Stil, nicht direkt auf die Variablen eines Objektes zuzugreifen. Damit man sie trozdem verändern kann, stellt man meist `get`- und `set`-Methoden bereit.

Objekt

Was ist das?

Ein Objekt ist ein Abbild einer Klasse. Von ihnen kann es viele geben und sie werden dann auch Instanzen der Klasse, von der sie stammen, genannt.
Im Code einer Methode kann man auf das Objekt, mit dem aufgerufen wurde, mit der Variable `this` zugreifen. Diese Variable wird von Java bereitgestellt.
Um von außerhalb eine Methode oder eine Variable eines Objektes oder eine Klasse zuzugreifen, benutzt man den „.“.
Ein Beispiel für einen Methodenaufruf ist: `phone.dial(...);`
Es gehört in Java zum guten Stil, nicht direkt auf die Variablen eines Objektes zuzugreifen. Damit man sie trozdem verändern kann, stellt man meist get- und set-Methoden bereit.

Aufgabe 3

Viele, viele Stifte...

a) Erstelle eine Java-Klasse für die Stifte in deinem Federmäpchen. Wir nehmen an du hättest nur Buntstifte, sie haben eine Größe, Farbe und können malen und zerbrochen werden.

b) Desweiteren soll noch ein Hauptprogramm geschrieben werden, dass ein Federmäpchen zufällig mit 10 verschiedenen Stiften füllt, einen bricht und damit malt.

Lösung a)

```
class Crayon {
  // my colour and length. not static, thus per object
  double red, blue, green, length;
  // abrasion per drawin. static, as shared by all objects
  static double abrasion;

  // my constructor
  public Crayon(double length, double red, double ↵
      blue, double green){
    this.length = length;
    this.red = red;
    this.blue = blue;
    this.green = green;
  }

...
```

Lösung a) (Fortsetzung)

```
    public void draw() {
        // only draw if enogh material is left
        if (this.length >= this.abrasion) {
            // here be code for drawing around
            this.length -= this.abrasion;
        }
    }

    // break the crayon in half (beware: break is a java keyword!)
    public Crayon snap() {
        Crayon secondHalf = new Crayon(this.length/2, ↵
            this.red, this.blue, this.green);
        this.length /= 2;
        return secondHalf;
    }

    // insert getLength() and setLength()-Methods
    // set-Methods make no sense here.
}
```

Lösung b)

```
class ManyCrayons {
  public static void main(String[] arg) {
    int i;
    // declare all the funny crayons
    Crayon[] crayons = new Crayon[10];
    Crayon halfling;

    // create the crayons with random length ([0,10)) and random colour
    for(i=0; i<crayons.length; i++) {
      crayons[i] = new Crayon(Math.random()*10, ↵
          Math.random(), Math.random(), Math.random());
    }

    // break the second crayon
    halfling = crayons[1].snap();
    // and draw with it
    halfling.draw();
  }
}
```