

Automaten und Formale Sprachen

Vorlesung 1:

Vorbemerkungen, Wörter und Sprachen



Prof. Dr. Dietrich Kuske



FG Automaten und Logik, TU Ilmenau

Wintersemester 2026/27

Einführung

Wörter, Sprachen und Grammatiken

Rechtslineare Sprachen

Kontextfreie Sprachen

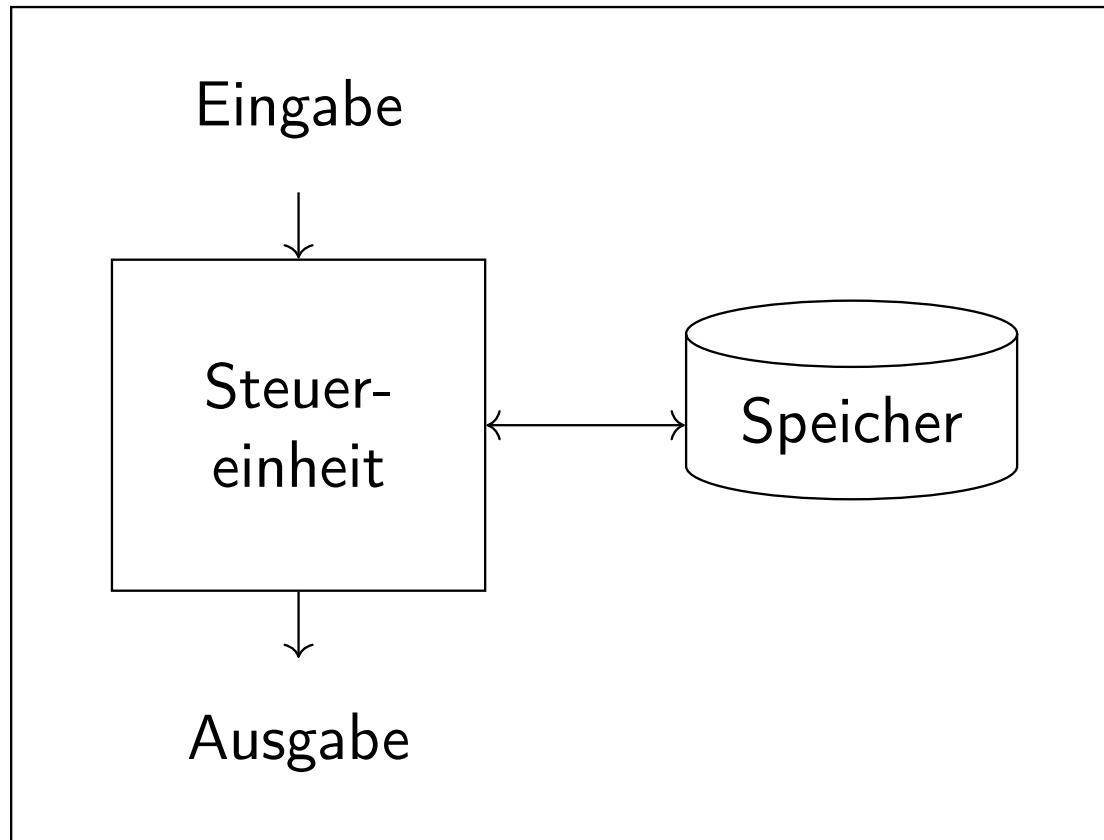
Grundthema von AFS und BuK

Grundfrage

Welche **Probleme** können von einem Algorithmus bzw. von einer **Maschine** mit beschränkten **Ressourcen** gelöst werden und welche können nicht gelöst werden?

- Was ist ein „Problem“?
- Was meinen wir mit „Maschine“?
- Welche „Ressourcen“ betrachten wir und wie werden diese „beschränkt“?

Maschinen



Steuereinheit:

- endlich viele Zustände
- ändern sich in Abhängigkeit von Eingabe und Speicherkonfiguration
- produziert Ausgaben
- gibt „Speicherbefehle“

Eingabe: Folge von „Buchstaben“

Ausgabe: gut/schlecht, Folge von „Buchstaben“ oder $\mathbb{N}$

Probleme

Wir betrachten Probleme als Abbildungen

$$f: \begin{array}{ccc} \text{Menge der mgl. Eingaben} & \longrightarrow & \text{Menge der mgl. Ausgaben} \\ E & \longrightarrow & A \end{array}$$

$E =$ alle „Wörter über Eingabealphabet“ oder Tupel natürlicher Zahlen

$A =$ gut/schlecht ($\hat{=}\{1, 0\}$), alle „Wörter über Ausgabealphabet“, oder natürliche Zahlen

Spezialfall $A = \{0, 1\}$

Solche Probleme f heißen **Entscheidungsprobleme**.

Sie sind gegeben durch die Menge $\{w \in E \mid f(w) = 1\} \subseteq E$. Solche Mengen nennen wir „Sprachen“.

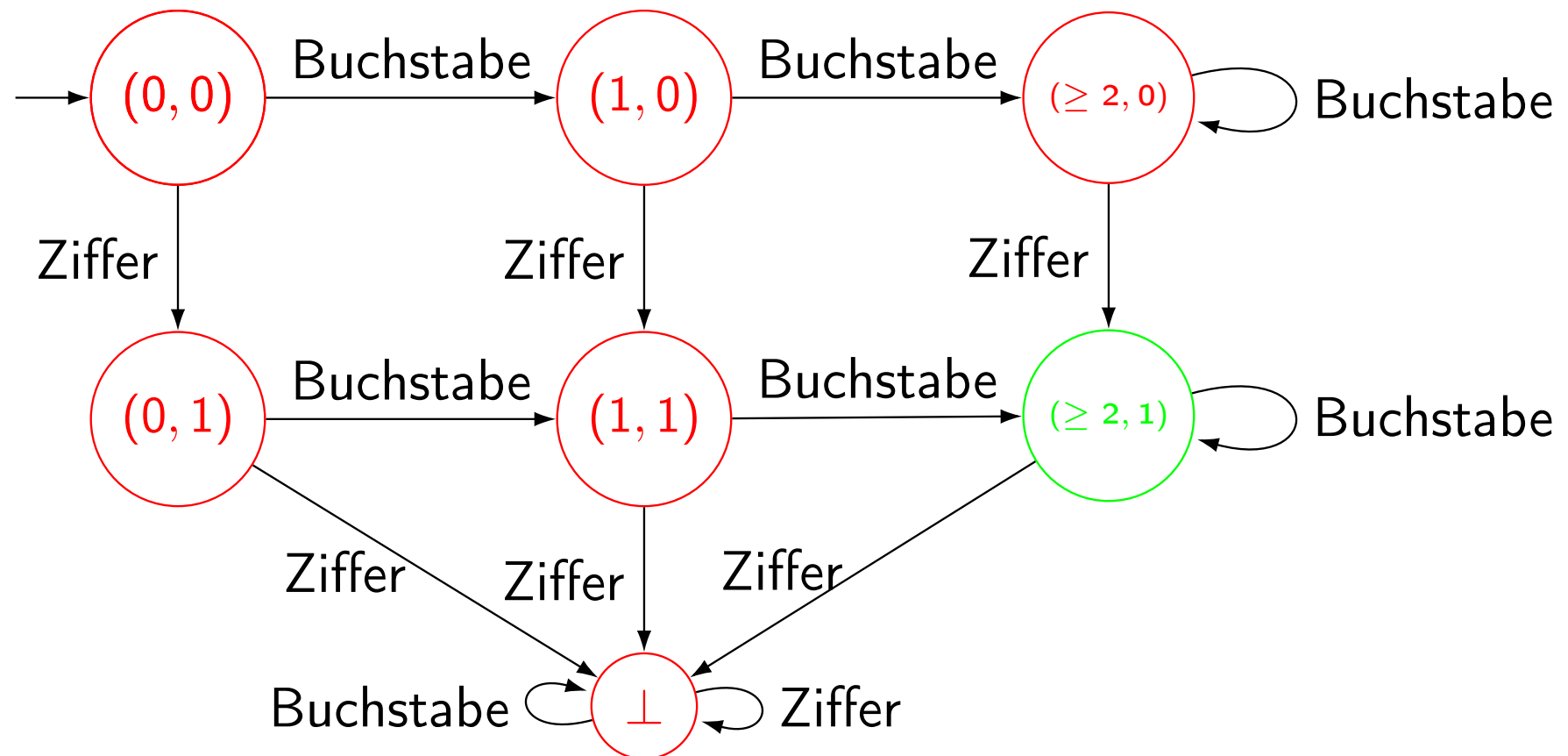
beschränkte Ressourcen

- Art des Speicherzugriffs: verboten / Kellerspeicher / Register / Turing-Band
- Art der Steuereinheit: deterministisch / nichtdeterministisch
- Dauer der Berechnung: polynomiell / exponentiell / ...
- Größe des Speichers: fest / logarithmisch / polynomiell / ...

Beispiel: neue Passwörter

Aufgabe: Überprüfe, ob neues Passwort Vorgaben entspricht!

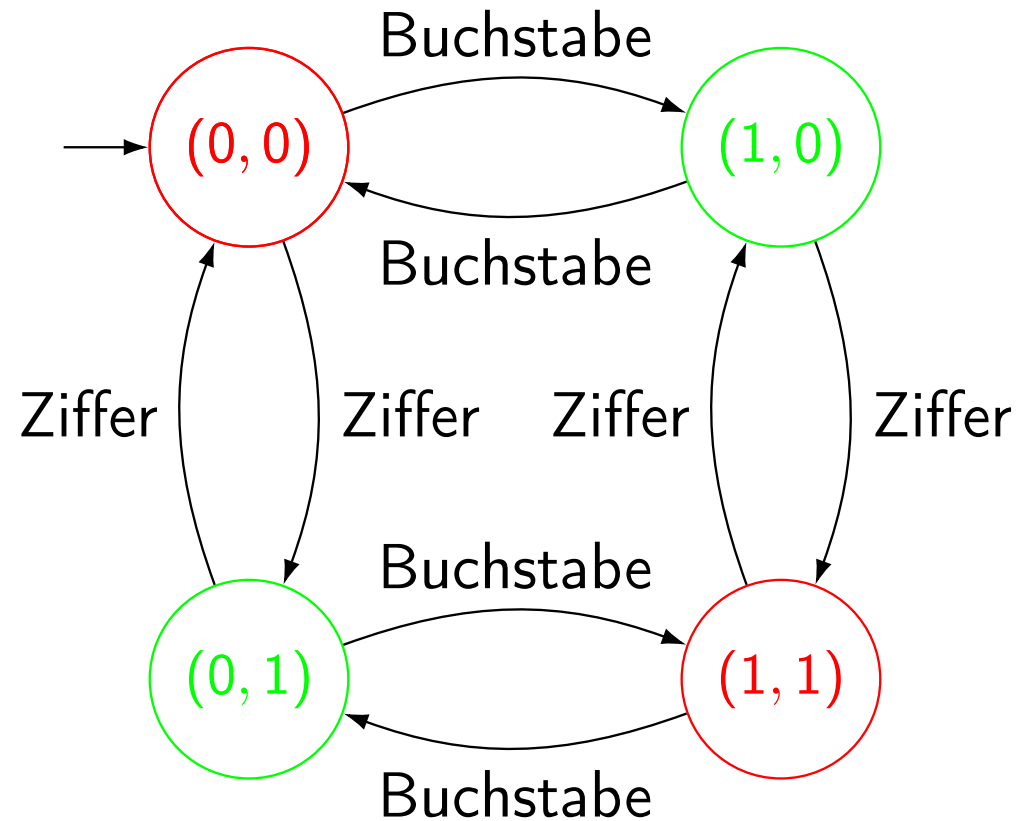
Vorgabe 1: Länge ≥ 3 , wenigstens ein Buchstabe, genau eine Ziffer



Dieser Moore-Automat hat z.B. den folgenden Ablauf:

Eingabe:		a	b	1	a	2	a
Zustand:	(0,0)	(1,0)	$(\geq 2, 0)$	$(\geq 2, 1)$	$(\geq 2, 1)$	$\perp$	$\perp$
Ausgabe:	nein	nein	nein	ja	ja	nein	nein

Vorgabe 2: Anzahl Buchstaben $\not\equiv$ Anzahl Ziffern $\pmod{2}$

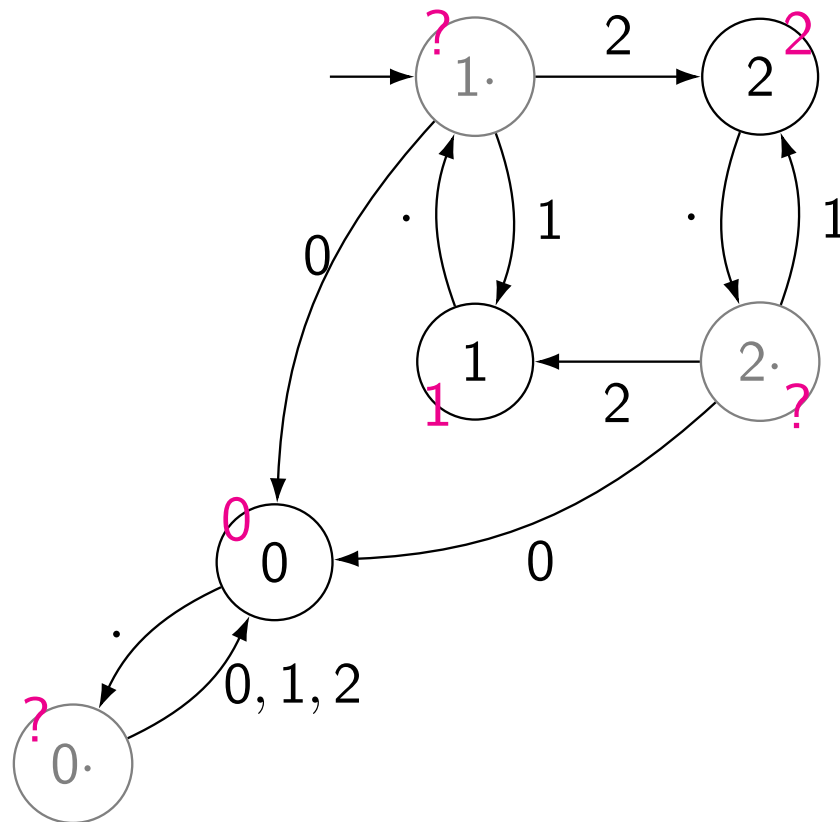


Kann dies mit weniger Zuständen realisiert werden?

Beispiel: algebraische Ausdrücke

Aufgabe: Werte einen algebraischen Ausdruck modulo 3 aus!

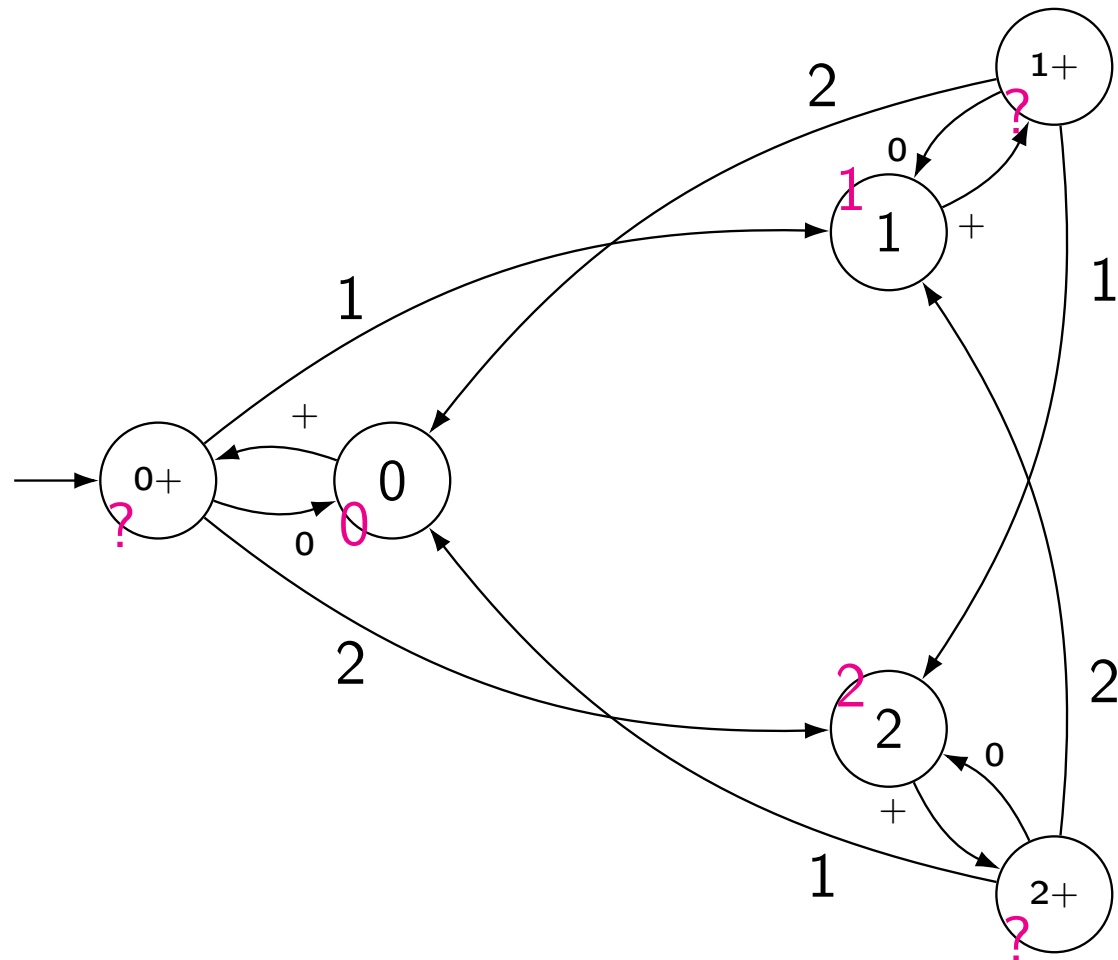
Variante 1: nur Multiplikation modulo 3



Dieser Moore-Automat hat z.B. den folgenden Ablauf:

Eingabe:		1	.	2	.	1	.	2
Zustand:	1.	1	1.	2	2.	2	2.	1
Ausgabe:	?	1	?	2	?	2	?	1

Variante 2: nur Addition modulo 3



Dieser Moore-Automat hat z.B. den folgenden Ablauf:

Eingabe:		1	+	2	+	1	+	2
Zustand:	0+	1	1+	0	0+	1	1+	0
Ausgabe:	?	1	?	0	?	1	?	0

Variante 3: Addition und Multiplikation modulo 3 (ohne Klammern)

z.B. soll Ausgabe wenigstens folgendes erfüllen:

Eingabe: $1 + 2 \cdot 2 + 2$

Zustand:

Ausgabe: ? 1 ? . 2 1

Gibt es solchen Moore-Automaten?

Variante 4: Addition und Multiplikation modulo 3 (mit Klammern)

z.B. soll Ausgabe wenigstens folgendes erfüllen:

Eingabe: $1 + 2 \cdot (2 + 2)$

Zustand:

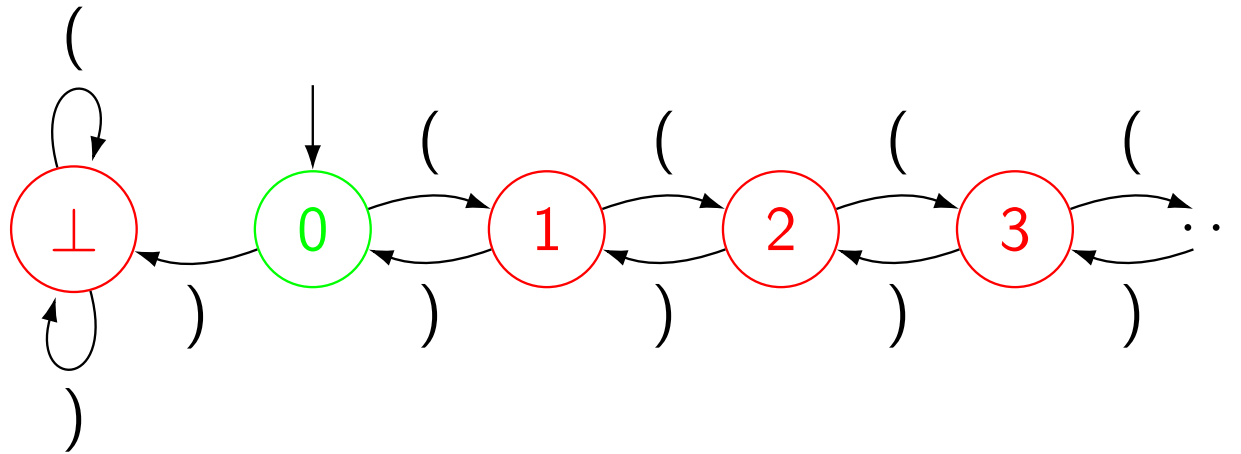
Ausgabe: 0

Gibt es solchen Moore-Automaten?

Beispiel: Klammern

Aufgabe: Erkenne wohlgeklammerte Ausdrücke!

Variante 1: Klammern (und)



Reichen endlich viele Zustände?

Variante 2: Klammern (,), [und]

Gibt es solchen Moore-Automaten?

Allgemeine Frage in der Vorlesung AFS

Welche Verhalten kann man so realisieren?

Dabei ist ein „Verhalten“ eine Menge von Paaren (Eingabefolge, Ausgabefolge).

Beschreibung eines „Verhaltens“: Abbildung Eingabefolgen $\rightarrow$ Ausgabe, wobei eine Eingabefolge auf das Ausgabesymbol abgebildet wird, das im erreichten Zustand produziert wird.

Beispiel: $() \mapsto \text{ja}$ und $((())) \mapsto \text{nein}$

Damit definiert jedes Ausgabesymbol x eine Menge von Eingabefolgen, nämlich diejenigen, die zur Ausgabe von x führen.

Wir untersuchen die „Verhalten“ daher unter dem Gesichtspunkt „Welche Eingabefolgen führen zur Ausgabe von x ?“

Organisatorisches/Rechtliches

- 5 Leistungspunkte ergeben **150 h** Arbeitsaufwand
(Satz 4 §5 PStO-AB Bachelor und Master: „Ein Leistungspunkt entspricht einem Arbeitsaufwand von 30 Stunden.“)
- die Vorlesungszeit hat 15 Wochen, also **10 h / Woche**
- eine Vorlesung, $\frac{1}{2}$ Übung: $2\frac{1}{4}$ h / Woche Präsenzveranstaltungen
- es bleiben also

$7\frac{3}{4}$ h / Woche für das Selbststudium

= Vorlesungs- und Übungsvor- und -nachbereitung

Organisatorisches zur Vorlesung

Informationen, aktuelle Version der Folien, Übungsblätter und Mitschnitte der Vorlesungen: Website der Veranstaltung (Startpunkt: www.tu-ilmenau.de/a1, nicht moodle)

Damit wir Informationen an Sie schicken können, melden Sie sich bitte trotzdem im moodle-Kurs an.

Prüfung: 90-minütige Klausur im Februar/März nächsten Jahres

Literaturempfehlung: Schöning: Theoretische Informatik – kurz gefasst, Spektrum Akademischer Verlag 2008

Erk, Prieze: Theoretische Informatik, Springer 2008.

Kozen: Automata and Computability, Springer 1997.

Hopcroft, Motwani, Ullman: Introduction to Automata Theory, Languages, and Computation, Addison-Wesley 2001.

Organisatorisches zur Übung

Die **Übungen** führt Frau M.Sc. Barthel durch.

Bonuspunkte 1: Die Lösungen der Übungsaufgaben werden abgegeben, (teilweise) korrigiert, und sie ergeben (anteilig) 7% der Klausurpunkte.

Abgabe der Übungsaufgaben: bis Montag 11 Uhr (gerade Woche) in Briefkasten vor Z 1047 oder in Montagsübung

In den Übungen werden mögliche Lösungen und Probleme besprochen ...

Bonuspunkte 2: ... und kurze Tests geschrieben (in jeder 2. Übung, erstmalig am 26. und 28.10.), in denen Definitionen, Sätze usw. abgefragt werden, diese ergeben (anteilig) 3% der Klausurpunkte.

Darüber hinaus führt Frau Barthel eine regelmäßige **Konsultation** durch, bei der Sie Fragen zum Vorstellungsstoff stellen können (Mo U 13 Uhr, Raum Sr HU 129).

Arbeitsweise

1. Sie kommen natürlich zu jeder Vorlesung und hören aktiv zu.
2. Der anspruchsvolle Stoff kann nicht durch alleiniges Hören verstanden werden. Daher werden Sie den Vorlesungsstoff semesterbegleitend nacharbeiten: Definitionen („Konzepte“) und Sätze („Sachverhalte“) herausschreiben und auswendig lernen, Beweise („Begründungen“) verstehen (= wiedergeben können), weitere Literatur zu Rate ziehen
3. Übungen leben von Ihrer Mitarbeit, sie werden (frühzeitig!) vor- und später nachgearbeitet.
4. Zu jeder Veranstaltung bringen Sie sämtliche Unterlagen zum Nachschlagen mit.
5. Bei Verständnisproblemen fragen Sie bitte frühzeitig!
6. Kein Vorlesungs- oder Übungsmaterial wird von Ihnen auf irgendwelchen Servern veröffentlicht.

Einführung

Wörter, Sprachen und Grammatiken

Rechtslineare Sprachen

Kontextfreie Sprachen

natürliche Zahlen: $\mathbb{N} = \{0, 1, 2, 3, \dots\}$

Definition

Sei X eine Menge.

- $X^* = \bigcup_{n \in \mathbb{N}} X^n$ ist die Menge der Tupel beliebiger Länge über X .
- $X^+ = \bigcup_{n \geq 1} X^n = X^* \setminus \{()\}$... Menge der nichtleeren Tupel über X .

Beispiele

Elemente von $\mathbb{N}^*$: $(2, 3, 2, 2, 1, 6), (4, 3, 0, 1), (3, 0, 5), (3, 3), (1), ()$

Elemente von $\{a, b, c, d\}^*$: $(a, b, a), ()$

Elemente von $\{A, \dots, Z, 0, \dots, 9\}^*$: $(A, F, S, 2, 0, 2, 6), ()$

$$\emptyset^* = \{()\}$$

$$\emptyset^+ = \emptyset$$

Definition

Sei X Menge und $w = (x_1, \dots, x_n) \in X^*$.

- $|w| = n$ ist die **Länge** von w .
- Für $x \in X$ ist $|w|_x$ die Anzahl der Vorkommen von x in w , d.h.
 $|w|_x = |\{i \mid 1 \leq i \leq n, x_i = x\}|$.

Bemerkung

Es gelten also

- $X^n = \{w \in X^* : |w| = n\}$ und
- $|()| = 0$.
- $|(a, b, b, a, c)|_a = 2, |(a, b, b, a, c)|_c = 1$

Definition

Ein **Alphabet** ist eine endliche nichtleere Menge.

üblicherweise heißen unsere Alphabete Σ , Γ , Δ

Beispiele

- Alphabete: $\{0\}$, $\{0, 1, 2\}$, ...
 $\{A, \dots, Z, 0, \dots, 9\}$
- keine Alphabete: $\emptyset$, $\mathbb{N}$, $\mathbb{Q}$

Ist Σ Alphabet, so nennen wir die Elemente oft **Buchstaben**.

Beispiel

Das Alphabet $\{0, 1, 2\}$ hat also die drei Buchstaben 0, 1 und 2.

Das Alphabet $\{\square, \diamond\}$ hat die zwei Buchstaben $\square$ und $\diamond$.

Ist Σ ein Alphabet, so heißen die Elemente von Σ^* auch **Wörter über Σ** (auch: **String**, **Zeichenkette**).

Beispiel

- (0) , $()$ und $(1, 2, 0, 0)$ sind also Wörter über dem Alphabet $\{0, 1, 2\}$.
- $(\square)$, $(\diamond, \square)$, $(\diamond, \square, \diamond)$ und $()$ sind Wörter über dem Alphabet $\{\square, \diamond\}$.

Schreibweise: Für das Wort $(a_1, a_2, \dots, a_n)$ schreiben wir auch $a_1 a_2 \dots a_n$, typische Wörter: $u, v, w, x, y, z, w_0, w_1, \dots$

Beispiel

$(1, 2, 0, 0)$ wird geschrieben als 1 2 0 0
 (1) wird geschrieben als 1
 $()$ wird geschrieben als ε (das **leere Wort**)

Definition

Sind $u = (a_1, a_2, a_3, \dots, a_m)$ und $v = (b_1, b_2, b_3, \dots, b_n)$ Wörter, so ist $u \cdot v$ das Wort $(a_1, a_2, a_3, \dots, a_m, b_1, b_2, b_3, \dots, b_n)$; es wird als **Verkettung** oder **Konkatenation** von u und v bezeichnet.

Schreibweise: An Stelle von $u \cdot v$ schreibt man auch uv .

Beobachtung

$\cdot : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ ist eine Abbildung (oder eine binäre Operation auf Σ^*) mit

- $(u \cdot v) \cdot w = u \cdot (v \cdot w)$ („Assoziativität“ - Klammern unnötig)
- $\varepsilon \cdot u = u \cdot \varepsilon = u$ („neutrales Element“- in Produkten unnötig).

Kürzer: $(\Sigma^*, \cdot, \varepsilon)$ ist ein **Monoid**.

$(\mathbb{N}, \cdot, 1)$ und $(\mathbb{R}^{2 \times 2}, \cdot, E_{2 \times 2})$ sind auch Monoide.

Definition

Für $w \in \Sigma^*$ und $n \in \mathbb{N}$ ist w^n induktiv definiert:

$$w^n = \begin{cases} \varepsilon & \text{falls } n = 0 \\ w \cdot w^{n-1} & \text{falls } n > 0 \end{cases}$$

Beispiel

$$\begin{aligned} 001100110011 &= 0011 \cdot 0011 \cdot 0011 = (0^2 1^2)^3 \\ abacabacabacabac &= abac \cdot abac \cdot abac \cdot abac \\ &= (abac)^4 \\ abacccc &= aba \cdot cccc \\ &= abac^4 \end{aligned}$$

d.h. die Potenzierung bindet stärker als die Verkettung.

Definition

- Sei Σ ein Alphabet.
Teilmenge von Σ^* werden **formale Sprachen über Σ** genannt.
- Eine Menge L ist eine **formale Sprache**, wenn es ein Alphabet Σ gibt, so daß L formale Sprache über Σ ist (d.h. so daß $L \subseteq \Sigma^*$).

Beispiel

1. Die Menge P_{10} der Dezimaldarstellungen von Primzahlen ist eine formale Sprache: das Alphabet ist $\{0, 1, 2, \dots, 9\}$.
2. Die Menge P_2 der Binärdarstellungen von Primzahlen ist eine formale Sprache: das Alphabet ist $\{0, 1\}$.
Wegen $10 \in P_2 \setminus P_{10}$ gilt $P_2 \neq P_{10}$.
3. Die Menge P der Primzahlen ist keine formale Sprache, denn was sollte das Alphabet sein?

Verabredung

Ab jetzt sagen wir oft einfach **Sprache** für „formale Sprache“.

Sprechweise: „Klasse“ von Sprachen (nicht „Menge“).

Beispiel

Die „Klasse aller Sprachen“,
die „Klasse der unendlichen Sprachen“,
die „Klasse der Sprachen über einem einelementigen Alphabet“,
die „Klasse der regulären Sprachen“, usw.

(Diese Gesamtheiten sind keine Mengen im strengen Sinne.)

beispielhafte Testfragen aus dieser Vorlesung (andere sind möglich):

- Was ist ein Alphabet?
- Was ist Σ^* für eine Menge Σ ?
- Was ist eine formale Sprache?

Zusammenfassung 1. Vorlesung

in dieser Vorlesung neu

- allgemeine Frage für AFS
- Grundbegriffe: Alphabet
- (formale) Sprache

kommende Vorlesung

- grundlegende Methode zur Beschreibung formaler Sprachen
- Chomsky-Hierarchie