Technische Universität Dresden

Formale Systeme

26. Vorlesung: Zusammenfassung und Ausblick

Markus Krötzsch

Professur für Wissensbasierte Systeme

TU Dresden, 29. Januar 2026

Euklid als Informatiker

Geometrie nach Euklid

Etwa im 3. Jhd. v. Chr. veröffentlicht Euklid sein Lehrbuch Die Elemente und begründet darin die euklidische Geometrie.

Zentrales Thema der euklidischen Geometrie ist die Konstruktion mit den euklidischen Werkzeugen:

- Lineal: beliebig lang, aber ohne Markierungen
- Zirkel: zeichnet Kreise, aber trägt bei Euklid keine Längen ab (kollabierend)

Die Konstruktion mit diesen idealen Werkzeugen gilt bei den Griechen und noch Jahrhunderte später als Königsdisziplin der Mathematik

Konstruktion mit Zirkel und Lineal

Erlaubte Konstruktionsschritte:

(1) Ziehen einer beliebig langen Geraden durch zwei verschiedene Punkte

(2) Zeichnen eines Kreises mit einen gegebenen Mittelpunkt, der durch einen gegebenen Punkt verläuft

(3) Abtragen einer Strecke mit dem Zirkel

Bei Euklid nicht direkt erlaubt, aber Euklid selbst hat bewiesen, dass diese Operation als Makro mithilfe der Operationen (1) und (2) darstellbar ist

Beispiel

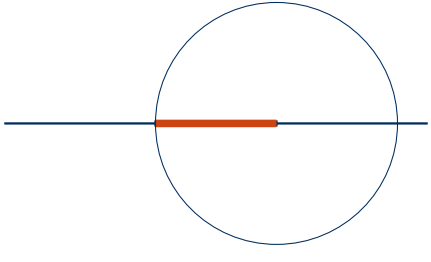
Man kann ein Quadrat wie folgt konstruieren:

Beispiel

Man kann ein Quadrat wie folgt konstruieren:

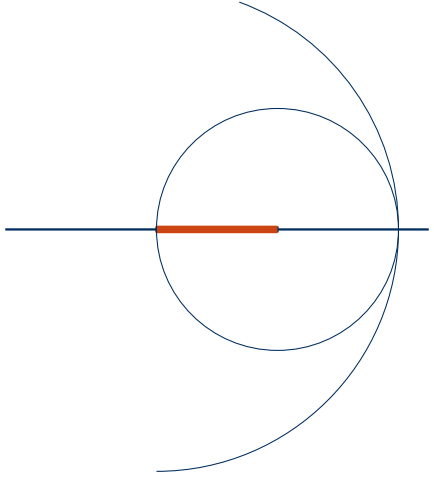
Beispiel

Man kann ein Quadrat wie folgt konstruieren:



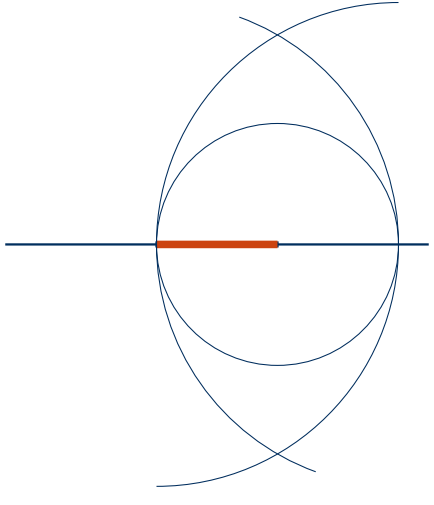
Beispiel

Man kann ein Quadrat wie folgt konstruieren:



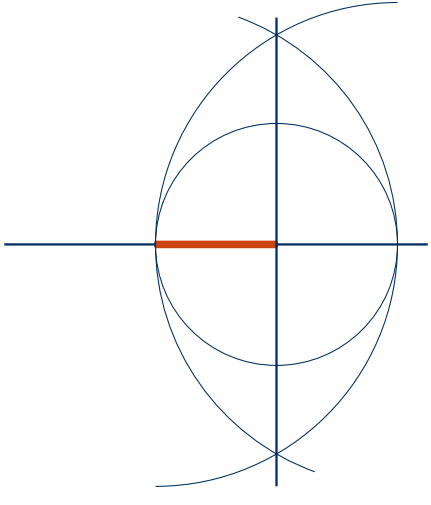
Beispiel

Man kann ein Quadrat wie folgt konstruieren:



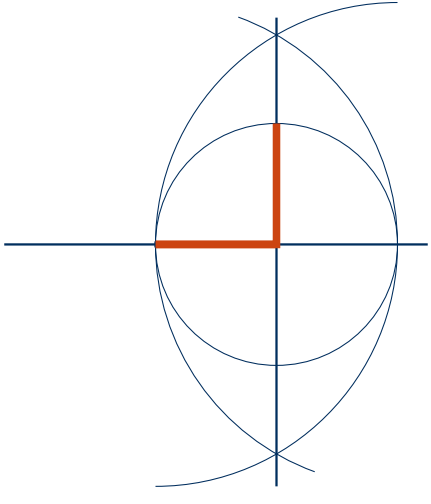
Beispiel

Man kann ein Quadrat wie folgt konstruieren:



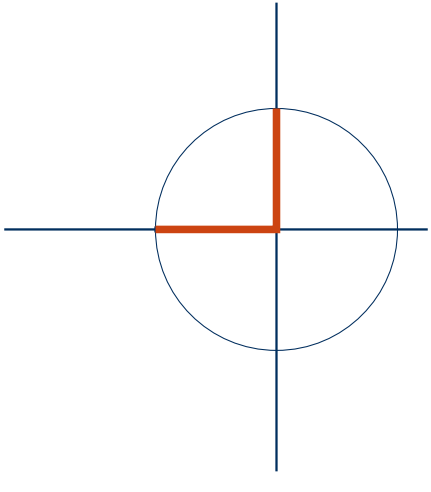
Beispiel

Man kann ein Quadrat wie folgt konstruieren:



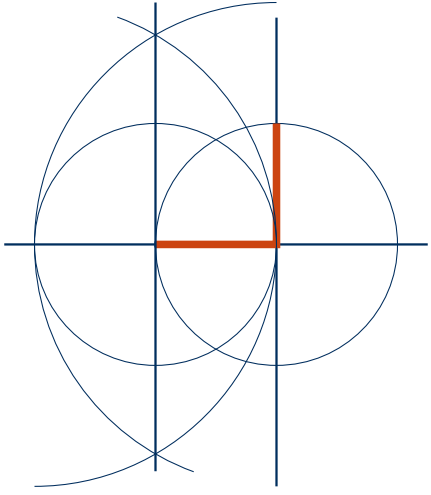
Beispiel

Man kann ein Quadrat wie folgt konstruieren:



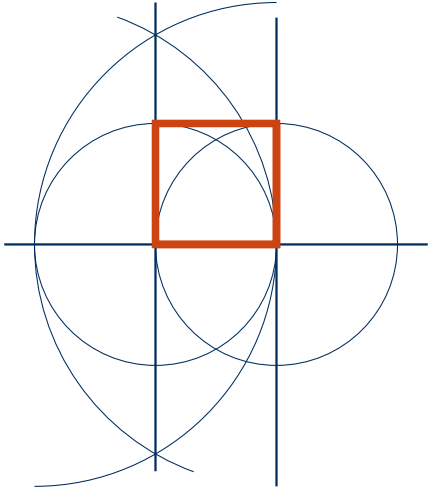
Beispiel

Man kann ein Quadrat wie folgt konstruieren:



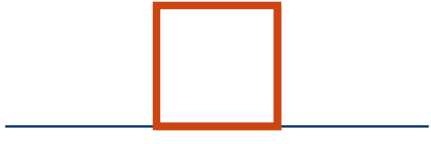
Beispiel

Man kann ein Quadrat wie folgt konstruieren:



Beispiel

Man kann ein Quadrat wie folgt konstruieren:



Weitere Konstruktionsbeispiele

Es lassen sich zahlreiche weitere Konstruktionen durchführen, z.B.:

- Halbierung eines Winkels
- Konstruktion des regelmäßigen Sechsecks
- Konstruktion eines flächengleichen Quadrates aus einem gegebenen Rechteck

Weitere Konstruktionsbeispiele

Es lassen sich zahlreiche weitere Konstruktionen durchführen, z.B.:

- Halbierung eines Winkels
- Konstruktion des regelmäßigen Sechsecks
- Konstruktion eines flächengleichen Quadrates aus einem gegebenen Rechteck
- Konstruktion des regelmäßigen 17-Ecks
  (entdeckt von Gauss – „Durch angestrengtes Nachdenken . . . am Morgen . . . (ehe ich aus dem Bette aufgestanden war)“)

Weitere Konstruktionsbeispiele

Es lassen sich zahlreiche weitere Konstruktionen durchführen, z.B.:

- Halbierung eines Winkels
- Konstruktion des regelmäßigen Sechsecks
- Konstruktion eines flächengleichen Quadrates aus einem gegebenen Rechteck
- Konstruktion des regelmäßigen 17-Ecks
  (entdeckt von Gauss – „Durch angestrengtes Nachdenken . . . am Morgen . . . (ehe ich aus dem Bette aufgestanden war)“)
- Konstruktion des regelmäßigen 65537-Ecks (Hermes)
- . . .

Rechnen mit Euklid

1637: René Descartes publiziert die Idee des Koordinatensystems

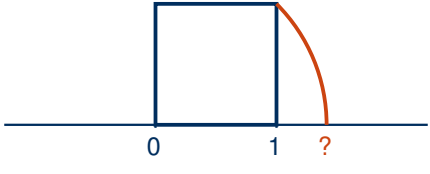
$\rightsquigarrow$ Geometrie wird numerisch!

Rechnen mit Euklid

1637: René Descartes publiziert die Idee des Koordinatensystems

$\rightsquigarrow$ Geometrie wird numerisch!

Beispiel: Beginnend mit Punkten an den Koordinaten $(0, 0)$ und $(1, 0)$ können wir einen neuen Punkt konstruieren:

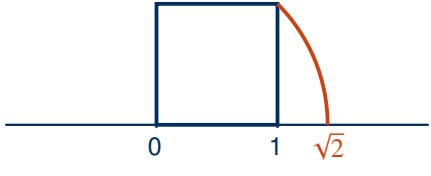


Rechnen mit Euklid

1637: René Descartes publiziert die Idee des Koordinatensystems

$\rightsquigarrow$ Geometrie wird numerisch!

Beispiel: Beginnend mit Punkten an den Koordinaten $(0, 0)$ und $(1, 0)$ können wir einen neuen Punkt konstruieren:



Wir haben also $\sqrt{2}$ „berechnet“!

Was kann dieser „Rechner“?

Was kann dieser „Rechner“?

Man kann Geometrie durch Gleichungen darstellen:

- Gerade durch Punkte (a, b) und (c, d):
  $y = \frac{d-b}{c-a}x + \frac{bc-da}{c-a}$
- Kreis um Mittelpunkt (a, b) durch Punkt (c, d): $(x-a)^2 + (y-b)^2 = (a-c)^2 + (b-d)^2$

Zeichnen = Systeme solcher Gleichungen grafisch Lösen

Was kann dieser „Rechner“?

Man kann Geometrie durch Gleichungen darstellen:

- Gerade durch Punkte (a, b) und (c, d):
  $y = \frac{d-b}{c-a}x + \frac{bc-da}{c-a}$
- Kreis um Mittelpunkt (a, b) durch Punkt (c, d): $(x-a)^2 + (y-b)^2 = (a-c)^2 + (b-d)^2$

Zeichnen = Systeme solcher Gleichungen grafisch Lösen

Es stellt sich heraus: Alle so konstruierbaren Zahlen ergeben sich mit folgenden Rechnungen:

- Addition und Subtraktion
- Multiplikation und Division
- Ziehen der Quadratwurzel

Was kann dieser „Rechner“?

Man kann Geometrie durch Gleichungen darstellen:

- Gerade durch Punkte (a, b) und (c, d):
  $y = \frac{d-b}{c-a}x + \frac{bc-da}{c-a}$
- Kreis um Mittelpunkt (a, b) durch Punkt (c, d): $(x-a)^2 + (y-b)^2 = (a-c)^2 + (b-d)^2$

Zeichnen = Systeme solcher Gleichungen grafisch Lösen

Es stellt sich heraus: Alle so konstruierbaren Zahlen ergeben sich mit folgenden Rechnungen:

- Addition und Subtraktion
- Multiplikation und Division
- Ziehen der Quadratwurzel

$\rightsquigarrow$ Unmöglich („euklidisch unberechenbar“) sind zum Beispiel die Konstruktion von π („Quadratur des Kreises“) und die Berechnung von Kubikwurzeln („Verdoppelung des Würfels“)

Euklid statt Turing?

Liefert uns das eine alternatives Berechenbarkeitsmodell?

Euklid statt Turing?

Liefert uns das eine alternatives Berechenbarkeitsmodell?

Vermutlich nicht:

- Exaktes Zeichnen ist nicht physisch implementierbar (es gibt z.B. keinen perfekten Kreis)
- Die Ergebnisse sind nicht exakt ablesbar (Messfehler)

Euklid statt Turing?

Liefert uns das eine alternatives Berechenbarkeitsmodell?

Vermutlich nicht:

- Exaktes Zeichnen ist nicht physisch implementierbar
  (es gibt z.B. keinen perfekten Kreis)
- Die Ergebnisse sind nicht exakt ablesbar (Messfehler)

Dennoch illustriert das wichtige Ideen der Informatik:

> Informatik erforscht, was Computer sind
> und welche Probleme man mit ihnen lösen kann.

Man sollte trotz Church-Turing immer neu fragen, was Rechnen noch sein kann ...

Zusammenfassung

Sprachen und Berechnung

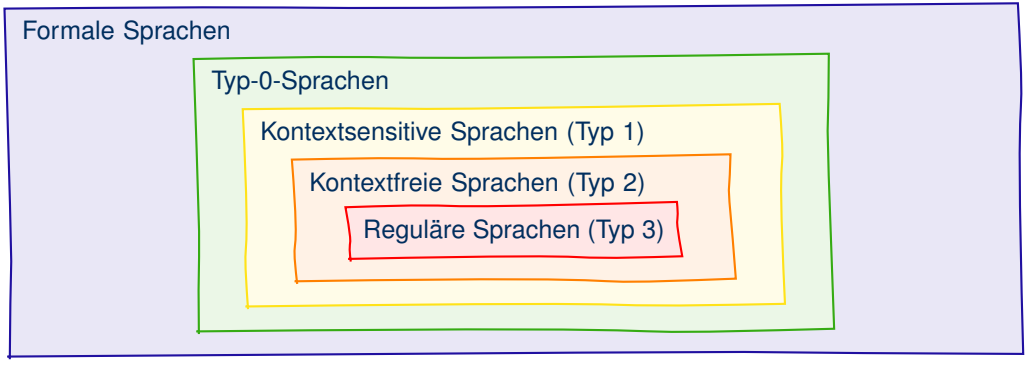
- Formale Wörter als allgemeine Abstraktion aller Daten, die in Computern verarbeitet werden können
- Formale Sprachen als Mengen von Ein- oder Ausgaben
- Worterkennung als allgemeine Berechnungsaufgabe

Beispiel: Auch die Berechnung von Funktionen kann als Wortproblem ausgedrückt werden. Anstatt zu fragen, „Was ergibt $n + m$?“ kann man fragen „Ist $n + m = r$?“

Daraus ergibt sich ein Kernthema dieser Vorlesung:

Sprachen zu klassifizieren heißt Rechenaufgaben klassifizieren

Die Chomsky-Hierarchie



Trennung der Sprachklassen

Die Chomsky-Hierarchie ist echt. Methoden, um Nichtenthaltensein einer Sprache in einer bestimmten Hierarchieebene zu zeigen:

- Typ 3: reguläres Pumping-Lemma, Myhill-Nerode-Index, Abschlusseigenschaften (V10)
- Typ 2: kontextfreies Pumping-Lemma (V13), Abschlusseigenschaften (V14)
- det. Typ 2: Abschlusseigenschaften (V16)
- Typ 1: Entscheidbarkeit, Abschlusseigenschaften (V19/V20)
- Typ 0: Semi-Entscheidbarkeit (V20, V25)

Übersicht Abschlusseigenschaften

	Abschluss unter . . .					
Sprache	$\cap$	$\cup$	$\overline{}$	$\circ$	*	Automat
Typ 0	✓	✓	×	✓	✓	TM (DTM/NTM)
Typ 1	✓	✓	✓	✓	✓	LBA ($\stackrel{?}{=}$ det. LBA)
Typ 2	×	✓	×	✓	✓	PDA
Det. Typ 2	×	×	✓	×	×	DPDA
Typ 3	✓	✓	✓	✓	✓	DFA/NFA

Übersicht Abschlusseigenschaften

	Ergebnis von Typ ...					
Sprache	$\cap$	$\cup$	$\overline{}$	$\circ$	*	Automat
Typ 0	0	0	–	0	0	TM (DTM/NTM)
Typ 1	1	1	1	1	1	LBA ($\stackrel{?}{=}$ det. LBA)
Typ 2	1	2	1	2	2	PDA
Det. Typ 2	1	2	D2	2	2	DPDA
Typ 3	3	3	3	3	3	DFA/NFA

Übersicht Probleme

Die Entscheidbarkeit verschiedener relevanter Probleme ist je nach Sprachklasse unterschiedlich:

Sprache	Wortproblem	Leerheit	Äquivalenz	Regularität	Inklusion	Schnitt
Typ 0	×	×	×	×	×	×
Typ 1	✓	×	×	×	×	×
Typ 2	✓	✓	×	×	×	×
Det. Typ 2	✓	✓	✓	✓	×	×
Typ 3	✓	✓	✓	(✓)	✓	✓

Wortprobleme lösen

Wie schwer ist es, dass Wortproblem zu lösen, wenn die Eingabe eine (geeignet kodierte) Sprache und ein zu testendes Wort ist?

Sprache	Zeitkomplexität bzgl. Wortlänge n
Typ 3	$O(n)$ (Abarbeitung DFA)
Det. Typ 2	$O(n)$ (Abarbeitung DPDA)
Typ 2	$O(n^3)$ (CYK-Algorithmus)
Typ 1	$O(n \cdot \lvert\Gamma\rvert^n)$ (z.B. über LBA-Konfigurationsgraph)
Typ 0	unentscheidbar

Das Wortproblem für Typ 1 ist PSpace-vollständig. Es ist nicht bewiesen aber wahrscheinlich, dass es keine subexponentiellen Algorithmen gibt.

Nichtdeterminismus

Nichtdeterministische Akzeptanzbedingung:
Gibt es mindestens einen Lauf, der akzeptiert?

Det. Automatenmodell	Nichtdet. Automatenmodell	äquivalent?
DFA	NFA	✓
DPDA	PDA	×
DLBA	LBA	?
DTM	NTM	✓

- Oft ist es schwierig, Nichtdeterminismus unter Ressourcenbeschränkungen aufzulösen, nicht nur bei LBAs
  z.B. ist auch $P \neq NP$ offen
- Wegen der Asymmetrie hat jede nichtdeterministische Klasse eine Komplementärklasse, die oft (vermutlich) unterschiedlich ist (z.B. coNP vs. NP)

$\rightsquigarrow$ mehr dazu in Theoretische Informatik und Logik im Sommersemester

Ausblick und Anwendungen

Formale Sprachen

Formale Sprachen in der Praxis:

- Typ 3: extrem weit verbreitet in Form von regulären Ausdrücken; im Kompilerbau als Lexer; noch einfachere Sprachen bei Anfrage/Auswahlmechanismen z.B. CSS-Selektoren
- det. Typ 2: besonders relevant im Kompilerbau (LR(k)-Grammatiken)
- nichtdet. Typ 2: in der Sprachverarbeitung; in dieser Anwendung teils auch etwas stärkere Sprachklassen (z.B. Tree-Adjoining Grammars)

Typ-1-Sprachen haben kaum praktische Anwendungen, Typ-0-Sprachen fallen mit allgemeinen TMs zusammen

Automatentheorie

Es gibt viele Automatenmodelle jenseits der hier vorgestellten:

- Baumautomaten arbeiten auf Baumstrukturen, die sie von oben oder unten her lesen
- Automaten für unendliche Strukturen verwenden andere Akzeptanzbedingungen, die für unendliche Abarbeitungen Sinn ergeben
- Hybride Automaten modellieren komplexe dynamische Systeme mithilfe von Differentialgleichungen
- Eingeschränkte Automatenmodelle z.B. partiell geordnete Automaten, erkennen spezielle reguläre Sprachen
- …

Wesentliche Anwendungen von Automaten:

- Definition „interessanter“ Sprachklassen
- Lösung algorithmischer Probleme (z.B. Inklusionstest von Sprachen)

Berechenbarkeitstheorie

Klassifikation unentscheidbarer Probleme, alternative Berechnungsmodelle
$\leadsto$ wichtig für das Grundverständnis von Berechnung in der Informatik

Praktisch bedeutsamer ist die Struktur der entscheidbaren Probleme, die wir im Sommersemester näher betrachten: Komplexitätstheorie

1. Eigenständiges Forschungsgebiet, das sich vielen grundlegenden Fragen widmet (einschl. P $\neq$ NP?); Theorie der Kryptographie; Quantenkomplexität
2. Methoden für andere Forschungsfelder, welche die komplexitätstheoretische Analyse von Problemen in vielen Fachgebieten ermöglichen; Themen wie parametrisierte Komplexität oder Ausgabekomplexität sind aus Anwendungen motiviert

Zusammenfassung

Formale Sprachen sind die Grundlage zahlreicher Forschungs- und Anwendungsfelder der Informatik.

Berechnungsmodelle erlauben uns, allgemeine Aussagen über die Schwere und Lösbarkeit von Berechnungsaufgaben zu treffen

Was erwartet uns als nächstes?

- Probeklausur und Repetitorium
- Klausurvorbereitung
- Prüfung