

# Grundbegriffe der Theoretischen Informatik

Sommersemester 2017 - Beate Bollig

Die Folien basieren auf den Materialien von Thomas Schwentick.

0: Einleitung

# Inhalt

## ▷ **0.1 Was ist Theoretische Informatik?**

0.2 Themen der GTI-Vorlesung

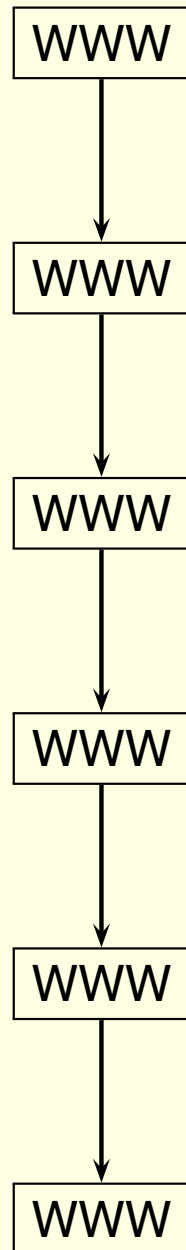
0.3 Literatur


0.4 Organisatorisches

# Warum Theorie der Informatik helfen kann

- 2013 in der Presse: <http://heise.de/-1803813>
- Um auf die Kontaktliste eines gesperrten iPhones zuzugreifen, konnte man
  - es einschalten
  - den „Entsperren“-Slider nach rechts ziehen
  - auf „Notruf“ tippen
  - auf den Ausschalt-Knopf drücken, bis die Ausschalt-Option angezeigt wird
  - „Abbrechen“ klicken
  - eine Notrufnummer wählen, verbinden und sofort wieder auflegen
  - das Gerät aus- und einschalten
  - den „Entsperren“-Slider ziehen
  - den Ausschaltknopf gedrückt halten
  - kurz bevor die Ausschalt-Option angezeigt wird, auf „Notruf“ drücken
- Die Kontakte öffnen sich und bleiben geöffnet, solange der Ausschaltknopf gedrückt bleibt
- Besser wäre es, beweisen zu können, dass es eine solche Lücke nicht gibt...

# Wichtige Themen der Informatik – sehr grob



- Schritte auf dem Weg von der Programmidee zum Programm
    - ... und zugehörige Lehrveranstaltungen  ganz grob
  - Spezifikation:
    - Software-Technik, DAP I, Formale Methoden des System-Entwurfs
  - Wahl geeigneter Algorithmen:
    - DAP II, Effiziente Algorithmen, Komplexitätstheorie
  - Umsetzung der Algorithmen in ein Programm:
    - DAP I, Software-Technik, SoPra
  - Übersetzung des Programms:
    - Übersetzerbau
  - Ausführung des Programms:
    - Rechnerstrukturen, Betriebssysteme, HaPra
- Die GTI bezieht sich auf mehrere dieser Schritte

# Wunschzettel der Informatiker

- Wünsche für die einfache Softwareerstellung
  - Zu jeder Spezifikation sollte es einen Algorithmus und ein Programm geben
  - Programme sollten aus der Spezifikation automatisch erzeugt werden
  - Es sollte automatisch überprüft werden können, ob ein Programm seiner Spezifikation entspricht
  - Programme sollten möglichst klein sein
  - Programme sollten möglichst schnell sein
  - Die Übersetzung von Programmen in Maschinenprogramme sollte automatisch erfolgen
- In dieser Vorlesung werden wir Grenzen für die Erfüllung dieser Wünsche kennen lernen
- Diese Grenzen waren unter den ersten Erkenntnissen der Theoretischen Informatik — bevor es mit der praktischen Informatik so richtig los ging
- Aber was ist überhaupt Theoretische Informatik?

# Was ist Theoretische Informatik? (1/2)

- Was ist Theoretische Informatik?
- Die Antwort auf diese Frage hat zwei Komponenten:
  - die Methoden der Theoretischen Informatik
  - die Inhalte der Theoretischen Informatik
- Die Theoretische Informatik verwendet die **mathematische Methode** für die **Grundlagen der Informatik**
- Ganz grob lässt sich das durch die folgende „Gleichung“ ausdrücken:

$$\frac{\text{Theoretische Informatik}}{\text{Informatik}} \approx \frac{\text{Mathematik}}{\text{Physik}}$$

# Methodik der Theoretischen Informatik

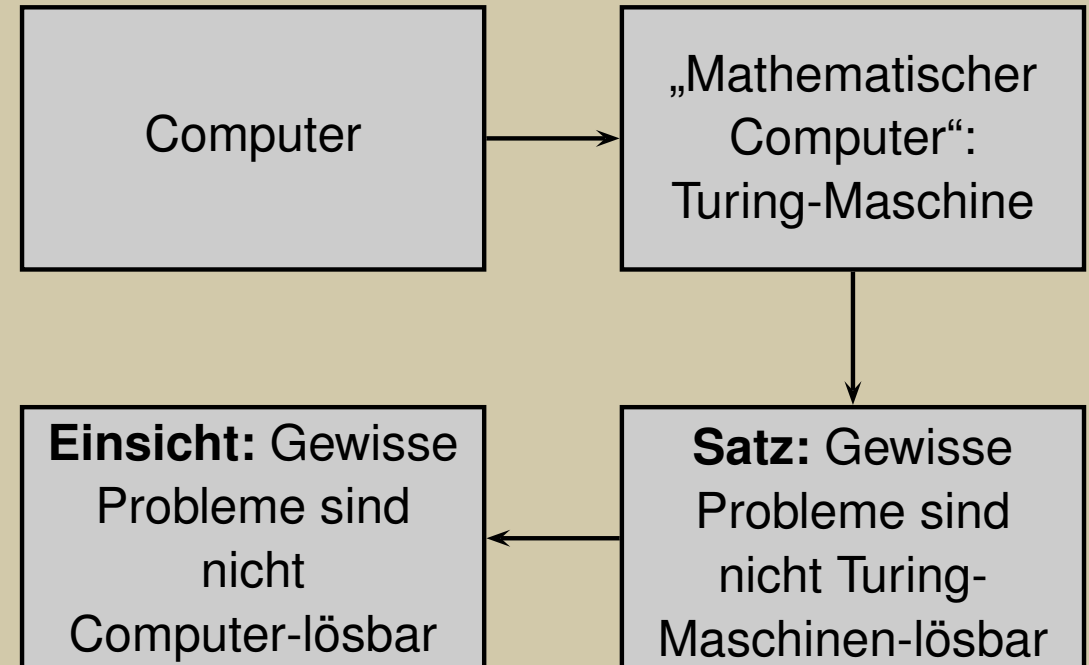
- **Typische Vorgehensweise**

- Modellierung von im Zusammenhang mit Computern und Berechnungen auftretenden Phänomenen durch mathematische Objekte
- Formulierung von Aussagen über diese Objekte
- Beweis dieser Aussagen

- Warum ist der mathematisch präzise Ansatz (inklusive Beweisen) für die Informatik sinnvoll?

- Die Semantik von Spezifikationen sollte präzise definierbar und beispielsweise die Äquivalenz von Spezifikation und Modell beweisbar sein
- Auch die Semantik von Algorithmen/Programmen sollte präzise definierbar sein und Korrektheits- und Aufwandsaussagen sollten sich beweisen lassen

## Beispiel



# Was ist Theoretische Informatik? (2/2)

Artikel

Diskussion

Lesen

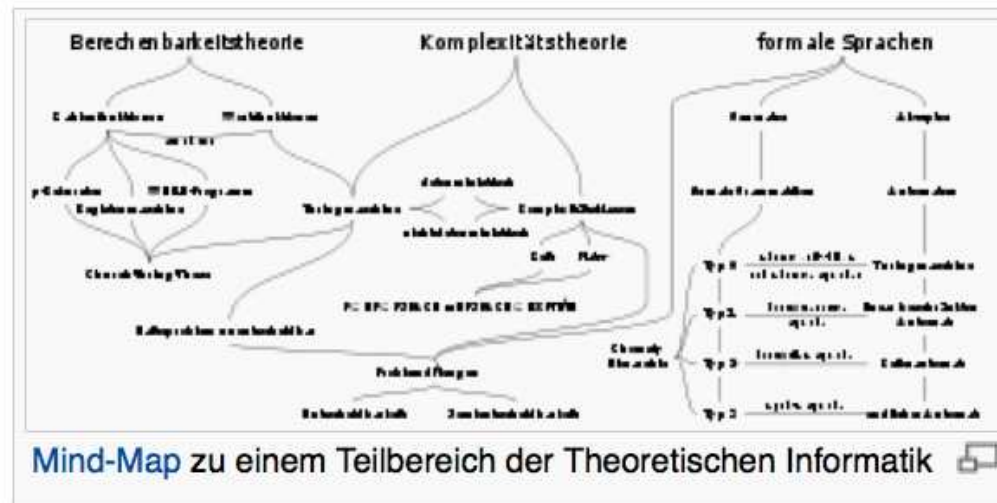
Bearbeiten

Suche



## Theoretische Informatik

Die **Theoretische Informatik** beschäftigt sich mit der Abstraktion, Modellbildung und grundlegenden Fragestellungen, die mit der Struktur, Verarbeitung, Übertragung und Wiedergabe von **Informationen** in



Mind-Map zu einem Teilbereich der Theoretischen Informatik

Zusammenhang stehen. Ihre Inhalte sind **Automatentheorie**, Theorie der **formalen Sprachen**, **Berechenbarkeits-** und **Komplexitätstheorie**, aber auch **Logik** und **formale Semantik** sowie die **Informations-**, **Algorithmen-** und **Datenbanktheorie**.

©Wikipedia

- Wir werfen nun einen Blick auf die Inhalte der Theoretischen Informatik



# Teilgebiete der Theoretischen Informatik (1/3)

- **Berechenbarkeit: „Was ist überhaupt möglich?“**
  - Was ist ein Algorithmus?
  - Welche Probleme lassen sich algorithmisch lösen?
  - Wie lassen sich die algorithmisch nicht lösbaren Probleme klassifizieren?
- **Komplexitätstheorie: „Was ist effizient möglich?“**
  - Klassifikation algorithmischer Probleme nach ihrer prinzipiellen Schwierigkeit
  - Vergleich der dabei entstehenden Klassen
  - Besondere Berechnungsmodi: zufallsgesteuert, parallel, nicht-deterministisch
  - **P** vs. **NP**
- **Effiziente Algorithmen: „Wie geht es am schnellsten?“**
  - Effiziente Algorithmen für konkrete Probleme
  - Datenstrukturen
  - Algorithmische Geometrie, Graph-Algorithmen, ...
  - Parallele Algorithmen, Online-Algorithmen, ...

## Teilgebiete der Theoretischen Informatik (2/3)

- **Formale Sprachen: „Wie sieht ein Programm aus?“**
  - Wie lässt sich die syntaktische Struktur von Computer-Programmen beschreiben?
  - Reguläre Sprachen
  - Kontextfreie Sprachen
  - Sprachen von unendlichen Zeichenketten, Bäumen etc.
- **Semantik: „Was bedeutet ein Programm?“**
  - Wie lässt sich die Bedeutung/Wirkung von Computer-Programmen so formalisieren, dass z.B. ihre Korrektheit bewiesen werden kann?
  - Semantik von Programmiersprachen
  - Funktionale und logische Programmiersprachen
  - Model Checking
- **Viele weitere Gebiete:**
  - Theorie verteilter Systeme
  - Datenbanktheorie
  - ...

## Teilgebiete der Theoretischen Informatik (3/3)

- Diese Aufzählung der Teilgebiete der Theoretischen Informatik ist nicht vollständig
- Die **Special Interest Group on Algorithms and Computation Theory (SIGACT)** der **Association for Computing Machinery (ACM)** schreibt dazu:
  - „TCS covers a wide variety of topics including algorithms, data structures, computational complexity, parallel and distributed computation, probabilistic computation, quantum computation, automata theory, information theory, cryptography, program semantics and verification, machine learning, computational biology, computational economics, computational geometry, and computational number theory and algebra“
- Die GTI-Vorlesung behandelt hiervon nur einen kleinen Ausschnitt
  - Allerdings ist die Stoffauswahl ziemlich kanonisch und unterscheidet sich von Uni zu Uni meist nur wenig
  - Darüber hinaus werden viele der hier behandelten Grundbegriffe auch in den anderen Bereichen der Theoretischen Informatik und in vielen angewandten Teilen der Informatik verwendet

# Inhalt

0.1 Was ist Theoretische Informatik?

▷ **0.2 Themen der GTI-Vorlesung**

0.3 Literatur

0.4 Organisatorisches

# Die 4 Themen der GTI-Vorlesung - A: Reguläre Sprachen

- Die **regulären Sprachen** bilden eine Klasse von einfachen Sprachen mit
  - vielen Anwendungen
  - vielen äquivalenten Charakterisierungen
  - weiteren angenehmen Eigenschaften

- **Beispiele für Anwendungen**
  - Bezeichner in Programmiersprachen
  - Verifikation zustandsbasierter Systeme
  - Schemasprachen für XML
  - Zeichenkettensuche (grep)
  - Spezifikation zulässiger Eingaben für Web-Formulare

- **Viele Charakterisierungen**
  - Reguläre Ausdrücke
  - Endliche Automaten
    - \* deterministisch oder nicht-deterministisch
    - \* 1-Weg oder 2-Weg
    - \* mit oder ohne  $\epsilon$ -Übergänge
  - Grammatiken, logische Formeln, ...
- **Angenehme Eigenschaften**
  - einfache Spezifikationssprache
    - reguläre Ausdrücke
  - einfache Testalgorithmen
    - endliche Automaten
  - automatische Umwandlung von Spezifikationen in Testalgorithmen
  - optimaler Testalgorithmus konstruierbar
  - Methode zum Testen, ob Spezifikationen und Programme äquivalent sind
  - Methoden zum Nachweis, dass eine Sprache nicht regulär ist

# Die 4 Themen der GTI-Vorlesung - B: Kontextfreie Sprachen

- Die **kontextfreien Sprachen** bilden eine weitere Klasse einfacher Sprachen:
  - ausdrucksstärker als die regulären Sprachen
  - auch viele Charakterisierungen
  - angenehme Eigenschaften, aber nicht mehr ganz so viele
- Hauptanwendung: Syntaktische Struktur von Programmiersprachen

- **Charakterisierungen**
  - Spezifikationssprache: Grammatiken, bzw. Syntaxdiagramme
  - Algorithmen: Kellerautomaten (richtig effizient nur für Teilklassen)
- **Angenehme Eigenschaften**
  - automatische Umwandlung von Spezifikation in Testalgorithmus
  - Methode zum Nachweis, dass eine Sprache nicht kontextfrei ist

# Die 4 Themen der GTI-Vorlesung - C: Berechenbarkeit

- **Hauptfragen**

- Was heißt „berechenbar“?
- Gibt es algorithmische Probleme, die nicht berechenbar sind?
- Falls ja, wie sehen solche Probleme aus?

- **Was heißt berechenbar?**

- Es gibt viele Modelle, die den Begriff eines Algorithmus und damit Berechenbarkeit formalisieren
- Praktisch alle sind äquivalent!
- Church-Turing-These: Diese Modelle definieren Berechenbarkeit
- Wir werden kennen lernen:
  - \* Turing-Maschinen
  - \* WHILE-Programme
  - \* Rekursive Funktionen

- **Und was ist nicht berechenbar?**

- Wir werden sehen: es gibt viele, auch praktisch relevante Probleme, die sich algorithmisch nicht lösen lassen

# Die 4 Themen der GTI-Vorlesung - D: Komplexitätstheorie

- **Hauptfragen**

- Was heißt „effizient berechenbar“?
- Gibt es algorithmische Probleme, die berechenbar, aber nicht effizient berechenbar sind?
- Falls ja, wie sehen solche Probleme aus?

- **Was heißt effizient berechenbar?**

- Weitgehender Konsens:
  - \* Rechenzeit wächst höchstens polynomiell in Größe der Eingabe
- Die genannten Berechnungsmodelle sind auch in dieser Hinsicht äquivalent
- Komplexitätsklasse: **P**

- **Wie sehen nicht effizient berechenbare Probleme aus?**

- Es gibt tausende von praktisch relevanten Problemen, für die
  - \* kein effizienter Algorithmus bekannt ist
  - \* kein Beweis, dass sie nicht effizient berechenbar sind, bekannt ist
- Die meisten davon sind im folgenden Sinne äquivalent:
  - \* Hat eines einen effizienten Algorithmus, so haben alle einen
  - \* Hat eines keinen effizienten Algorithmus, so hat keines einen
- **NP**-vollständige Probleme
- **P = NP**-Frage



# Sichere Systeme

- Das iPhone-Beispiel illustriert, dass sichere Systeme ein wichtiges Thema für die Informatik darstellen
  - Sie werden uns als Querschnittsthema in mehreren Teilen der Vorlesung beschäftigen
  - Meistens in der Form eines algorithmischen Problems dieser Art:
    - Gegeben: ein Modell  $S$  eines Systems und eine formale Beschreibung  $E$  einer Eigenschaft
    - Lässt sich automatisch überprüfen, ob  $S$  die Eigenschaft  $E$  hat?
- Beispiele:
    - $S$  ein Smartphone-OS,  
 $E$ : „Kein Zugang ohne PIN möglich“
    - $S$  ein Programm,  
 $E$ : „ $S$  terminiert für jede Eingabe“
    - $S$  ein Kommunikationsprotokoll,  
 $E$ : „ $S$  führt niemals zu einer Verklemmung“
    - $S$  ein Schaltkreis,  
 $E$  eine Boolesche Funktion, die  $S$  berechnen soll
- Wir werden Fälle sehen, in denen eine solche automatische Überprüfung möglich ist, aber auch andere...

# Was sollen Sie in GTI lernen?

- **Erkenntnisse**

- Grenzen der Möglichkeiten von Computern: nicht berechenbare Probleme
- Praktische Grenzen von Computern: nicht effizient lösbare Probleme
- ...

- **Fähigkeiten**

- Abstraktion
- Formalisieren
- Analyse, z.B. Erkennen von schwierigen Berechnungsproblemen
- ...

- **Grundwissen & Handwerkszeug**

- Formale Grundlagen des Rechnens mit Computern
- Umgang mit regulären Sprachen, regulären Ausdrücken und Automaten
- Reguläre Sprachen / endliche Automaten „erkennen“
- Theoretische Grundlagen für Übersetzer
- ...

# Das Semester im Überblick

## 0: Einleitung

### A: Reguläre Sprachen

- 1: Reguläre Ausdrücke
- 2: Endliche Automaten
- 3: Äquivalenz der Modelle
- 4: Minimierung endlicher Automaten
- 5: Abschlusseigenschaften, Grenzen und Algorithmen
- 6: Anwendungen regulärer Sprachen

### B: Kontextfreie Sprachen

- 7: Kontextfreie Grammatiken
- 8: Normalformen und Korrektheitsbeweise
- 9: Kellerautomaten
- 10: Pumping-Lemma, Algorithmen und Abschlusseigenschaften
- 11: Wortproblem und Syntaxanalyse

### C: Berechenbarkeit

- 12: Verschiedene Berechnungsmodelle
- 13: Die Church-Turing-These
- 14: Unentscheidbare Probleme 1
- 15: Unentscheidbare Probleme 2
- 16: Varianten, Einschränkungen und Erweiterungen

### D: Komplexitätstheorie

- 17: Polynomielle Zeit
- 18: **NP** und **NP**-Vollständigkeit
- 19: **NP**-vollständige Probleme
- 20: **NP**: Weitere Erkenntnisse
- 21: Zufallsbasierte Algorithmen
- 22: Zufallsbasierte Komplexitätsklassen
- 23: Parametrisierte Algorithmen und Komplexität

Die Kapitelstruktur kann sich noch verändern

# Inhalt

0.1 Was ist Theoretische Informatik?

0.2 Themen der GTI-Vorlesung

▷ **0.3 Literatur**

0.4 Organisatorisches

# Literatur

- Hopcroft, E., Motwani, R. und Ullman, J.D. (2002). Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie. Pearson.
  - umfassend, viele Beispiele, gut aufgebaut, verständlich
- Wegener, I. (1999). Theoretische Informatik - eine algorithmenorientierte Einführung. Teubner.
  - gut aufgebaut, verständlich, mittlerer Umfang
- Wegener, I. (1996). Kompendium der Theoretischen Informatik - eine Ideensammlung. Teubner.
  - Ergänzung (deckt Vorlesungsinhalt nicht vollständig ab), sehr verständlich
- Wegener, I. (2003). Komplexitätstheorie - Grenzen der Effizienz von Algorithmen. Springer.
  - verständlich für den vierten Teil der Vorlesung
- Schöning, U. (1997). Theoretische Informatik kurz gefasst. Spektrum.
  - gut aufgebaut, sehr verständlich, kürzer als die anderen, nicht so umfassend

# Inhalt

0.1 Was ist Theoretische Informatik?

0.2 Themen der GTI-Vorlesung

0.3 Literatur

▷ **0.4 Organisatorisches**

# Die Bestandteile der Veranstaltung

- Vorlesung
- Übungsaufgaben
- Übung
- GTI HelpDesk (Lernraumbetreuung)
- Tutorien
- Prüfungen
- GTI im Web

# Vorlesung

- **Termine**

- Dienstag, 10:15 – 11:45 Uhr, HG II, HS 3
- Donnerstag, 12:15 – 13:45 Uhr, HG II, HS 3

- Durchgängiger Einsatz von **Folien**

- Erläuterungen an der Tafel
- Die Folien können über moodle geladen werden

- Kapitel 1: 20.4.17

- **Vorausgesetzte Kenntnisse**

- U.a. gründliche Kenntnisse von DAP 2

- **Zweck der Vorlesung**

- Vermittlung aller wesentlichen Inhalte

- **Gebrauchsanleitung für die Vorlesung**

- Denken Sie mit
- Stellen Sie Fragen
- Schreiben Sie nur das Nötigste mit
- **Klappen Sie Ihr Notebook bitte zu!**

- **Nachbereitung**

- Arbeiten Sie die Vorlesung nach
- Geben Sie sich erst zufrieden, wenn Sie jedes Detail (mindestens einmal!) verstanden haben



# Übungsaufgaben (1/3)

- **Zweck der Übungsaufgaben**

- Wiederholung der Inhalte der Vorlesung
- Erkennen, wo es mit dem Verständnis noch hapert
- Entwicklung der Fähigkeit, die in der Vorlesung gelernten Techniken **anzuwenden**
- Klausurvorbereitung

- **Anforderungen an die Lösungen**

- Lösungen sind nur vollständig, wenn sie begründet und erklärt werden
- Falls Beweise erwartet werden, wird dies in der Aufgabenstellung ausdrücklich erwähnt

- **Gruppenarbeit**

- Sie können die Übungsaufgaben zusammen mit anderen bearbeiten und gemeinsam Lösungswege diskutieren
- Das **Aufschreiben** der Lösungen erfolgt aber individuell

- **Sanktionen**

- Wenn mehrere Personen offensichtlich voneinander abgeschriebene Lösungen abgeben, erhalten sie alle 0 Punkte
- Im Wiederholungsfall ist die Studienleistung verwirkt und wir behalten uns weitere Maßnahmen vor

# Übungsaufgaben (2/3)

## • Übungsaufgaben-Lebenszyklus

- Ausgabe Übungsblatt:
  - \* Dienstag Woche  $n$
- Abgabe Lösungen:
  - \* Dienstag Woche  $n+1$  bis 10:00 Uhr
- Besprechung:
  - \* Übung von Donnerstag Woche  $n+1$  bis Mittwoch Woche  $n+2$

## • Abgabe und Korrektur

- Abgabe:
  - \* in die Briefkästen zwischen OH 14 und OH 12
- Ihre Lösungen werden korrigiert
- Zu jedem regulären Übungsblatt wird eine **Beispiel-Lösung** veröffentlicht (am Ende von Woche  $n+2$ )
- Wichtig: es kann mehrere Lösungswege geben, die Beispiel-Lösung repräsentiert meist nur **einen** davon

# Übungsaufgaben (3/3)

- **Anzahl und Art der Übungsblätter**

- Blatt 0 mit Präsenzaufgaben (erste Übungsstunde)
- Übungsblätter 1-12 zur Bearbeitung und Abgabe

- **Aufgabentypen**

- pro Blatt ca. drei Aufgaben, deren Bearbeitung in der Regel jeweils 5 Punkte ergeben kann
- reguläre Aufgaben bestehen aus einer Kurzaufgabe und einer Hauptaufgabe
- möglicherweise eine schwierigere Zusatzaufgabe, die bei Bearbeitung korrigiert und bepunktet, jedoch nicht notwendigerweise besprochen wird
- zusätzlich Testfragen

# Übung

- **Lernziele in den Übungsstunden**

- mündliche Präsentation von Inhalten der Theoretischen Informatik
- die Fähigkeit zur Reflexion von Inhalten der Theoretischen Informatik, auch im Gespräch mit anderen

- Um dieses Ziel zu erreichen, werden

- die Lösungen der Übungsaufgaben von den Studierenden vorgeführt
- alternative Lösungswege und fehlerhafte Ansätze unter den Studierenden diskutiert
- Präsenzaufgaben bearbeitet

- **Termine:** siehe moodle

- **Beginn:** Donnerstag, 27.4.17

In der ersten Übungsstunde werden „Präsenzaufgaben“ besprochen

- **Anmeldung zu den Übungsgruppen**

- <http://ess.cs.tu-dortmund.de/ASSESS/>
- Anmeldung ab sofort bis Samstag, 22.4.17
- Bekanntgabe der Gruppeneinteilung bis Montag, 25.4.17, 18:00 Uhr

# Tutorien und GTI HelpDesk

## • Zweck des Tutoriums

- Wiederholung des Stoffes und Prüfungsvorbereitung
- Besondere Unterstützung der „ThIfAI“-Studierenden

## • Termine

- (B2) Mo 14-16, OH12, 1.055 (ThIfAI)
- (A1) Mi 14-16, OH12, E.003 (ThIfAI)
- (A2) Do 10-12, OH14, E 23
- (B1) Fr 12-14, OH12, 3.031
  - Beginn: Freitag, 21.4.17
  - ThIfAI-Studierende haben bei Raumüberfüllung an den gekennzeichneten Terminen Vorrang

## • Themen

- Bekanntgabe jeweils per moodle
  - \* gleiche Themen jeweils in den beiden A- und B-Tutorien
- Themenvorschläge per INPUD und moodle möglich

## • Termine GTI HelpDesk

- Mo 12-14 Uhr, OH12, 4.027
  - Fr 14-16 Uhr, OH12, 4.027
- Beginn: Freitag, 28.4.17

# Prüfung und Studienleistung (1/2)

- **Klausurtermine**

- 1. Klausur: 09.08.17, 8:00-11:00 Uhr
- 2. Klausur: 05.10.17, 8:00-11:00 Uhr

- **Erlaubte Hilfsmittel:** zwei DIN A4 Blätter mit selbst erstellten handschriftlichen Notizen (Details folgen)

- Anmeldung zur Klausur im BOSS

- **Vergewissern Sie sich bitte, dass die Anmeldung im BOSS geklappt hat!!!**
- Studierende, für deren Studiengänge keine Anmeldung im BOSS möglich ist, melden sich direkt bei uns an (nähere Informationen folgen)

## Prüfung und Studienleistung (2/2)

- Voraussetzung zur Teilnahme an der Klausur ist das Erbringen der **Studienleistung**:
  - Erreichen von
    - \* 50% der erreichbaren Punkte der regulären Übungsaufgaben der Blätter 1-6
    - \* 50% der erreichbaren Punkte der Blätter 7-12
  - Punkte müssen alle im gleichen Jahr erworben werden
- Lehramtsstudierende der Informatik benötigen zusätzlich eine Studienleistung in *Formale Methoden 2*
- Zum Erwerb der genannten Kompetenzen und als Vorbereitung zur Klausur empfehlen wir dringend die aktive Teilnahme an den Übungsgruppen
- Die Studienleistung muss nicht erbracht werden von:
  - Studierenden der Diplomstudiengänge
  - Studierenden, die die Studienleistung 2016 erbracht haben
  - Studierenden, die die Studienleistung irgendwann erbracht haben **und** bereits eine Klausur geschrieben haben (laut Auskunft Prüfungsausschuss)
- Dringende Empfehlung für diese Gruppen: nehmen Sie aktiv am Übungsbetrieb teil

# GTI im Web

- Auf der **Vorlesungsseite**/in **moodle** finden sich:

- die Vorlesungsankündigung
- die Übersicht der Übungstermine
- aktuelle Informationen
- Vorlesungsfolien
- Übungsblätter
- Beispiel-Lösungen
- Informationen zu den Klausuren

- Das Kennwort für moodle lautet GTI17turing

- Die Vorlesungsfolien gibt es in drei Varianten:
  - zum Anschauen am Bildschirm  
(jeder „Klick“ auf Extra-Seite)
  - zum Ausdrucken
  - zum Ausdrucken mit reduziertem Farbeinsatz

Öko-Tipp: 4-auf-1 ist noch gut lesbar

- Hinweis: nicht alle Abbildungen/Fotos werden in den Foliensätzen zur Verfügung gestellt

- Im INPUD-Forum und in moodle gibt es eine moderierte Diskussionsgruppe zu GTI
- Dort können Sie Fragen stellen, mit anderen über Vorlesung und Übung diskutieren, sich Lernpartner suchen usw.



## Das weitere GTI-Team

- Gaetano Geck (LS 1)
  - Thomas Harweg (LS 7)
  - **David Mezlaf** (LS 2)
  - Christopher Spinrath (LS 1)
- 
- Jonas Bode
  - Jana Frieese
  - Fabian Gürtler
  - Roland Ihle
  - Arthur Matei
  - Christoph Meyer
  - Bianca Ruland
  - Nina Runde
  - Marko Schmellenkamp
  - Jonas Wielage

**Zu guter Letzt...**

**Viel Erfolg!**

**Und auch etwas Spaß!**