Modulbeschrieb

Theoretische Informatik

Allgemeine Informationen

Anzahl ECTS-Credits

3

Modulkürzel

FTP_TheoComp

Version

19.03.2014

Modulverantwortliche/r

Olivier Biberstein, BFH

Sprache

	Lausanne	Bern	Zürich
Unterricht	☐ E ☐ F	☐ D ☐ E ☐ F	☒ D ☐ E
Unterlagen	☐ E ☐ F	☐ D ☐ E ☐ F	☒ D ☒ E
Prüfung	☐ E ☐ F	☐ D ☐ E ☐ F	☒ D ☐ E

Modulkategorie

☒ Erweiterte theoretische Grundlagen

☐ Technisch-wissenschaftliche Vertiefung

☐ Kontextmodule

Lektionen

☒ 2 Vorlesungslektionen und 1 Übungslektion pro Woche

☐ 2 Vorlesungslektionen pro Woche

Kurzbeschreibung /Absicht und Inhalt des Moduls in einigen Sätzen erklären

Ziel dieses Moduls ist die Vertiefung einiger grundlegenden theoretischen Aspekte der Informatik. Die Master-Studierenden werden lernen, dass…

- *formale Sprachen* und *Automaten* wesentliche Konzepte sind, um verschiedene Problematypen und Berechnungen zu beschreiben;
- *Berechenbarkeit/Entscheidbarkeit* zentral sind, um erklären zu können, dass es für viele Probleme eine intuitive Lösung zu geben scheint, obwohl sie nicht mittels Algorithmen gelöst werden können;
- *Komplexität* sich mit dem zur Lösung eines Problems benötigten Aufwand an Platz oder Zeit befasst, und es zudem viele sehr praktische Probleme gibt, die mit einem vernünftigen Aufwand an Zeit oder Platz nicht gelöst werden können.

Ziele, Inhalt und Methoden

Lernziele, zu erwerbende Kompetenzen

- Die Studierenden verstehen, dass drei unterschiedliche mathematische Formalismen (endliche Automaten, reguläre Grammatiken, reguläre Ausdrücke) gleichwertig sind und die Gruppe der regulären Sprachen definieren. Endliche Automaten und reguläre Ausdrücke sind weit verbreitet und werden im Rahmen dieser Vorlesung anhand von Beispielen aus der lexikalischen Analyse, der Modellierung einfacher, zustandsbasierter Systeme sowie anhand von Telekommunikationsprotokollen und der Programm–verifikation erklärt.
- Die Studierenden erkennen, dass Programmiersprachen mit regulären Sprachen nicht vollständig beschrieben werden können. Kontextfreie Grammatiken hingegen eignen sich, um alle modernen Programmiersprachen zu entwickeln. Parsing ist eng verknüpft mit den kontextfreien Sprachen. Anhand von Parser-Generatoren können die Studierenden Top-Down- und Bottom-Up-Parsing erklären.
- Die Studierenden wissen, dass viele Probleme unentscheidbar sind, d.h. dass es keine Algorithmen gibt, um sie zu lösen, oder vielmehr, dass nicht alles berechenbar ist. Solche intuitiv schwer zu verstehenden Probleme treten z.B. bei Betriebssystemen (Deadlock-Problem), objektorientierten Programmiersprachen (Subtyp-Entscheid) oder der Programmverifikation auf.
- Die Studierenden verstehen, dass entscheidbare Probleme gemäss den zur Lösung benötigten Ressourcen (Zeit oder Platz) klassifiziert werden und kennen die wichtigsten Komplexitätsklassen (P, NP, EXP, PSPACE), ihre Unterschiede und wechselseitigen Beziehungen (z.B. Hierarchie).

- Die Studierenden verstehen das Konzept des Nichtdeterminismus, welches bei der Untersuchung von Komplexität eine wesentliche Rolle spielt. Die Komplexitätsklasse NP (nichtdeterministisch polynomiale Zeit) beinhaltet eine Unter-Klasse von sehr praktischen Problemen, die nicht lösbar sind (NP-vollständig). Kryptologie, maschinelles Sehen, verschiedene Optimierungsprobleme und viele andere Gebiete sind von solch unlösbaren Problemen betroffen. Die Studierenden können aufzeigen, dass solche Probleme tatsächlich unlösbar sind, und kennen einige der Möglichkeiten, um diese Unlösbarkeit mittels Annäherungsverfahren zu umgehen. Viele als NP-vollständig bekannte Probleme werden vorgestellt und untersucht.

Modulinhalt mit Gewichtung der Lehrinhalte

Das Modul gliedert sich in drei Teile:

1. Sprachen und Automaten (ca. 36 %)
    a. Alphabete, Wörter, formale Sprachen, Grammatiken
    b. Endliche Automaten, reguläre Sprachen/Grammatiken, reguläre Ausdrücke, Nichtdeterminismus
    c. Kellerautomaten, kontextfreie Sprachen/Grammatiken
    d. Turingmaschinen
2. Berechenbarkeit/Entscheidbarkeit (ca. 21 %)
    a. Verschiedene Berechnungsmodelle, Church-Turing-These
    b. Reduktion eines Problems auf ein anderes
    c. Entscheidbare/unentscheidbare Probleme
    d. Berechenbare/nicht berechenbare Funktionen
3. Komplexität (ca. 43 %)
    a. Komplexitätsarten (Zeit, Platz)
    b. Komplexitätsklassen, Polynomialzeit-Komplexität, NP, Reduzierbarkeit der Polynomialzeit, NP-Vollständigkeit
    c. Annäherungsverfahren

Lehr- und Lernmethoden

- Frontalunterricht
- Präsentation und Diskussion theoretischer Fragestellungen
- Diskussion praktischer Beispiele, um die Lücke zwischen Theorie und Praxis zu schliessen
- Übungen und Selbststudium ausgewählter Themen

Voraussetzungen, Vorkenntnisse, Eingangskompetenzen

Gute Kenntnisse der Programmierung, Algorithmen und diskreten Mathematik.

Bibliografie

- *Introduction to the Theory of Computation*, Michael Sipser, Cengage Learning, 3rd International Edition, 2013.
  Referenz: http://www-math.mit.edu/~sipser/book.html (Homepage des Autors zum Buch)
- *Computers Ltd.: What They Really Can't Do*, David Harel, Oxford University Press, 2000.
- *Introduction to automata theory, languages, and computation*, J.E. Hopcroft and J.D. Ullman, Addison-Wesley Publishing Company, Reading, MA, 1979.

Leistungsbewertung

Zulassungsbedingungen für die Modulschlussprüfung (Testatbedingungen)

Keine

Schriftliche Modulschlussprüfung

Prüfungsdauer :	120 Minuten
Erlaubte Hilfsmittel:	Zusammenfassung (10 Seiten)