Theoretische Informatik – Schreibomat

Florian

September 5, 2012

Tools

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

1 Was ist Informatik?

- Konkrete[1] 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? ⇒ 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 ⇒ Es dauert expontiell lange.
- Leider ist es in der Praxis so, dass die Antwort auf ähnliche Fragestellungen nicht immer so offensichtlich ist, und deshalt müssen wir es uns ein bisschen genauer betrachten.

Übung. Gegeben sei das folgende Strassennetz:
(Wildes Gekritzel)

[1] 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.

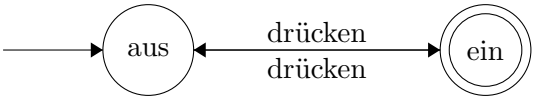
(b) Traveling salesman (TSP). Schwierig! Es ist keine wesentlich bessere Methode bekannt, als einfach alles durchzuprobieren.

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)

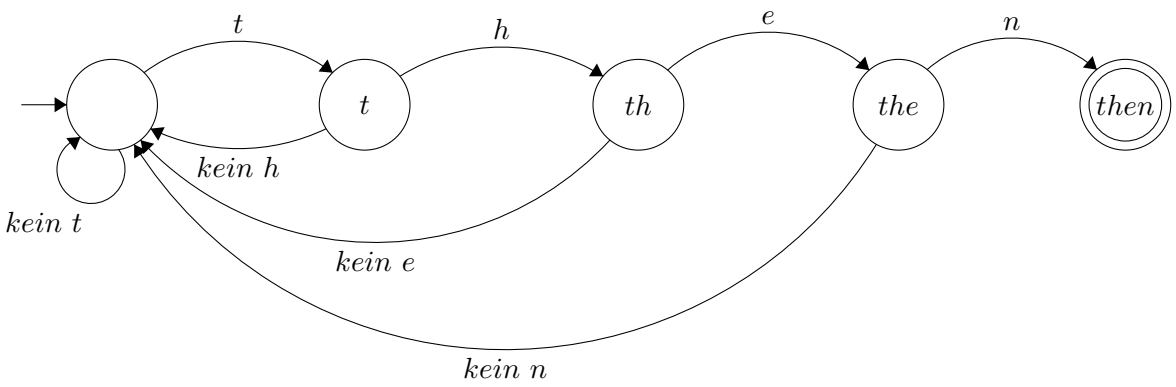
Noch nicht ganz so formal, bisschen intuitiv. Modellierung eines Kippschalters.



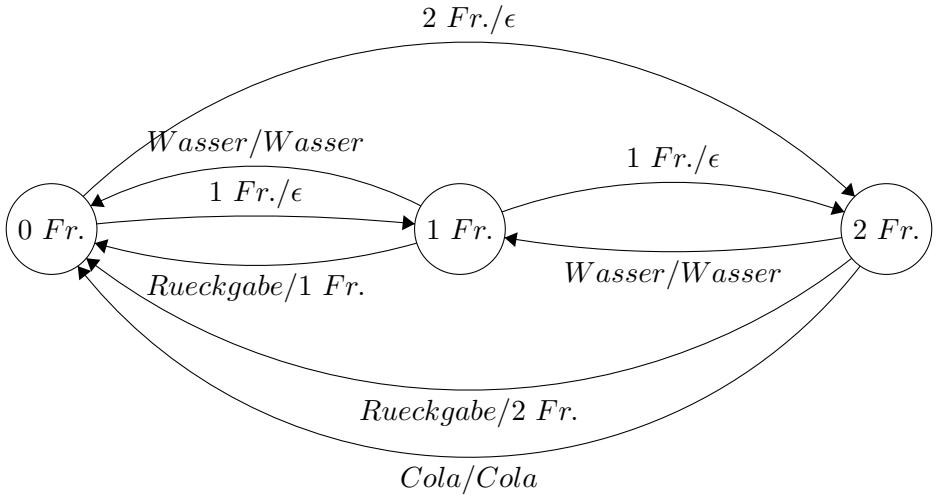
Definition 1 (Endlicher Automat)**.** Wir haben:

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

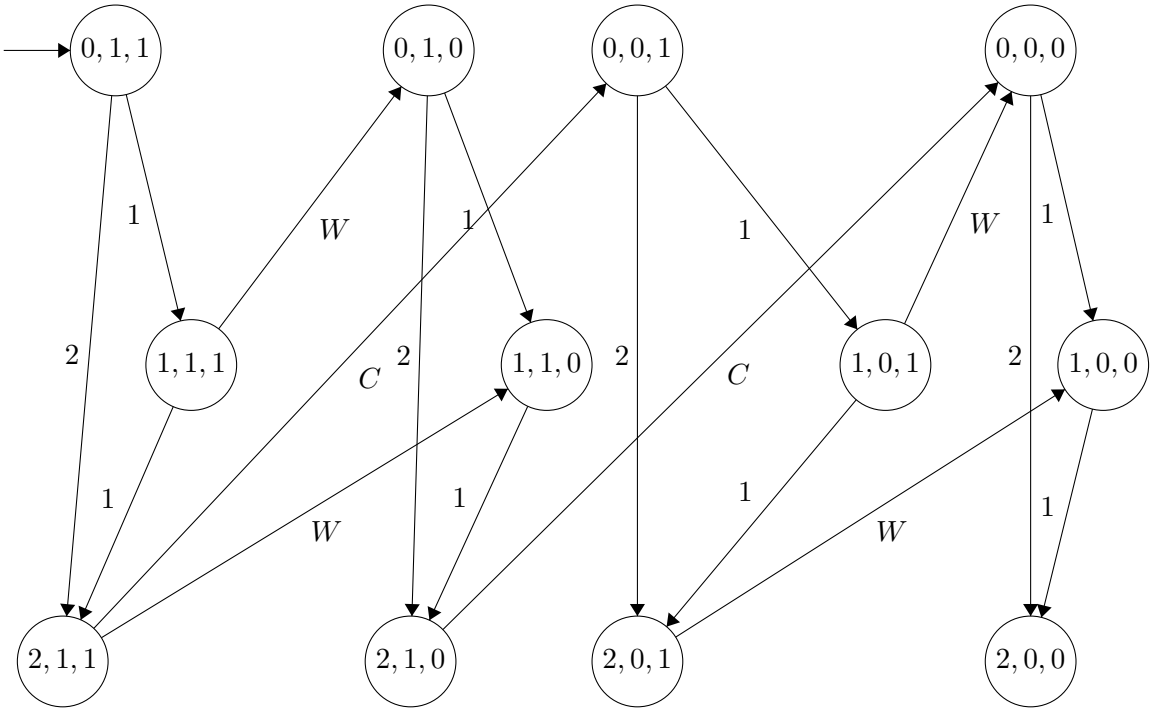
Beispiel: Mustererkennung in Texten: z.B. Suche das Wort "then"



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 $\Sigma_{\text{tast}} = \{a, \ldots, z, A, \ldots, Z, 0, \ldots 9, \ldots\}$

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

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

- Leeres Wort: ϵ (manchmal λ) = 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:

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

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

- Menge aller Wörter einer bestimmten Länge:

Sei Σ Alphabet, so:

$$\begin{aligned} \Sigma^0 &= \{\epsilon\} \\ \Sigma^1 &= \Sigma \\ \Sigma^2 &= \{ab | a, b \in \Sigma\} \\ \Sigma^i &= \{a_1 \dots a_i | a_1 \dots a_i \in \Sigma\} \end{aligned}$$

- Menge aller Wörter über Σ:

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

- Menge aller nichtleeren Wörter über Σ:

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

Beispiel.

Beispiel:

$$\begin{aligned} \Sigma &= \{0, 1\} \\ \Sigma^2 &= \{00, 01, 10, 11\} \\ \Sigma^* &= \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\} \end{aligned}$$

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)**.**

$$\begin{aligned} v \cdot (v \cdot w) &= (u \cdot v) \cdot w \\ |x \cdot y| &= |x| + |y| \\ x \cdot \epsilon &= \epsilon \cdot x = x \end{aligned}$$

Bemerkung (Infixe, Präfixe, Suffixe)**.** Seien $v, w \in \Sigma^*$ für ein Alphabet Σ, 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 funktionert analog.

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

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

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

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 as vor allen bs vorkommen.

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

(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)

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

(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 Σ 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

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

Eine Zahl $p \in N$ ist Primzahl genau dann, wenn $Bin(p) \in L$.
Identifizierung von Sprachen als Probleme.