



Achtung: Vorlesung in der nächsten Woche,
am 28.10. um **12.15** in **P72.01** !!!

Am kommenden Freitag findet die Zentralübung statt.

Themen der Vorlesung



- Berechenbarkeit
- Automaten und Formale Sprachen
- Komplexitätstheorie

Was ist ein Algorithmus?

Welche Funktionen sind berechenbar?

Welche nicht?

Wieviel Speicherplatz / Rechenzeit benötigt man zur Berechnung einer (berechenbaren) Funktion?

höchstens?

Effizienten Algorithmus angeben,
vgl. DuA

mindestens?

Untere Schranke beweisen, Funktionen klassifizieren nach ihrer Komplexität

- Wie beschreibt man Sprachen $L \subseteq \Sigma^*$?
→ Grammatiken
- Wie entscheidet man dann: " $x \in L$?"
(Wortproblem, Kern der Syntaxanalyse)
→ Automaten

- Wie beschreibt man Semantik?
- Wie verifiziert man Korrektheit von Programmen?
 - etwas in Modellierung, Hauptstudium.

Drei grundlegende Techniken



- Simulation
- Reduktion
- Diagonalisierung

- ein Verfahren, wie ein in einem Formalismus beschriebenes Programm durch ein in einem anderen Formalismus beschriebenes Programm ersetzt werden kann, so dass das Ein/Ausgabeverhalten unverändert bleibt.
- **Bsp:** Simulation eines Programms, in dem FOR-Schleifen erlaubt sind, durch eins, in dem nur WHILE-Schleifen erlaubt sind.

LOOP-Programme: nur For-Schleifen erlaubt, keine bedingten Sprünge, keine Prozeduren etc.

WHILE-Programme: auch WHILE-Schleifen erlaubt (und auch Prozeduren,.....)

WHILE-Programme beschreiben ein *universelles Rechenmodell*.

- **Jedes LOOP-Programm kann durch ein WHILE-Programm simuliert werden.**
- Umgekehrt??

Die folgende Funktion ack: $\mathbb{N} \times \mathbb{N}$ kann durch WHILE- aber nicht durch LOOP-Programme berechnet werden.

$$\text{ack}(0, m) = m + 1$$

$$\text{ack}(n, 0) = \text{ack}(n-1, 1), \text{ falls } n \geq 1,$$

$$\text{ack}(n, m) = \text{ack}(n-1, \text{ack}(n, m-1)), \text{ falls } n, m \geq 1$$

ack wächst sehr schnell !!

geht zurück auf Cantor'sche Diagonalisierungsverfahren,
eng verwandt mit den Paradoxien z. B. von Russel.

**Es gibt kein Programm HALTEN, das bei Eingabe eines
Programms P und einer Eingabe x für P entscheidet, ob P
gestartet mit x anhält.**

→ **Das Halteproblem ist nicht entscheidbar.**

Beweis durch Widerspruch:

Annahme: Es gibt Programm „*Halten*(P, x)“, das bei Eingabe P, x entscheidet, ob P gestartet mit x hält.

Dann gibt es auch folgendes Programm:

Widerspruch(P)

Falls *Halten*(P, P) „ja“ liefert, gehe in Endlosschleife, sonst halte an.

Eine Möglichkeit, formal zu beschreiben, was es bedeutet „Problem A ist nicht schwerer als Problem B“:

→

A ist ein schwieriges Problem, aber einfach zu lösen, falls wir einen Algorithmus für B haben.

A: Cliquesproblem (CLIQUE)

Eingabe: Graph G , Zahl k ,

Gibt es einen vollständigen Subgraph der Größe k in G ?

B: Independent Set Problem (IS)

Eingabe: Graph G , Zahl k ;

Gibt es einen leeren induzierten Subgraph der Größe k in G ?

CLIQUE kann auf IS reduziert werden!

Formalisierungen des Begriffs „Algorithmus“



- Programmiersprache mit vollständiger, formaler Syntax- und Semantik-Beschreibung, z.B. Java, C, Pascal, Algol
 - zu kompliziert, um formal zu argumentieren –
- einfache Rechenmodelle oder Kalküle, die „das gleiche können wie z. B. Java-Programme“
 - Turingmaschinen, Registermaschinen,
 μ -Rekursivität, uniforme Schaltkreisfamilien, ...



Vorlage für eine Folie im Querformat

Überschrift 2





***Vorlage für eine
Folie im
Hochformat
Überschrift 2***

