

AUTOMATENTHEORIE UND FORMALE SPRACHEN

Sommersemester 2023

13. April 2023

Prof. Dr. Steffen Reith

Theoretische Informatik
Studienbereich Angewandte Informatik
Hochschule **RheinMain**



Notizen

Notizen

ADMINISTRATIVES

TERMINE

Vorlesung:

Donnerstag 8¹⁵ - 9⁴⁵ UDE-B002

Übung (Montag):

Gruppe	Zeit	Raum
AI Grp A	Do 10 ⁰⁰ - 11 ³⁰	UDE-C037 (Reith)
AI Grp B	Mi 16 ⁰⁰ - 17 ³⁰	UDE-C037 (Reith)
AI Grp C	Mi 17 ⁴⁵ - 19 ¹⁵	UDE-C037 (Reith)

Start der Übungen in der KW 16

3

Notizen

ÜBER DEN DOZENTEN

- Prof. Dr. Steffen Reith, geboren 1968, verheiratet, eine Tochter
- Seit Sommersemester 2006 an der Hochschule RheinMain (FH Wiesbaden)
- Früher: Entwickler für kryptographische und mathematische Algorithmen für tief eingebettete System in KFZs.
- Spezialgebiete: Komplexitätstheorie, Logik in der Informatik, eingebettete Systeme und Kryptographie (Zahlentheorie)
- Abschlussarbeiten: Kryptographie, Kryptographie für eingebettete Systeme, paralleles Rechnen, Komplexitätstheorie, Logik in der Informatik

E-Mail: Steffen.Reith@hs-rm.de

Skype: [Steffen.Reith](#)

Büro: Raum C202

Sprechzeiten: Nach Vereinbarung (per BBB: immer!)

4

Notizen

WEITERE INFORMATIONEN ZUR VORLESUNG

Webseite: <http://www.cs.hs-rm.de/~reith>

Auf der Webseite werden auch die Übungsblätter veröffentlicht (das erste **heute!**). Üblicherweise eine Woche Bearbeitungszeit. Bei **Feiertagschaos** bitte Dozenten befragen!

Literatur:

- John E. Hopcroft and Rajeev Motwani and Jeffrey D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison Wesley Publishing Company, 2001
- Uwe Schöning, Theoretische Informatik - kurzgefasst, Spektrum Akademischer Verlag, 2001
- **Michael Sipser, Introduction to the Theory of Computation, Wadsworth Inc Fulfillment, 2. Auflage, 2005**
- Juraj Hromkovič, Theoretische Informatik - Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kommunikation und Kryptographie, 3. Auflage, Teubner, 2007

5

Notizen

WEITERE INFORMATIONEN ZUR VORLESUNG (II)

Ersatztermine:

Werden mit Hörern abgestimmt

Erklärvideos werden Sie auf PanOpto finden.

Skript:

Wird in (sehr) unregelmäßigen Abständen auf der Webseite der Vorlesung verbessert und erweitert (eine alte Version ist heute online).

Folien:

Einzelne (kleine) Teile der Vorlesung werden in Folienform zur Verfügung stehen. Folien, die vom Skript abweichen, werden auf der Webseite (nachträglich) zur Verfügung stehen (user: theory / PW: fun).

Eine eigene Mitschrift sollte *angefertigt* werden!

6

Notizen

EIN ROTER FADEN

In der Vorlesung werden die folgenden Themen untersucht:

1. Einleitung (grundlegende Begriffe, L-Systeme)
2. Die Chomsky-Hierarchie (Sprachklassen, Wortproblem)
3. Endliche Automaten und reguläre Sprachen (Pumping Lemma)
4. Kontextfreie Sprachen (Normalformen, Kellerautomaten)
5. Kontextsensitive- und Typ0-Sprachen (Turingmaschinen, Unentscheidbarkeit)
6. Komplexität (**P** vs. **NP**)

7

Notizen

WARUM THEORETISCHE INFORMATIK?

Theoretische Informatik wird (wohl aufgrund der mathematischen Prägung) oft als „**schwach motiviert**“, „**langweilig**“ und „**nutzlos**“ beschrieben.

Warum lohnt sich die Theoretische Informatik?

- Konkrete Technologien ändern sich sehr schnell, deshalb sollte man die extrem langlebigen Konzepte verstehen.
- Hintergrundinformationen ermöglichen Chancen und Grenzen von Technologien zu verstehen.
- Theoretische Informatik gibt Hinweise, welche Wege zu keiner Lösung führen werden.
- Verbesserung des strukturierten Denkens und der Problemlösungskompetenz.
- (Tiefgreifende) Ideen führen zu schnellen Algorithmen

8

Notizen

PROGRAMMIERSPRACHEN - FORTRAN77

```

c      Erstellt von Gilberto E. Urroz
c00000000001111111112222222222333333333344444444445555555555666666666777
c234567890123456789012345678901234567890123456789012345678901234567890123
      program function
c      --- declaration of variables
      real x, y
c      --- show function
      print*, "=====
      print*, "Calculate the function y = f(x) defined as"
      print*, "          y = x+1  if x < 1"
      print*, "          y = 2-x  if x >= 1"
      print*, "=====
c      --- request x as input
      print*, "enter a value of x:"
      read*, x
c      --- evaluation of function
      if (x.lt.1.0) then
        y = x+1
      else
        y = 2-x
      end if
c      --- print result
      print*, "the corresponding value of y is: ", y
c      --- end program
      end

```

Warum haben wir heute Programmiersprachen mit „schönerer“ Syntax?

9

Notizen

SPIELREGELN

- *Rechner* und Handys sind zu Beginn der Veranstaltung **aus**
- Wir (Dozent + Hörer) sind **pünktlich**
- Es *redet nur eine Person*
- Bei Fragen und Problemen **sofort melden / fragen**
- Es wird Eigeninitiative und selbstständiges Arbeiten erwartet
- Eine Vorlesung ist keine (wöchentliche) Fernsehserie
 - Eine Vorlesung wird von **den Hörern** und vom Dozenten **gestaltet**
 - aktive Mitarbeit erwünscht und erforderlich
 - Der Dozent will motiviert werden
 - Umfangreiche Vor- und Nachbereitung notwendig
 - Lernen kurz vor der Klausur ist tödlich! (kontinuierliches Lernen)
- Vergessen Sie den (angeblichen) Konflikt von Theorie und Praxis

Was wünschen Sie sich?

10

Notizen

NATÜRLICHE VS. FORMALE SPRACHEN

Notizen

Administratives

Natürliche vs. Formale Sprachen

L-Systeme

NATÜRLICHE SPRACHEN

Natürliche Sprachen legen ihre Struktur durch

- die Regeln einer **Grammatik**
- und eine Menge von **erlaubten Worten** (\triangleq Strings gebildet aus Buchstaben)

fest.

Allerdings müssen syntaktisch korrekte Sätze einer natürlichen Sprache keinen Sinn tragen:

- Wiesbaden wohnt weiterhin weich
- Der bissige Student jagt die verschlafene Mensa

⇒ syntaktisch korrekte Sätze müssen keinen Sinn (\triangleq Semantik) tragen.

Wie kann man diese Beobachtungen in der Informatik ausnutzen?

Notizen

FORMALE REGELN ZUR ERZEUGUNG EINER SPRACHE

Der Linguist Noam Chomsky hatte folgende Idee:

Korrekte Sätze einer (natürlichen) Sprache sollen durch ein **(endliches System) von formalen Regeln** erzeugt werden.

Bis heute ist diese Idee

- in der Linguistik umstritten, aber
- extrem bedeutsam in der Informatik.

Basis für z.B. alle Programmiersprachen / Compilerbau, Auszeichnungssprachen (SGML, XML, HTML, ...).

Ähnlich sind die sogenannten **(Semi) Thue Systeme**, die heute z.B. in Spezialformen in der Computergraphik Bedeutung erlangt haben.

13

Notizen

EINIGE GRUNDLEGENDE BEGRIFFE

Eine endliche Menge Σ heißt **Alphabet**. Die **Elemente** von Σ werden **Buchstaben** genannt. Eine Folge von Buchstaben nennt man **Wort** (über Σ). Eine beliebige Menge von Worten über Σ nennt man dann eine **(formale) Sprache**.

Beispiel (arithmetische Ausdrücke)

Sei $\Sigma = \{ \}, (, +, -, *, /, x \}$ und EXPR die Menge aller korrekten arithmetischen Ausdrücke. Damit gilt

- $(x - x) \in \text{EXPR}$
- $((x + x) * x) / x \in \text{EXPR}$
- $))(x -) * x \notin \text{EXPR}$

EXPR ist eine Menge von Worten über Σ , also kann man EXPR als **formale Sprache** (über $\{ \}, (, +, -, *, /, x \}$) bezeichnen.

14

Notizen

WEITERE BEISPIELE FÜR FORMALE SPRACHEN

Beispiel (Zahlenmengen)

Sei $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, dann sind die folgenden Mengen auch formale Sprachen über Σ :

- PRIMES = $\{2, 3, 5, 7, 11, 13, 17, 19, 23, \dots\}$
- EVEN = $\{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, \dots\}$
- 2POT = $\{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, \dots\}$

15

Notizen

WEITERE BEISPIELE FÜR FORMALE SPRACHEN (II)

Beispiel (Wortmengen über $\{a, b\}$)

Sei $\Sigma = \{a, b\}$, dann sind die folgenden Mengen auch formale Sprachen über Σ :

- BRACKET = $\{ab, aabb, aaabbb, aaaabbbb, \dots\}$
- UODD = $\{a, aaa, aaaaa, aaaaaaa, aaaaaaaaa, \dots\}$
- $\Sigma^* = \text{ALL} = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, \dots\}$

16

Notizen

GRAMMATIKEN UND AUTOMATEN

(Formale) Sprachen enthalten meist unendlich viele Wörter

- Wir brauchen endlich viele **Erzeugungsregeln**, um (algorithmisch) mit formalen Sprachen umgehen zu können. Die Rolle der Regeln übernehmen **Grammatiken**.
- Weiterhin werden **Erkenner** benötigt, die entscheiden, ob ein **Wort zu einer Sprache gehört**. Die Rolle der Erkenner spielen die **Automaten**, die wir in dieser Vorlesung studieren.

17

Notizen

TEIL EINER NATÜRLICHEN SPRACHE

Beispiel (Eine Grammatik)

Satz	→	Subjekt Prädikat Objekt
Subjekt	→	Artikel Attribut Substantiv
Artikel	→	ε der die das
Attribut	→	ε
Attribut	→	Adjektiv
Attribut	→	Adjektiv Attribut
Adjektiv	→	kleine bissige verschlafene
Substantiv	→	Student Katze
Prädikat	→	jagt betritt
Objekt	→	Artikel Attribut Substantiv

Das Symbol „|“ markiert eine Alternative, d.h. $A \rightarrow B \mid C$ ist Abkürzung für die beiden Regeln $A \rightarrow B$ und $A \rightarrow C$

18

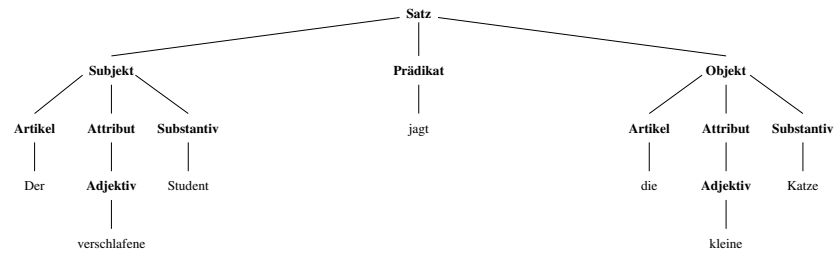
Notizen

TEIL EINER NATÜRLICHEN SPRACHE (II)

Durch Anwendung der Regeln und Ersetzung der fett gedruckten Wörter können z.B. die folgenden Sätze gebildet werden:

- Der kleine bissige Student betritt die verschlafene Mensa
- Der verschlafene Student jagt die kleine Katze

Mit **Syntaxbäumen** man man die **Ableitungsschritte graphisch** verdeutlichen:



19

L-SYSTEME

Notizen

Notizen

L-SYSTEME

- Die **L-Systeme** wurden 1968 durch **Aristid Lindenmeyer** als mathematisches Modell des Pflanzenwachstums eingeführt.
- L-Systeme werden heute in der Computergraphik benutzt, um natürlich wirkende Pflanzen schnell generieren zu können.
- Hier betrachten wir die einfachste Klasse von L-Systemen, die so genannten DOL-System.
- Die Regeln sind deterministisch, d.h. für jeden Buchstaben gibt es **genau eine Regel**.
- Die Regeln sind kontextfrei, d.h. **Ersetzungen hängen nicht** von den **umgebenden Buchstaben** (\triangleq Kontext) ab.

21

Notizen

GRUNDLEGENDE BEGRIFFE UND EIGENSCHAFTEN

Definition (OL-Systeme)

- Mit Σ^* bezeichnen wir die Menge **aller Wörter** über Σ .
- Ein **OL-System** G ist ein Tripel $G = (\Sigma, \omega, P)$, wobei
 - Σ das **Alphabet**, ω das **Axiom** und
 - $P \subseteq \Sigma \times \Sigma^*$ die Menge der **Produktionen**.
- Eine Produktion $(a, \chi) \in P$ wird als $a \rightarrow \chi$ geschrieben. Der Buchstabe a heißt **Vorgänger** und χ **Nachfolger** dieser Produktion.
- Für **jeden** Buchstaben $a \in \Sigma$ **existiert eine Produktion** $(a, \chi) \in P$.
- Ein OL-System heißt **deterministisch**, wenn es für jeden Buchstaben $a \in \Sigma$ nur **genau eine** Produktion $(a, \chi) \in P$ gibt.

22

Notizen

DOL-SYSTEME (II)

Definition

Deterministische OL-Systeme heißen **DOL**-Systeme.

Definition (Ableitung)

Sei $\mu = a_1 \dots a_m$ ein beliebiges Wort über Σ , dann kann $\nu = \chi_1 \dots \chi_m$ aus μ **abgeleitet** werden, wenn

- **für alle** $i = 1, \dots, m$ $(a_i, \chi_i) \in P$ gilt, wobei
- man $\mu \vdash \nu$ schreibt.
- Ein Wort ν heißt **von G generiert**, wenn es in **endlich** vielen Schritten aus dem Axiom abgeleitet werden kann.

23

Notizen

DOL-SYSTEME (III)

Geben wir aus **Bequemlichkeitsgründen** für einen Buchstaben a keine Produktion an, dann gilt **implizit** $(a, a) \in P$.

Achtung: Alle Regeln aus P werden **gleichzeitig** angewendet.

Wird ein Wort ν von $G = (\Sigma, \omega, P)$ generiert, dann können wir also

$$\omega \vdash \mu_1 \vdash \mu_2 \vdash \dots \vdash \mu_n = \nu$$

schreiben (kurz: $\omega \vdash^* \nu$).

24

Notizen

EIN BEISPIEL

Sei $G = (\Sigma, \omega, P)$, wobei

$$\rightarrow \Sigma = \{a, b, c\},$$

$\rightarrow \omega = abc$ und

$$\rightarrow P = \{a \rightarrow aa, b \rightarrow bb, c \rightarrow cc\}.$$

Mit Hilfe dieses DOL-Systems können Worte der Form

$$a^{2^n} b^{2^n} c^{2^n}$$

für $n \geq 0$ abgeleitet werden.

Bemerkung: a^n ist die Abkürzung für $\underbrace{aaa \dots a}_{n\text{-mal}}$

25

Notizen

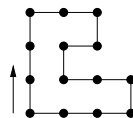
[illegible]

TURTLE-GRAPHIK

Sei δ ein beliebiger Winkel, dann werden die Buchstaben F, f, + und – wie folgt interpretiert:

F	Bewege den Stift um die Länge d und zeichne eine Linie
f	Bewege den Stift um die Länge d und zeichne keine Linie
—	drehe um δ Grad nach rechts
+	drehe um δ Grad nach links

Mit $\delta = 90^\circ$ wird $FFF - FF - F - F + F + FF - F - FFF$ in die Graphik



umgesetzt.


Notizen

[illegible]

EIN BEISPIEL

Beispiel (Kochsche Schneeflocke)

Gegeben sei $G = (\Sigma, \omega, P)$ mit Alphabet $\Sigma = \{F, +, -\}$, Axiom $\omega = F$ und der Menge der Produktionen $\{F \rightarrow F + F - -F + F\}$.
Wir legen $\delta = 45^\circ$ fest. Für die Anzahl der Schritte n ergibt sich:

$n = 1$	$F + F - -F + F$	
$n = 2$	$F + F - -F + F + F + F$ $- -F + F - -F + F - -$ $F + F + F + F - -F + F$	

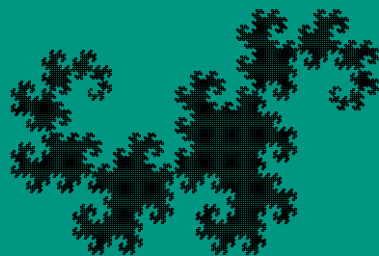
27

Notizen

EIN ZWEITES BEISPIEL

Beispiel (Drachenkurve)

Sei $\delta = 90^\circ$ und das L-System $G = (\{F_r, F_l, +, -\}, F_l, \{F_l \rightarrow F_l + F_r +, F_r \rightarrow -F_l - F_r\})$, dann ergibt sich



Sowohl F_l als auch F_r werden als „Bewege den Stift einen Schritt der Länge d und zeichne eine Linie“ interpretiert.

28

Notizen

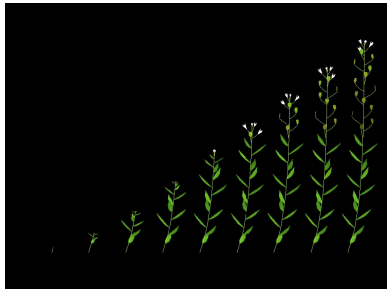
WEITERFÜHRENDE LITERATUR

Przemyslaw Prusinkiewicz und Aristid Lindenmayer, The Algorithmic Beauty of Plants

unter

<http://algorithmicbotany.org/papers/#abop>

Die folgenden Graphiken wurden diesem Buch entnommen:



Notizen

Notizen
