

# Theoretische Informatik – Schreibomat

Florian

September 11, 2012

## Tools

- Finite State Machine Designer: <http://madebyevan.com/fsm/>

## 1 Was ist Informatik?

- Konkrete<sup>1</sup> Konzepte: Algorithmus, Berechnung, Komplexität
- Was will die theoretische Informatik? Formale Bestimmung der Begriffe, unabhängig von Hard- und Software
- Ziel: Grundlegende Eigenschaften von Algorithmen, Berechnungen erkennen. Methoden entwickeln zum Entwurf beweisbar korrekter Hard- und Software (nicht Schwerpunkt dieser Vorlesung)
- Grenzen der automatischen Berechnung aufzeigen
- Typische Fragestellungen: Ist es möglich, ein Programm zu schreiben, das ein anderes Programm als Eingabe erhält und feststellen kann, ob dieses in eine Endlosschleife gerät oder nicht?  $\Rightarrow$  Halteproblem. Nein, es ist natürlich nicht möglich. Eine andere typische Fragestellung ist die Folgende: Wir haben einen Rucksack, und der hat 30 Liter Fassungsvermögen. Und wir haben noch 50 Gegenstände, und jetzt wollen wir wissen: wie lange dauert es, herauszufinden, wieviele Gegenstände in den Rucksack passen? Rucksackproblem, nicht in polynomialer Zeit lösbar  $\Rightarrow$  Es dauert exponentiell lange.
- Leider ist es in der Praxis so, dass die Antwort auf ähnliche Fragestellungen nicht immer so offensichtlich ist, und deshalb müssen wir es uns ein bisschen genauer betrachten.

**Übung.** Gegeben sei das folgende Strassennetz:  
(Wildes Gekritzelt)

---

<sup>1</sup>4. September 2012

- (a) Gibt es eine Möglichkeit, alle Strassen genau 1x zu durchlaufen und dann wieder am Ausgangspunkt anzulangen,
- (b) Gibt es eine Möglichkeit, alle Kreuzungen genau 1x zu passieren und dann wieder am Ausgangspunkt anzulangen?

Die Antworten sind imfall:

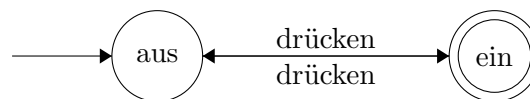
- (a) Einfache Aufgabe: Geht immer, wenn die Anzahl der Abzweigungen an jeder Kreuzung gerade ist.
- (b) Traveling salesman (TSP). Schwierig! Es ist keine wesentlich bessere Methode bekannt, als einfach alles durchzuprobieren.

Ziel für die Vorlesung: Um solche Fragestellungen systematisch beantworten zu können, brauchen wir ein exaktes mathematisches Modell eines Computers.

## 2 Automatentheorie

Endliche Automaten, Kontextfreie Grammatiken und Keller-Automaten. Zusätzliche Motivation: Das sind Konzepte, denen man auch häufig in der Praxis begegnet (Compilerbau, Textsuche, Textverarbeitung)

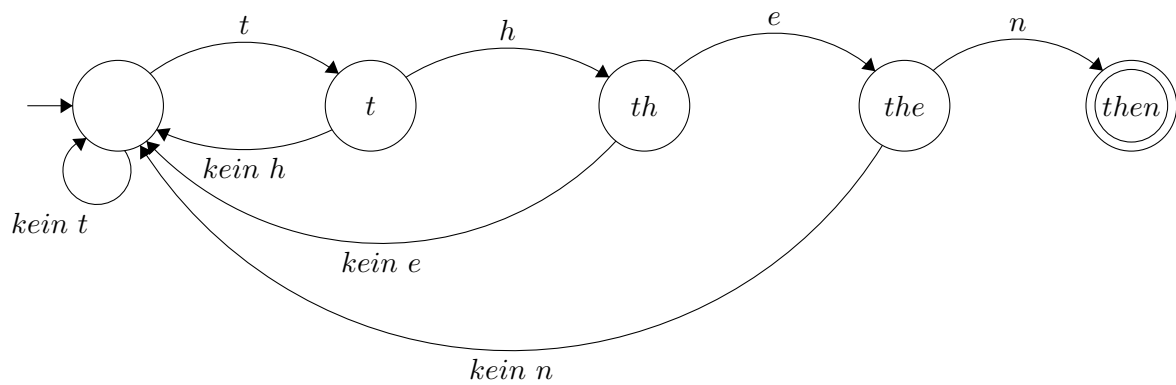
Noch nicht ganz so formal, bisschen intuitiv. Modellierung eines Kippschalters.



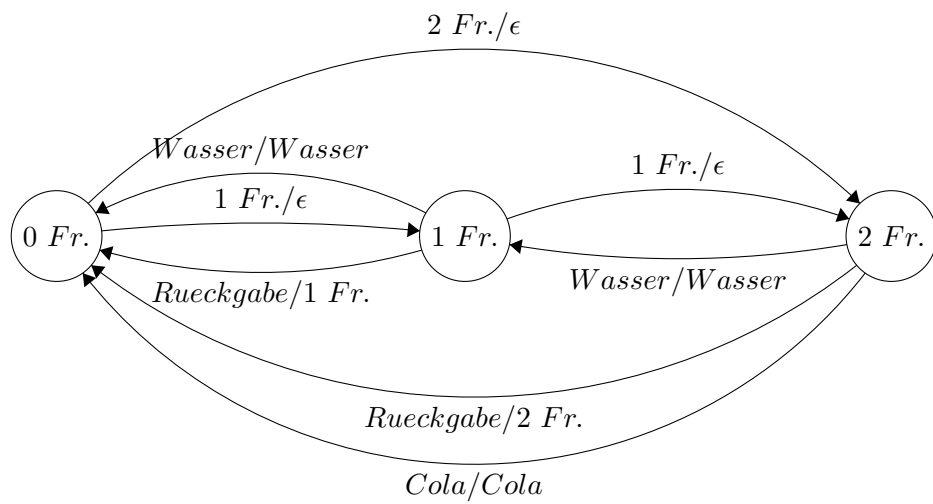
**Definition 1** (Endlicher Automat). Wir haben:

- Zwei Zustände, davon ein Startzustand
- und ein akzeptierender Zustand
- Transitionen (Zustandsübergänge): führen den Automaten anhand einer Eingabe von einem Zustand in den nächsten.

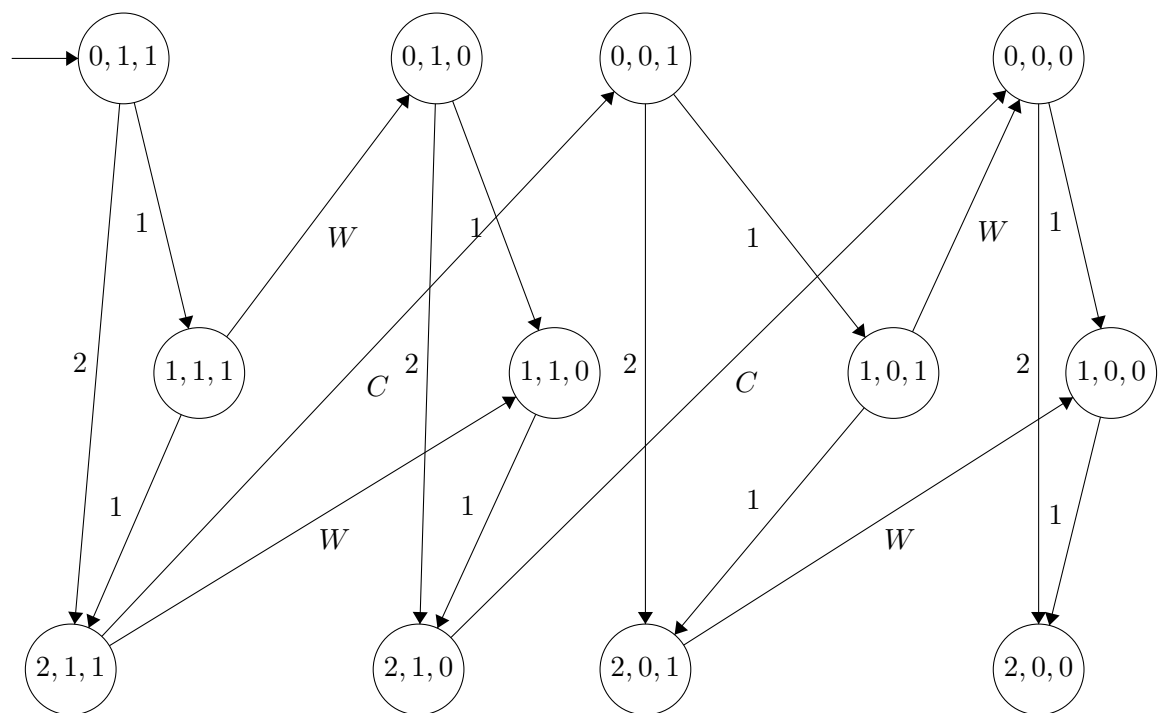
Beispiel: Mustererkennung in Texten: z.B. Suche das Wort "then"



Getränkeautomat (Beispiel für Automat mit Ausgabe): Cola für CHF 2, Wasser für CHF 1. Münzannahme: Münzen zu 1, 2 CHF.



Wenn der Automat nur eine Flasche Cola und eine Flasche Wasser enthält:



### 3 Formale Sprachen

Ziel: Genaue Beschreibung der Ein- und Ausgaben eines Automaten.

**Definition 2** (Alphabet). (endliche, nichtleere Menge von Symbolen). Bsp: Binäres Alphabet  $\Sigma_{\text{bin}} = \{0, 1\}$ , Tastaturalphabet  $\Sigma_{\text{tast}} = \{a, \dots, z, A, \dots, Z, 0, \dots, 9, \dots\}$

**Definition 3** (Wort (= Zeichenkette, String)). Endliche Folge von Symbolen eines gegebenen Alphabets. Beispiel: 01110 ist ein Wort über  $\Sigma_{\text{bin}}$ .

**Bemerkung** (Eigenschaften von Wörtern). Die da wären:

- Leeres Wort:  $\epsilon$  (manchmal  $\lambda$ ) = leere Folge von Symbolen (über einem beliebigen Alphabet)
- Länge eines Wortes:  $|w|$  bezeichnet die Anzahl der Symbole im Wort  $w$ .  $|\epsilon| = 0$ .

**Bemerkung** (Konventionen für Darstellung von Wörtern). Wir sagen:

- $a, b, c, \dots$  für Buchstaben, Symbole
- $v, w, x, \dots$  für Wörter

**Definition 4** (Potenzen von Wörtern). Es gibt im Angebot:

- Menge aller Wörter einer bestimmten Länge:

Sei  $\Sigma$  Alphabet, so:

$$\Sigma^0 = \{\epsilon\}$$

$$\Sigma^1 = \Sigma$$

$$\Sigma^2 = \{ab | a, b \in \Sigma\}$$

$$\Sigma^i = \{a_1 \dots a_i | a_1 \dots a_i \in \Sigma\}$$

- Menge aller Wörter über  $\Sigma$ :

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i, \epsilon \in \Sigma^*$$

- Menge aller nichtleeren Wörter über  $\Sigma$ :

$$\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i, \epsilon \notin \Sigma^+$$

### Beispiel.

Beispiel:

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

$$\Sigma^2 = \{00, 01, 10, 11\}$$

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$$

**Definition 5** (Konkatenation (Verkettung) von Wörtern).

$$v = a_1 \dots a_k, w = b_1 \dots b_k \text{ über } \Sigma \Rightarrow v \cdot w = a_1 \dots a_k b_1 \dots b_k = vw$$

**Bemerkung** (Rechenregeln für Konkatenation von Wörtern).

$$v \cdot (v \cdot w) = (u \cdot v) \cdot w$$

$$|x \cdot y| = |x| + |y|$$

$$x \cdot \epsilon = \epsilon \cdot x = x$$

**Bemerkung** (Infixe, Präfixe, Suffixe). Seien  $v, w \in \Sigma^*$  für ein Alphabet  $\Sigma$ , dann ist  $v$  ein Teilwort (Infix) von  $w$ , falls es  $x, y \in \Sigma^*$  gibt, so dass  $w = x \cdot v \cdot y$ ;  $v$  ist ein Präfix von  $w$ , falls es  $y \in \Sigma^*$  gibt, so dass  $w = v \cdot y$ . Suffix funktioniert analog.

Beispiel:  $abc$  ist Teilwort von  $aabcc$ ,  $aa$  ist Präfix und Suffix von  $aabcaa$ .  $\epsilon$  ist Teilwort von jedem Wort.

**Definition 6** (Sprache).  $L$  über Alphabet  $\Sigma$  ist eine Teilmenge von  $\Sigma^*$ ,  $L \subseteq \Sigma^*$ . Sprache ist Menge von Wörtern, kann unendlich gross sein. Leere Sprache:  $\emptyset$  enthält keine Wörter, ist über jedem Alphabet definiert. Spezielle Sprache:  $L_\epsilon$  ist die Sprache, die nur das leere Wort enthält.  $L_\epsilon \neq \emptyset$ .

**Definition 7** (Konkatenation von Sprachen).  $L_1, L_2 \subseteq \Sigma^* \Rightarrow L_1 \cdot L_2 = L_1 L_2 = \{v \cdot w | v \in L_1, w \in L_2\}$

**Definition 8** (Potenzen von Sprachen).

$$\begin{aligned} L^0 &= L_\epsilon = \{\epsilon\} \\ L^{i+1} &= L^i \cdot L \text{ für } i \in \mathbb{N} \end{aligned}$$

**Definition 9** (Kleene-Stern).

$$\begin{aligned} L^* &= \bigcup_{i \in \mathbb{N}} L^i \\ L^+ &= \bigcup_{i \in \mathbb{N}} L^i - L_\epsilon \end{aligned}$$

**Beispiel.** Sei  $\Sigma = \{a, b\}, L_1 = \{a^i | i \geq 0\} = \{\epsilon, a, aa, aaa, \dots\}, L_2 = \{b^i | i \geq 0\} = \{\epsilon, b, bb, bbb, \dots\}$ :

$L_1 \cdot L_2 = \{\epsilon, a, b, ab, aab, abb, aaab, \dots\} = \{a^i b^j | i \geq 0, j \geq 0\}$  ist die Menge aller Wörter über  $\{a, b\}$ , in dem alle  $a$ s vor allen  $b$ s vorkommen.

**Übung.** Seien  $L_1, L_2, L_3 \subseteq \Sigma^*$ .

- (a) Gilt  $(L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$ ? Ja.
- (b) Gilt  $(L_1 \cup L_2) \cdot L_3 = L_1 \cdot L_3 \cup L_2 \cdot L_3$ ? Ja.
- (c) Gilt  $(L_1 \cdot L_2) \cup L_3 = (L_1 \cup L_3) \cdot (L_2 \cup L_3)$ ? Quatsch.
- (d) Gilt  $L_1 \cdot L_2 = L_2 \cdot L_1$ ? Quatsch.

**Beweis.** (a)

$$\begin{aligned} (L_1 \cdot L_2) \cdot L_3 &= \{vw | v \in L_1 \cdot L_2, w \in L_3\} \\ &= \{xyz | x \in L_1, y \in L_2, z \in L_3\} \\ &= \{xz | x \in L_1, z \in L_2 L_3\} \\ &= L_1 \cdot (L_2 \cdot L_3) \end{aligned}$$

(b)

$$\begin{aligned} (L_1 \cup L_2) \cdot L_3 &= \{vw | v \in L_1 \cup L_2, w \in L_3\} \\ &= \{vw | v \in L_1, w \in L_3\} \cup \{vw | v \in L_2, w \in L_3\} \\ &= (L_1 \cdot L_3) \cup (L_2 \cdot L_3) \end{aligned}$$

(c) Gegenbeispiel:  $L_1 = L_2 = \{a\}, L_3 = \{b\}$ . Daraus folgt:

$$\begin{aligned} L_1 L_2 &= \{aa\} \Rightarrow (L_1 L_2) \cup L_3 = \{aa, b\} \\ L_1 \cup L_3 &= \{a, b\} = L_2 \cup L_3 \Rightarrow \dots \end{aligned}$$

**Bemerkung.** (d) gilt aber für  $|\Sigma| = 1$ .

**Definition 10** (Entscheidungsproblem). Die Eingabe ist eine Sprache  $L$  über einem Alphabet  $\Sigma$  sowie ein Wort  $w \in \Sigma^*$ . Die Ausgabe soll lauten: JA falls  $w \in L$  oder NEIN falls  $w \notin L$ .

Die Modellierung von vielen alltäglichen Berechnungsproblemen im Formalismus der formalen Sprachen.

**Beispiel** (Primzahltest). Das Alphabet  $\Sigma = \{0, 1\}$ . Die Sprache

$$L = \{w \in \Sigma^* \mid w \text{ ist Binärdarstellung einer Primzahl}\}$$

Eine Zahl  $p \in \mathbb{N}$  ist Primzahl genau dann, wenn  $\text{Bin}(p) \in L$ .

Identifizierung von Sprachen als Probleme.

## 4 Kurze Einführung in formale Beweise

Behauptungen<sup>2</sup> mit Unanfechtbarkeitsanspruch wollen bewiesen werden. Insbesondere negative Aussagen der Form "Dieses Rechenmodell kann diese Aufgabe nicht lösen" brauchen eine präzise Formulierung und Begründung, um glaubwürdig zu sein<sup>3</sup>.

Als Beweismethoden haben wir im Angebot:

Unkontrollierte  
Hausauf-  
gabe

**Deduktion (zum Beweis einer Implikation)** Zum Beweis von Wenn-Dann-Aussagen,

z.B.

$$\text{Wenn } \underbrace{x \geq 4}_{\text{Hypothese}}, \text{ dann } \underbrace{x^2 \geq 16}_{\text{Konklusion}}$$

Vorgehen: Hypothese als wahr annehmen, dann Folgerungen darauf anwenden, bis die Konklusion folgt.

**Beispiel.** Sei  $x \geq 4$ . Es gilt  $4^2 = 16$  und die Quadratfunktion ist steigend. Daraus folgt:  $x^2 \geq 4^2 = 16$

Hierfür ist es oft hilfreich, Definitionen von Begriffen einzusetzen.

**Beispiel.** Wenn  $L = \{w \in \{a, b\}^* \mid |w| \text{ gerade}\}$ , dann gilt für alle  $x \in L^2$ , dass  $|x|$  gerade ist.

**Beweis.** Sei  $x \in L^2$ .

Definition von  $L^2$  einsetzen. Es existieren  $u, v \in L$ , so dass  $x = uv$ .

Definition von  $L$  einsetzen.  $|u|$  ist gerade,  $|v|$  ist gerade.

$\Rightarrow |x| = |u| + |v| \Rightarrow |x|$  ist gerade.

**Doppelte Deduktion (zum Beweis einer Äquivalenz)** Zerlegung in zwei Wenn-Dann-Aussagen, wird getrennt bewiesen.

<sup>2</sup>11. September 2012

<sup>3</sup>Hopcraft et al., Abschnitt 1.2–1.4

**Beispiel** (Gleichheit von Mengen).

$$A = B \Rightarrow A \subseteq B \wedge B \subseteq A \Rightarrow A = B$$

**Beispiel.** Sprachen sind ja auch Mengen. Also: seien  $L_1, L_2 \in \Sigma^*$ . Dann gilt:

$$L_1 \cdot (L_1 \cup L_2) = L_1^2 \cup L_1 \cdot L_2$$

**Beweis.** Zweimal:

- " $\subseteq$ ":

Sei  $x \in L_1 \cdot (L_1 \cup L_2) \Rightarrow \exists x = uv, u \in L_1, v \in L_1 \cup L_2$ .

Fallunterscheidung: (a)  $v \in L_1 \Rightarrow x = uv, u \in L_1, v \in L_1 \Rightarrow x \in L_1^2$ .

(b)  $v \in L_2 \Rightarrow x = uv, u \in L_1, v \in L_2 \Rightarrow x \in L_1 \cdot L_2$ .

- " $\supseteq$ ":

Sei  $x \in L_1^2 \cup L_1 \cdot L_2$ .

Fallunterscheidung: (a)  $x \in L_1^2 \Rightarrow x = uv, u, v \in L_1$

$\Rightarrow v \in L_1 \cup L_2$

$\Rightarrow x \in L_1 \cdot (L_1 \cup L_2)$ .

(b)  $x \in L_1 \cdot L_2 \Rightarrow x = uv, u \in L_1, v \in L_2$

$\Rightarrow v \in (L_1 \cup L_2)$

$\Rightarrow x \in L_1 \cdot (L_1 \cup L_2)$

**Widerspruchsbeweis** Es wird eine Annahme getroffen und dann solange gefolgert, bis Quatsch herauskommt. Daraus folgt, dass die Annahme auch falsch sein muss.

**Beispiel.** Seien  $a, b \in \mathbb{N}$  und  $a, b \geq 2$ . Dann gilt:  $a$  teilt nicht  $ab + 1$ .

**Beweis.** Seien  $a, b \in \mathbb{N}$ . Angenommen,  $a$  teilt  $ab + 1$ .

Ziel: Annahme zum Widerspruch führen, dann folgt daraus Negation, also die gewünschte Konklusion.

$a$  teilt  $ab \Rightarrow a$  teilt  $ab$  und  $ab + 1$ .

$\Rightarrow a$  teilt  $(ab + 1) - (ab) = 1$

$\Rightarrow a = 1$ . Bonk! Widerspruch zu  $a \geq 2$ .

**Beispiel.** Es gibt unendlich viele Primzahl (äquivalente Formulierung: Es gibt keine grösste Primzahl).

**Beweis.** Angenommen, es gäbe eine grösste Primzahl. Wir nennen die Anzahl der Primzahlen  $k$ .

Seien  $p_1 < p_2 < \dots < p_k$  die Primzahlen. Betrachten wir  $x = p_1 \cdot p_2 \cdot \dots \cdot p_k + 1$ . Da  $p_k$  die grösste Primzahl ist und  $x > p_k$ , kann  $x$  keine Primzahl sein.

Also gibt es eine Primfaktorzerlegung von  $x$ , und  $p_l$  sei die kleinste Primzahl in dieser Zerlegung.



Dann gilt:  $p_l$  teilt  $p_1 \cdot p_2 \cdot \dots \cdot p_k$  und  $p_l$  teilt darum auch  $x$ .

Daraus folgt:  $p_l$  teilt  $p_1 \cdot p_2 \cdot \dots \cdot p_k + 1$ . Aus dem vorhergehenden Beweis ( $a$  teilt  $ab + 1$ ) folgt, dass das ein Widerspruch ist.

**Induktion** Vor allem verwendet für Beweise über natürliche Zahlen. Ziel: Zeige, dass die Aussage  $S(n)$  für alle  $n \geq n_0$  gilt.

Methode:

1. Induktionsanfang (Anker). Zeige  $S(n_0)$ .
2. Induktionsschritt. Zeige  $S(i) \Rightarrow S(i + 1)$ .

Idee: Wenn  $S(n_0)$  gilt und  $S(i) \Rightarrow S(i + 1)$  für alle  $i \geq n_0$ , dann folgt aus  $S(n_0)$  und  $S(n_0) \Rightarrow S(n_0 + 1)$ , dann gilt  $S(n_0 + 1)$ . Undsoweiter, undsofort.

**Beispiel** (Der kleine Gauss). . Für alle  $n \geq 1$  gilt:  $\sum_{i=1}^n i = \frac{n(n+1)}{2}$

**Beweis.** Anker:  $\sum_{i=1}^1 = \frac{1+(1+1)}{2}$

Schritt: es gelte  $\sum_{i=1}^k i = \frac{k(k+1)}{2}$  (Induktionsvoraussetzung).