Theoretische Grundlagen der Informatik

Reinhard Bauer | 19. Oktober 2010

Übersicht

- Organisatorisches
- Inhalt der Vorlesung
- Wiederholung aus Grundbegriffe der Informatik

Organisatorisches

- Termine
- Homepage
- Übungsblätter
- Tutorien
- Klausur

Team

Vorlesung:
Prof. Dorothea Wagner

Übung:
Andrea Schumm (Andrea.Schumm@kit.edu),
Sprechstunde Dienstags 10:00-11:00 Uhr

Reinhard Bauer (Reinhard.Bauer@kit.edu),
Sprechstunde Donnerstags 10:00-11:00 Uhr

Tutorium:
Simon Stroh, Stephan Kochendörfer, Steffen Kühner,
Kai Wallisch, Philipp Schneider, Stefan Kober, Robert Franke,
Ivonne Braun, Tobias Maier, Romuald Brillout, Vladislav Sablin,
Jürgen Haßelwander, Moritz von Looz, Benny Fuhry,
Christian Paul, Andreas Bauer

(vorläufige) Termine

Dienstags		Donnerstags	
19.10.	Vorlesung	21.10.	Vorlesung
26.10.	Vorlesung	28.10.	Vorlesung
02.11.	Vorlesung	04.11.	Übung
09.11.	Vorlesung	11.11.	Vorlesung
16.11.	Vorlesung	18.11.	Übung
23.11.	Vorlesung	25.11.	Vorlesung
30.11.	Vorlesung	02.12.	Übung
07.12.	Vorlesung	09.12.	Vorlesung
14.12.	Vorlesung	16.12.	Übung
21.12.	Vorlesung	23.12.	Vorlesung
11.01.	Vorlesung	13.01.	Vorlesung
18.01.	Vorlesung	20.01.	Übung
25.01.	Vorlesung	27.01.	Vorlesung
01.02.	Vorlesung	03.02.	Übung
08.02.	Vorlesung	10.02.	Vorlesung

Homepage

- `http://i11www.iti.uni-karlsruhe.de/`
- aktuelle Informationen / Termine
- alte Klausuren
- Skript für Vorlesung Informatik III
  (Vorgängervorlesung zu TGI mit ähnlichem Inhalt)
- Folien
- Übungsblätter
- Forum
- Literaturempfehlungen

Forum zur Vorlesung

- verlinkt auf der TGI-Homepage
- Registrierung ohne weitere Zugangsbeschränkung möglich
- Für Fragen an die Übungsleiter
- Für Austausch untereinander

Literaturempfehlungen

- Ingo Wegener
  Theoretische Informatik
  B.G. Teubner Verlag Stuttgart, 1993
- Uwe Schöning
  Theoretische Informatik - kurzgefasst
  Hochschultaschenbuch, Spektrum Akademischer Verlag, 1997

Literaturempfehlungen

- R. Garey und D. S. Johnson
  Computers and Intractibility: A Guide to the Theory of NP-Completeness
  W. H. Freeman, New York, 1979
- T. Cormen, C. Leiserson, R. Rivest
  Introduction to Algorithms
  The MIT press, 1997, 2001.
- A. Asteroth, C. Baier
  Theoretische Informatik: eine Einführung in Berechenbarkeit, Komplexität und formale Sprachen mit 101 Beispielen
  Pearson Studium, 2002

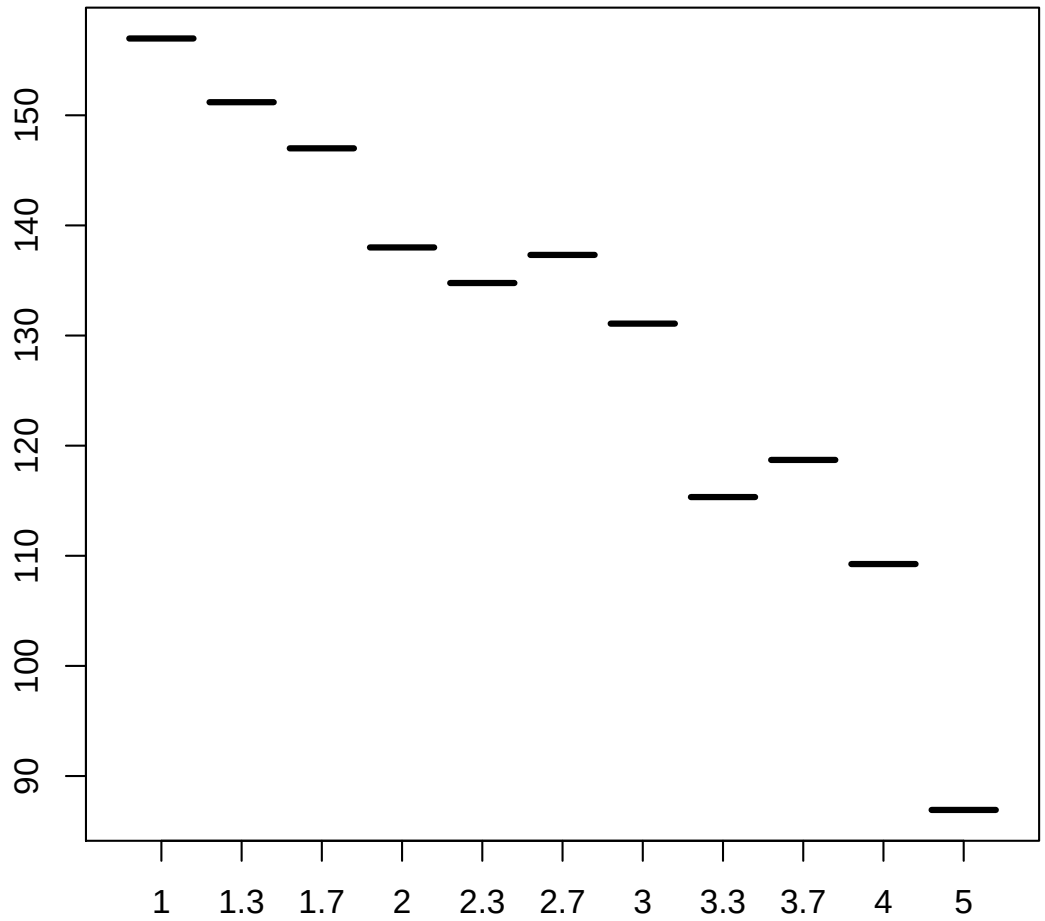
Übungsblätter

- Ausgabe jeden zweiten Donnerstag
- Bearbeitungszeit: 2 Wochen.
- Abgabe: Kasten im Untergeschoss des Informatik-Hauptgebäudes (50.34)
- Abgabe bis Donnerstag, 11.00 Uhr im Kasten oder bis zum Start der Übung im Hörsaal
- Rückgabe der korrigierten Blätter in den Tutorien
- Lösungen in der Übung

Übungsblätter

- Doppelabgabe erlaubt
- Bei Abschreiben: Keine Punkte
- Ausgabe von erstem (halbem) Übungsblatt:
  Donnerstag 28.10.2010.
  Bearbeitungszeit: 1 Woche.
- Ab 50% der erreichbaren Punkte gibt es einen Klausurbonus
- Der Klausurbonus wird nur auf bestandene Klausuren angerechnet

Statistik aus dem WS07/08



Tutorien

- Start: Nächste Woche (ab 25.10.2010)
- Einteilung über webinscribe
- Rückgabe der Übungsblätter
- Zusätzliche Aufgaben und Beispiele zum Stoff der Vorlesung

Klausur

- Orientierung:
  Klausuren für Informatik III auf der Homepage
  aber: leicht veränderte Stoffwahl und 2-Stunden-Klausur
- Klausurbonus ab 50% der erreichbaren Übungsblattpunkte
- Klausurbonus wird nur auf bestandene Klausuren angerechnet
- Termine für Haupt- und Nachklausur stehen noch nicht fest

Ziele der Vorlesung TGI

Welche Fragestellungen werden in TGI behandelt?

- Theoretische Grundlagen zu Algorithmen- und Programmentwurf: Wie kann man allgemeingültige (rechner- und programmiersprachenunabhängige) Aussagen zu gegeben Problemstellungen machen?
- Gibt es Probleme die nicht von Computern gelöst werden können?
- Gibt es Probleme, für die Ausprobieren die beste Lösungsstrategie ist?
- Wie kann man konkrete Computer abstrakt betrachten oder modellieren?
- Wie kann man konkrete Programmiersprachen abstrakt betrachten oder modellieren?

Ziele der Vorlesung TGI

- Vertiefter Einblick in die Grundlagen der Theoretischen Informatik
- Beherrschen der Berechnungsmodelle und Beweistechniken der TI
- Verständnis für Grenzen und Möglichkeiten der Informatik in Bezug auf die Lösung von definierbaren aber nur bedingt berechenbaren Problemem
- Abstraktionsvermögen für grundlegende Aspekte der Informatik von konkreten Gegebenheiten wie konkreten Rechnern oder Programmiersprachen
- Anwendung der erlernten Beweistechniken bei der Spezifikation von Systemen der Informatik und für den systematischen Entwurf von Programmen und Algorithmen

Wiederholung aus Grundbegriffe der Informatik

- Wörter
- Formale Sprachen
- Reguläre Ausdrücke
- Endliche Automaten
- Kontextfreie Grammatiken

Wörter

- Ein *Alphabet* Σ ist eine endliche Menge von Zeichen.
  Beispiel: $\Sigma = \{0, 1\}$.
- Ein *Wort* w über einem Alphabet Σ ist eine (möglicherweise leere) Folge von Zeichen aus Σ.
  Beispiel: $\Sigma = \{0, 1\}$ und $w = 0010010$.
- Das *leere Wort* wird mit ε symbolisiert.
- Die *Menge aller Wörter* über einem Alphabet Σ wird mit Σ^* abgekürzt. Beispiel:
  $\Sigma = \{0, 1\}$ und $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

Wörter

- Die *Menge aller Wörter* mit Länge n über einem Alphabet Σ wird mit Σ^n abgekürzt. Beispiel $\Sigma = \{0, 1\}$:

$$\begin{aligned} \Sigma^0 &= \{\varepsilon\} \\ \Sigma^1 &= \{0, 1\} \\ \Sigma^2 &= \{00, 01, 10, 11\} \end{aligned}$$

- Die *Konkatenation* (Aneinanderhängen) zweier Wörter w_1 und w_2 wird mit $w_1 \cdot w_2$ oder $w_1 w_2$ abgekürzt.
  Beispiel: $w_1 = 001$ und $w_2 = 110$ dann $w_1 \cdot w_2 = 001110$.

Wörter

- Die *iterierte Konkatenation* eines Wortes w ist
  $w^k := \underbrace{w \cdot \ldots \cdot w}_{k\text{-mal}}$
  Beispiel: $w = 110$.

$$
\begin{aligned}
w^0 &= \epsilon \\
w^1 &= 110 \\
w^2 &= 110110 \\
w^3 &= 110110110
\end{aligned}
$$

Formale Sprachen

- Eine *formale Sprache* L über einem Alphabet Σ ist eine Teilmenge $L \subseteq \Sigma^*$.

  Beispiel: Sprache L' aller Wörter deren vorletztes Zeichen 0 ist

  $$L' = \{w0z \mid w \in \Sigma^*, z \in \Sigma\}$$

- Das Produkt $L_1 \cdot L_2$ der Sprachen L_1 und L_2 ist definiert als

  $$L_1 \cdot L_2 := \{w_1 w_2 \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$$

  Beispiel:

  $$L' =$$

Formale Sprachen

- Eine *formale Sprache* L über einem Alphabet Σ ist eine Teilmenge $L \subseteq \Sigma^*$.

  Beispiel: Sprache L' aller Wörter deren vorletztes Zeichen 0 ist

  $$L' = \{w0z \mid w \in \Sigma^*, z \in \Sigma\}$$

- Das Produkt $L_1 \cdot L_2$ der Sprachen L_1 und L_2 ist definiert als

  $$L_1 \cdot L_2 := \{w_1 w_2 \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$$

  Beispiel:

  $$L' = \Sigma^* \cdot \{00, 01\}$$

Formale Sprachen

- Produkte einer Sprache L mit sich selbst werden abgekürzt als

$$L^k := \underbrace{L \cdot \ldots \cdot L}_{k\text{-mal}}$$

Beispiel $L = \{00, 1\}$:

$$\begin{aligned}
L^0 &= \{\varepsilon\} \\
L^1 &= \{00, 1\} \\
L^2 &= \{00, 1\} \cdot \{00, 1\} = \{0000, 001, 100, 11\} \\
L^3 &= \{0000, 001, 100, 11\} \cdot \{00, 1\} \\
&= \{000000, 00100, 10000, 1100, 00001, 0011, 1001, 111\}
\end{aligned}$$

Formale Sprachen

Läßt sich ein Wort w schreiben als $w = u \cdot v \cdot x$, wobei u, v, x beliebige Wörter sind, so heißt:

$$\left.\begin{array}{ll} u & \text{Präfix} \\ v & \text{Teilwort} \\ x & \text{Suffix} \end{array}\right\} \text{von } w$$

Beispiel w=TAL

Präfixe	$P = \{\varepsilon, \text{T}, \text{TA}, \text{TAL}\}$
Suffixe	$S = \{\varepsilon, \text{L}, \text{AL}, \text{TAL}\}$
Teilworte	$\{A\} \cup P \cup S$

Formale Sprachen

Seien $L, L_1, L_2 \subseteq \Sigma^*$ Sprachen.

Produktsprache	$L_1 \cdot L_2 := \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$
k-faches Produkt	$L^k := \{w_1 \cdot w_2 \cdot \ldots \cdot w_k \mid w_i \in L \text{ für } 1 \leq i \leq k\}$ $L^0 := \{\epsilon\}$
Quotientensprache	$L_1/L_2 := \{w \in \Sigma^* \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$
Kleene'scher Abschluss	$L^* := \bigcup_{i \geq 0} L^i$
Positiver Abschluss	$L^+ := \bigcup_{i \geq 1} L^i$
Komplementsprache	$L^c := \Sigma^* \setminus L$

Reguläre Sprachen

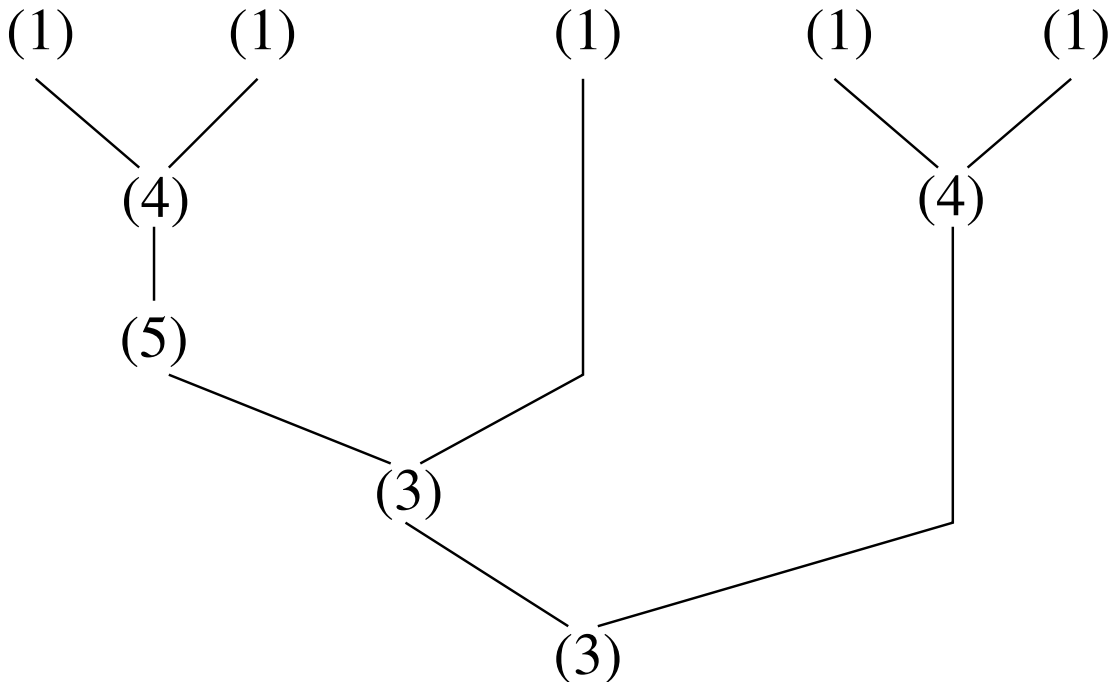
Eine Sprache $L \subseteq \Sigma^*$ heißt *regulär*, wenn für sie einer der folgenden Punkte gilt: (induktive Definition)

1. Verankerung:
   1. $L = \{a\}$ mit $a \in \Sigma$ oder
   2. $L = \emptyset$
2. Induktion: Seien L_1, L_2 reguläre Sprachen
   1. $L = L_1 \cdot L_2$ oder
   2. $L = L_1 \cup L_2$ oder
   3. $L = L_1^*$

Reguläre Sprachen

Beispiel: Die Sprache aller Wörter über $\{0, 1\}$, die als vorletztes Zeichen eine 0 haben

$$L := (\{0\} \cup \{1\})^* \cdot \{0\} \cdot (\{0\} \cup \{1\})$$



Reguläre Ausdrücke

- Wir benutzen eine leicht andere Schreibweise für reguläre Ausdrücke als in GBI.
- Sei Σ eine Alphabet. Eine reguläre Sprache über Σ kann durch einen *regulären Ausdruck* beschrieben werden.

Reguläre Ausdrücke

Dabei bezeichnet

- $\emptyset$ den regulären Ausdruck, der die leere Menge beschreibt.
- ε den regulären Ausdruck, der die Menge $\{\varepsilon\}$ beschreibt.
- a den regulären Ausdruck, der die Menge $\{a\}$ beschreibt.

Wenn α, β reguläre Ausdrücke sind, die die Sprachen $L(\alpha), L(\beta)$ beschreiben, so schreiben wir

- $\alpha \cup \beta$ für $L(\alpha) \cup L(\beta)$
- $\alpha \cdot \beta$ für $L(\alpha) \cdot L(\beta)$
- α^+ für $L(\alpha)^+$
- a^* für $L(\alpha)^*$

Reguläre Ausdrücke

Notation

- Wir schreiben auch α statt $L(\alpha)$ und $w \in \alpha$ statt $w \in L(\alpha)$.
- Wir lassen unnötige Klammern weg.

Reguläre Ausdrücke - Beispiele

- L ist der reguläre Ausdruck für die Sprache aller Wörter über $\{0,1\}$, die als vorletztes Zeichen eine 0 haben

$$L = (0 \cup 1)^* 0 (0 \cup 1)$$

- $L := \{w \in \{0,1\}^* \mid w \text{ enthält 10 als Teilwort}\}$

$$L = (0 \cup 1)^* 10 (0 \cup 1)^*$$

- $L := \{w \in \{0,1\}^* \mid w \text{ enthält 101 als Teilwort}\}$

$$L = (0 \cup 1)^* 101 (0 \cup 1)^*$$

Reguläre Ausdrücke - Beispiele

- $L := \{w \in \{0,1\}^* \mid w \text{ enthält 10 nicht als Teilwort}\} = 0^*1^*$,

  denn
  - $w \in 0^*1^* \Rightarrow w \in L$; also ist $0^*1^* \subseteq L$.
  - Sei $w \in L$. Dann kommen nach der ersten Eins keine Nullen mehr vor, d.h. $w = w'1 \ldots 1$ wobei w' keine 1 enthält. Also ist $w \in 0^*1^*$.

Endliche Automaten

- In GBI wurden Mealy- und Moore-Automaten behandelt.
- In TGI werden nur endliche Akzeptoren benötigt.

Endliche Automaten

Ein (deterministischer) endlicher Automat DEA $(Q, \Sigma, \delta, s, F)$ besteht aus:

- Q, einer endlichen Menge von *Zuständen*
- Σ, einer endlichen Menge von *Eingabesymbolen*
- $\delta\colon Q \times \Sigma \to Q$, einer *Übergangsfunktion*
- $s \in Q$, einem *Startzustand*;
- $F \subseteq Q$, einer Menge von *Endzuständen*.

Endliche Automaten

Der Automat heißt

- endlich, da die Zustandsmenge (vgl. mit Speicher, Gedächtnis) endlich ist.
- deterministisch, da δ eine Funktion ist und der Automat somit in jedem Schritt eindeutig arbeitet. Es gibt keine Zufälligkeiten oder Wahlmöglichkeiten.

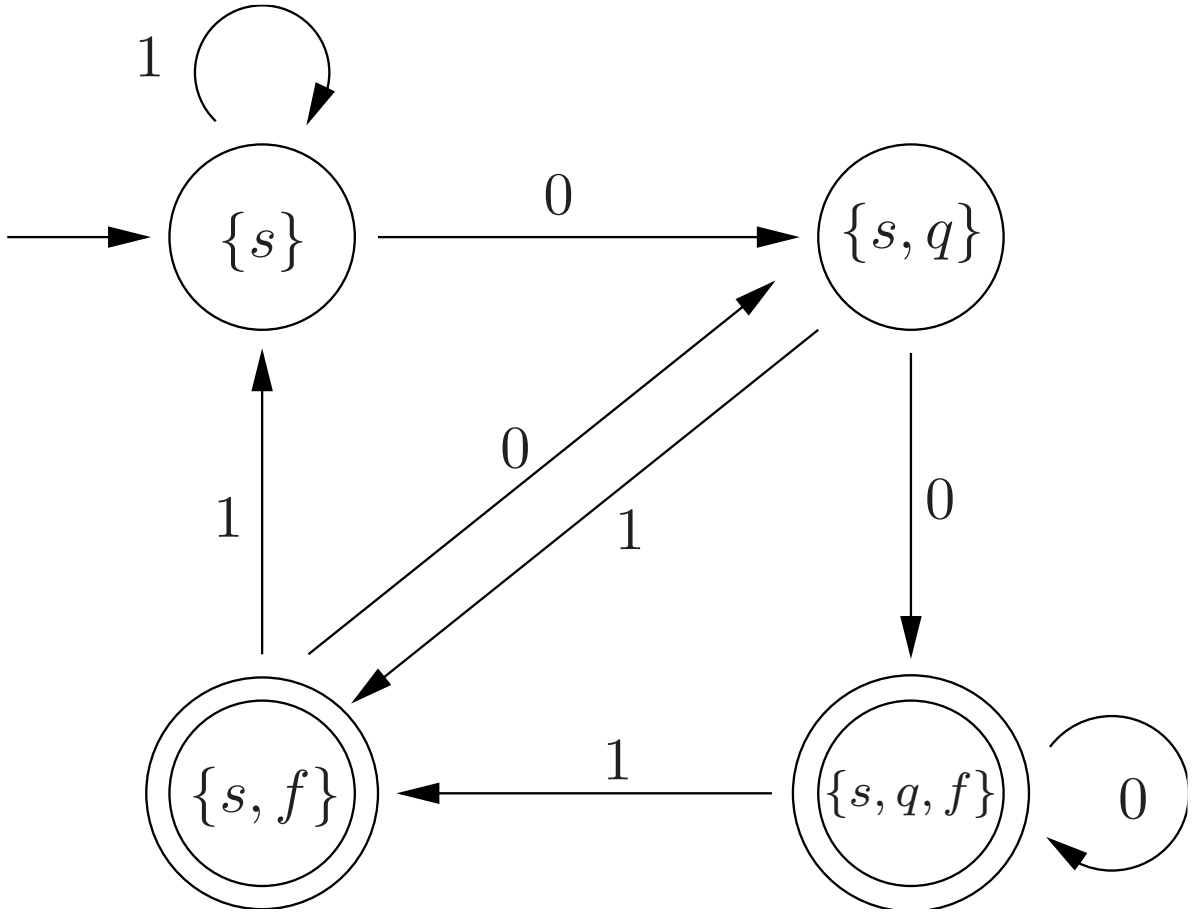
Was kann ein endlicher Automat?

- Gegeben ist eine Eingabe als endliche Folge von Eingabesymbolen. Der Automat entscheidet, ob die Eingabe zulässig ist oder nicht, indem er in einem Endzustand endet oder nicht.

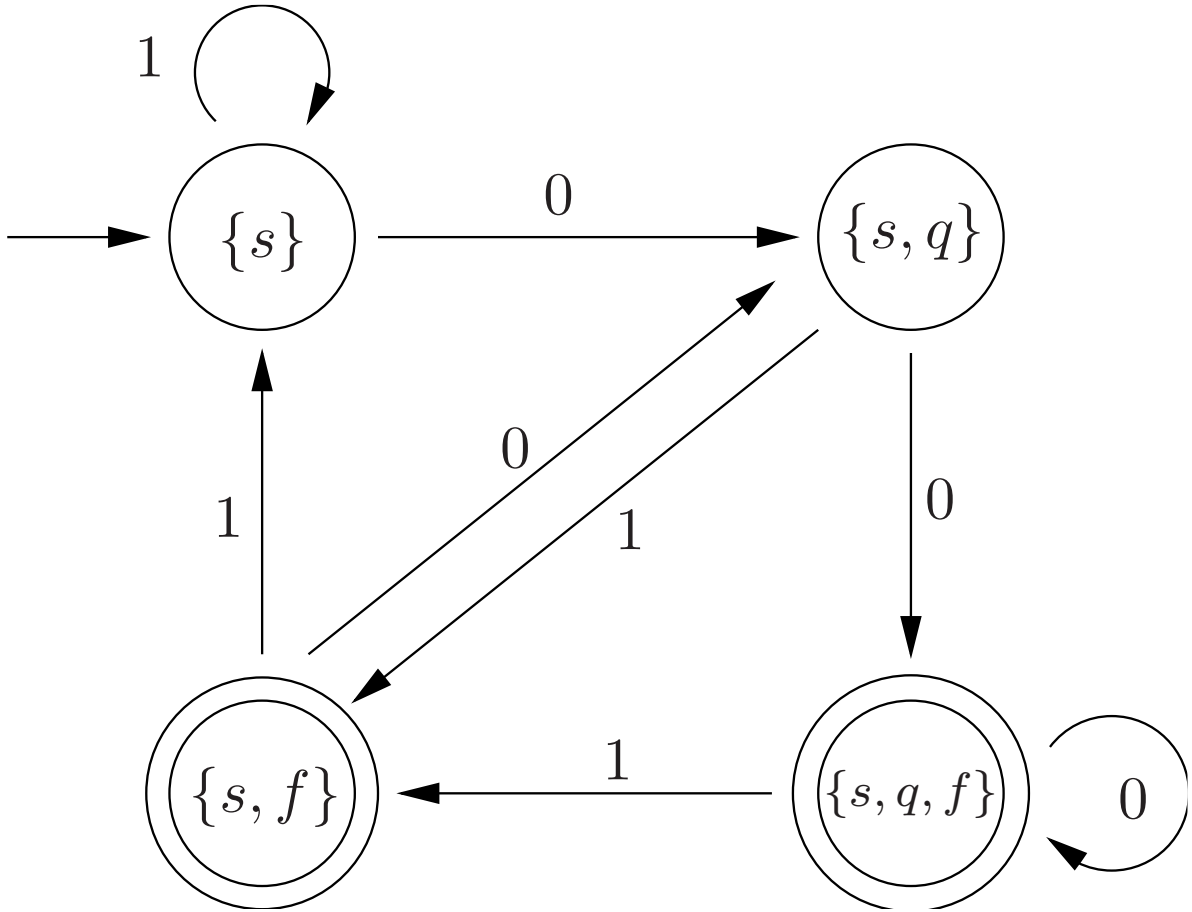
Endliche Automaten

- Ein endlicher Automat **erkennt** oder **akzeptiert** eine Sprache L, d.h. eine Menge von Wörtern über dem Alphabet des Automaten, wenn er nach Abarbeitung eines Wortes w genau dann in einem Endzustand ist, wenn das Wort w in der Sprache L ist ($w \in L$).
- Eine formale Sprache heißt *endliche Automatensprache*, wenn es einen endlichen Automaten gibt, der sie erkennt.

Endliche Automaten



Endliche Automaten



Sprache aller Wörter, deren vorletztes Symbol 0 ist:

$L = (0 \cup 1)^*0(0 \cup 1)$

Kontextfreie Grammatiken

Eine kontextfreie Grammatik $G = (\Sigma, V, S, R)$ ist gegeben durch

- ein endliches Alphabet Σ (auch Terminalalphabet genannt)
- eine endlichen Menge V mit $V \cap \Sigma = \emptyset$ von Variablen (auch Nichtterminale genannt)
- ein Startsymbol $S \in V$
- eine endlichen Menge von Ableitungsregeln R, d.h. durch eine Menge von Tupeln $(l, r) \in V \times (\Sigma \cup V)^*$ (auch Produktionen genannt)

Wir schreiben Produktionen auch in der Form $l \to r$.

Kontextfreie Grammatiken

Gegeben ist eine Regel $l \to r$.

- Wenn in einem Wort w das Zeichen l vorhanden ist, darf l in w durch r ersetzt werden.
- Wir schreiben $w \to z$, wenn w durch Anwendung *einer* Ableitungsregel in z verwandelt wird.
- Wir schreiben $w \overset{*}{\to} z$, wenn w durch Anwendung *mehrerer* Ableitungsregel in z verwandelt wird.

Die von einer Grammatik $G = (\Sigma, V, S, R)$ erzeugte Sprache $L(G)$ ist die Menge aller Wörter $z \in \Sigma^*$ für die gilt $S \overset{*}{\to} z$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S\} \\ R &:= \{S \to 01, S \to 0S1\} \text{ wir schreiben dafür kurz } \{S \to 01|0S1\} \end{aligned}$$

Welche Wörter erzeugt G, d.h. was ist $L(G)$?

$$\begin{aligned} S &\to 01|0S1 \to 01|0011|00S11 \\ &\to 01|0011|000111|000S111 \to \ldots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S\} \\ R &:= \{S \to 01, S \to 0S1\} \text{ wir schreiben dafür kurz } \{S \to 01|0S1\} \end{aligned}$$

Welche Wörter erzeugt G, d.h. was ist $L(G)$?

$$\begin{aligned} S &\to 01|0S1 \to 01|0011|00S11 \\ &\to 01|0011|000111|000S111 \to \ldots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S\} \\ R &:= \{S \to 01, S \to 0S1\} \text{ wir schreiben dafür kurz } \{S \to 01|0S1\} \end{aligned}$$

Welche Wörter erzeugt G, d.h. was ist $L(G)$?

$$\begin{aligned} S &\to 01|0S1 \to 01|0011|00S11 \\ &\to 01|0011|000111|000S111 \to \ldots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S\} \\ R &:= \{S \to 01, S \to 0S1\} \text{ wir schreiben dafür kurz } \{S \to 01 | 0S1\} \end{aligned}$$

Welche Wörter erzeugt G, d.h. was ist $L(G)$?

$$\begin{aligned} S &\to 01 | 0S1 \to 01 | 0011 | 00S11 \\ &\to 01 | 0011 | 000111 | 000S111 \to \ldots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Welche Grammatik erzeugt die Sprache über dem Alphabet $\Sigma = \{0, 1\}$ deren vorletztes Zeichen 0 ist?

Beispiel Kontextfreie Grammatiken

Welche Grammatik erzeugt die Sprache über dem Alphabet $\Sigma = \{0, 1\}$ deren vorletztes Zeichen 0 ist?

Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &:= \{0, 1\} \\ V &:= \{S, A, B\} \\ R &:= \{S \to A0|A1, \quad A \to B0, \quad B \to \epsilon|B0|B1\} \end{aligned}$$

Ableitung:

$$S \to A0|A1 \to B00|B01 \to \ldots$$

Kontextfreie Grammatiken

In TGI werden wir Grammatiken kennenlernen, deren Produktionen *mächtiger* sind, als die von kontextfreien Grammatiken.