Technische Universität Dresden
Fakultät Informatik

Studienordnung
für den internationalen Masterstudiengang
Computational Logic
an der Technischen Universität Dresden

Vom 09.04.2003

Aufgrund von § 24 des Gesetzes über die Hochschulen im Freistaat Sachsen (Sächsisches Hochschulgesetz - SächsHG) vom 11. Juni 1999 (SächsGVBl. S. 293) erlässt die Technischen Universität Dresden die nachstehende Studienordnung.

Inhaltsverzeichnis

§ 1 Geltungsbereich

Diese Studienordnung regelt auf der Grundlage des Sächsischen Hochschulgesetzes und der Prüfungsordnung Ziel, Inhalt und Ablauf des Studiums im internationalen Masterstudiengang Computational Logic.

§ 2 Ziel des Studiums

(1) Ziel des Studiums ist es, dem Studenten[1] die für die Berufspraxis notwendigen gründlichen theoretischen und praktischen Fachkenntnisse zu vermitteln, ihm einen Überblick über die einzelnen Disziplinen der Computational Logic zu geben und seine Fähigkeit zu entwickeln, nach wissenschaftlichen Methoden zu arbeiten. Darüber hinaus wird dem Studenten die Möglichkeit geboten, sein Studium tätigkeitsfeldbezogen zu gestalten. Zur Vermittlung eines an spezifischen Tätigkeitsfeldern orientierten Wissens kann er entsprechende Modulkombinationen wählen und dadurch seiner Ausbildung eine spezielle Richtung geben. Durch Auslandsaufenthalte und Englisch als Lehr– und Arbeitssprache soll der Student auf die zunehmende Internationalisierung von Wissenschaft, Wirtschaft und Industrie vorbereitet werden.

(2) Die Schwerpunkte bei der Ausbildung liegen in der Vermittlung des Wissens in den folgenden Gebieten: Mathematische Logik, Logikprogrammierung, Deduktionssysteme, Wissensrepräsentation, künstliche Intelligenz, Methoden der formalen Spezifikation und Verifikation, Inferenztechniken, syntaxgesteuerte Semantik sowie Verbindung zwischen theoretischer Informatik und Logik.

(3) Die Masterprüfung bildet den weiterführenden berufs- und forschungsqualifizierenden Abschluss des internationalen Masterstudiengangs Computational Logic. Durch sie soll festgestellt werden, ob der Kandidat die Zusammenhänge seines Faches überblickt, die für die Berufspraxis notwendigen Fachkenntnisse und Fertigkeiten erworben hat, und die Fähigkeit besitzt, selbständig wissenschaftliche Methoden und Kenntnisse anzuwenden. Aufgrund der bestandenen Masterprüfung verleiht die Technische Universität Dresden den akademischen Grad “Master of Science” (abgekürzt: M.Sc.). In dem Zeugnis und dessen Übersetzung wird vermerkt, dass der akademische Grad im internationalen Masterstudiengang Computational Logic erworben wurde.

§ 3 Studienvoraussetzungen

(1) Bewerber für den internationalen Masterstudiengang Computational Logic müssen die folgenden Studienvoraussetzungen erfüllen:

1. Nachweis von Mindestkenntnissen in Englisch durch Vorlage eines IELTS–Zertifikats oder eines vergleichbaren Nachweises. Bewerber mit Englisch als Muttersprache sind von dieser Regelung ausgenommen.
2. Bachelor in Informatik (Computer Science), mit einer Regelstudienzeit von sechs Semestern, oder einen durch den Prüfungsausschuss als gleichwertig anerkannten Abschluss.

[1] In der Ordnung gelten maskuline Personenbezeichnungen ebenso für Personen weiblichen Geschlechts.

3. Nachweis von mindestens guten bzw. sehr guten Kenntnissen im Bereich
   der Grundlagen der mathematischen Logik,
   der Grundlagen der Künstlichen Intelligenz und
   der Programmiersprache Prolog.
4. Die in Punkt 3. geforderten Studienleistungen können durch Zeugnisse, Prüfungsbescheinigungen oder andere schriftliche Leistungsnachweise nachgewiesen werden.

(2) Über das Vorliegen der in Absatz 1 genannten Voraussetzungen entscheidet der für diesen Studiengang zuständige Prüfungsausschuss.

(3) Die Studenten werden an der Technischen Universität nach den dafür geltenden Bestimmungen immatrikuliert.

§ 4 Studienbeginn, Studiendauer

(1) Das Studium beginnt für Studienanfänger in der Regel mit dem Wintersemester.

(2) Die Regelstudienzeit einschließlich Anfertigen und Verteidigung der Masterarbeit beträgt vier Semester.

(3) Studenten, die die Studienvoraussetzungen nach § 3 Abs. 1 Nr. 2 an einer deutschen Universität erworben haben, müssen in der Regel ein Semester der Regelstudienzeit an einer ausländischen Hochschule studieren. Das Auslandssemester kann auch dem Anfertigen der Masterarbeit unter Betreuung eines an der ausländischen Hochschule ansässigen Hochschullehrers dienen. Die Entscheidung über Antritt und Zeitpunkt des Auslandssemesters trifft der Prüfungsausschuss für den internationalen Masterstudiengang Computational Logic auf Antrag. Der Prüfungsausschuss entscheidet ebenfalls auf Antrag, ob im Einzelfall von der Regelung abgesehen werden kann. Rechtzeitig vor Antritt des Auslandssemesters muss sich der Student von einem Hochschullehrer bezüglich der an der ausländischen Hochschule zu besuchenden Lehrveranstaltungen oder der Thematik der Masterarbeit beraten lassen.

§ 5 Vermittlungsformen

(1) Der Lehrstoff ist modular strukturiert. In den einzelnen Modulen werden die Lehrinhalte durch Vorlesungen, Übungen, Seminare und Praktika vermittelt, gefestigt und vertieft.

(2) In Vorlesungen wird der Lehrstoff vermittelt. Übungen sind den Vorlesungen zugeordnet und dienen dem Durcharbeiten des Vorlesungslehrstoffes. In ihnen diskutieren die Studenten in arbeitsfähigen Gruppen unter Anleitung ihre Lösung zu Übungsaufgaben. Seminare dienen der Entwicklung der Fähigkeit des Studenten, sich vorwiegend auf der Grundlage von Literatur, Dokumentationen und sonstigen Unterlagen über einen Problemkreis zu informieren, das Erarbeitete vorzutragen und zu vertreten. Praktika dienen der praktischen Anwendung und Vertiefung des vermittelten Lehrstoffes sowie dem Erwerb von praktischen Fertigkeiten bei der Arbeit mit Hard- und Software.

(3) Die Lehr-, Arbeits- und Prüfungssprache ist Englisch. Studenten können mündliche Prüfungen in deutscher Sprache ablegen.

§ 6 Aufbau und Durchführung des Studiums

(1) Das Lehrangebot ist auf drei Semester verteilt. Es umfasst Lehrveranstaltungen mit einem Gesamtumfang von 90 ECTS–Punkten (credits; abgekürzt: crs).

(2) Die Ausbildung ist in einen für alle Studenten obligatorischen Teil (Pflichtmodule) und einen frei wählbaren Teil (Wahlpflichtmodule) gegliedert. Sie umfasst:

36 crs Pflichtmodule

42 crs Wahlpflichtmodule und

12 crs in einem Praktikum

30 crs Masterarbeit und deren Verteidigung

(3) Die Aufteilung der Module auf die einzelnen Semester ist in der beigefügten Stundentafel, Anlage 1, enthalten.

(4) Die Bildungsziele der einzelnen Module, die notwendigen Voraussetzungen und die Abhängigkeiten zwischen den Modulen sind der Anlage 2 Modulbeschreibungen zu entnehmen.

(5) In der abschließend zu erstellenden Masterarbeit soll der Student zeigen, dass er in der Lage ist, ein Problem der Computational Logic oder deren Anwendungen selbständig nach wissenschaftlichen Methoden zu bearbeiten.

(6) Das Studium schließt mit der Masterprüfung ab.

(7) Dem Studenten wird empfohlen, seine im Studium erworbenen Kenntnisse, Fähigkeiten und Fertigkeiten in einer berufspraktischen Tätigkeit zu vertiefen.

§ 7 Prüfungen und ECTS-Punktesystem

(1) Das Studium wird mit der Masterprüfung abgeschlossen. Die Masterprüfung besteht aus zwei Teilen, den Modulprüfungen sowie der Masterarbeit einschließlich der Verteidigung. Modulprüfungen werden in Form von studienbegleitenden Prüfungsleistungen erbracht. Der Notenausweis für Modulprüfungen erfolgt nach der ECTS-Skala mit den deutschen Notenbezeichnungen gemäß § 11, Abs. 1 der Prüfungsordnung.

(2) ECTS-Punkte werden dann gewährt, wenn die Modulprüfung erfolgreich bestanden wurde. Das ECTS-Punktesystem bietet eine einheitliche Vorgehensweise für die Anerkennung von im Ausland erbrachten Studienleistungen.

§ 8 Studienfachberatung

Die Beratungen in Studien– und Prüfungsangelegenheiten, zu Studienvoraussetzungen und Hochschulwechsel, zu Auslandsaufenthalten und zu allen mit dem Studium in Zusammenhang stehenden Angelegenheiten werden von der Studienfachberatung der Fakultät Informatik der Technischen Universität Dresden, entsprechend dem internationalen Charakter des Masterstudienganges auch über das Internet, durchgeführt.

§ 9 In-Kraft-Treten und Veröffentlichung

(1) Diese Studienordnung gilt für die ab Wintersemester 2002/03 immatrikulierten Studenten. Für alle früher immatrikulierten Studenten gilt die Studienordnung vom 26.11.1997, in der geänderten Fassung vom 09.05.2000).

(2) Diese Studienordnung tritt mit Wirkung vom 01.10.2002 in Kraft und wird in den Amtlichen Bekanntmachungen der Technischen Universität veröffentlicht.

Anlage 1: Stundentafel in crs

Module	**Semester**			
	1	**2**	**3**	**4**
Introduction to Computational Logic	9			
Foundations of Logic and Constraint Programming	9			
Advanced Logic		9		
Deduction Systems		9		
Wahlpflichtmodule	42			
Praktikum			12	
Masterarbeit				30
Gesamt crs	**120**			

Anlage 2: Modulbeschreibungen

Module: Introduction to Computational Logic

Contact person: Steffen Hölldobler

Keywords: propositional logic; first order logic; deduction; abduction and induction; knowledge representation and reasoning

The module offers a comprehensive introduction to Computational Logic covering the main subareas as well as main methods and techniques. After recalling basic notions from propositional and firster order logic the areas of equational reasoning, deduction, abduction and induction, non-monotonic reasoning, logic-based program development, natural language processing and machine learning as well as logic and connectionism are covered.

This module consists of lectures and tutorials. The total of 9 credit points can be scored by passing the final written examination of the module.

The module takes one semester and is offered every winter semester.

Prerequisites: none

Module: Foundations of Logic and Constraint Programming

Contact person: Michael Thielscher

Keywords: unification; declarative, procedural, and operational semantics; Prolog; nonmonotonic negation; constraint logic programming

This module is concerned with the foundations of logic programming and constraint logic programming. The basic computation mechanisms of unification and SLD-resolution are introduced. The declarative and the operational semantics of logic programs are given and related to the procedural semantics. Prolog is introduced as an example of a logic programming language. The module covers logic programs with negation by giving the basic computation mechanism of SLDNF-resolution and by discussing several competing, nonmonotonic standard semantics are discussed. Logic programs with constraints are introduced and basic computation mechanisms is given. The module concludes with examples of constraint logic programming languages. After the successful completion of this module, students will have acquired a profound understanding of the mathematical principles of logic programming. Students will also have experience in using logic programming languages and constraint logic programming languages for problem solving.

This module consists of lectures and tutorials. The total of 9 credit points can be scored by passing the final written examination of the module.

The module takes one semester will be offered every winter semester.

Prerequisites: none

Module: Advanced Logics

Contact person: Horst Reichel

Keywords: higher order logics, typed lambda calculus, lambda prolog, modal logics, epistemic logic, temporal logic, mu-calculus, CTL*, schematic tableaux

The aim of this module is to introduce basic concepts behind first-order predicate logics. In Computer Science many different logics and deductive systems exist. First we introduce higher order logic (HOL) as a framework for specifying syntactic and deductive notions of different logics. HOL is used in several interactive proof tools, like PVS and Isabelle. In addition, specific families of logics aimed at different application areas are introduced: logics of time and computation (modal logics, temporal Logics), logics for reasoning about knowledge (epistemic logic). Finally we introduce the mu-calculus which allows to define recursive temporal properties and we present a tableau based deduction calculus for the mu-calculus. The mu-calculus and its deduction system can be used to define problem oriented systems of modal operators and corresponding deduction systems.

This module consists of lectures and tutorials. The total of 9 credit points can be scored by passing the final written examination of the module.

The module takes one semester will be offered every summer semester.

Prerequisites: none

Module: Deduction Systems

Contact person: Michael Thielscher

Keywords: Prolog implementation techniques; abstract machines; low-level logic data structures and algorithms; calculi for first-order deduction; theorem proving systems

This module is concerned with theory and practice of systems for automated deduction. It introduces the concept of abstract machines as the basis for implementing Prolog systems. The basic low-level representation techniques for logical terms are covered and the key algorithms in logic programming, namely, unification and backtracking, are presented. Based on these implementation mechanisms, various specific behaviors of Prolog systems are analysed. Building on basic knowledge of the resolution calculus, the standard first-order deduction calculi of natural deduction and sequent calculus are covered. The design and use of automated theorem proving is studied with the help of exemplary deduction systems. After the successful completion of this module, students will have acquired a profound understanding of the implementation principles of logic-based systems, which enables them to analyze specific behaviors of systems and to build efficient deduction systems.

This module consists of lectures and tutorials. The total of 9 credit points can be scored by passing the final written examination of the module.

The module takes one semester will be offered every summer semester.

Prerequisites: none

Module: Knowledge Representation and Artificial Intelligence

Contact person: Michael Thielscher

Keywords: declarative representation of knowledge; automated reasoning with knowledge; knowledge-based systems; artificial intelligence applications

This module is concerned with techniques for the declarative representation of knowledge and inference methods based on formalized knowledge. Introduced are standard representation formalisms for various kinds of knowledge (like temporal, dynamic, categorical, or grammatical knowledge). The mathematical properties of the formalisms are discussed. Calculi for inferring knowledge are given and analyzed. Principles for designing and building knowledge-based systems are introduced, and applications of knowledge representation and reasoning techniques to artificial intelligences are covered. The successful completion of this module enables students to understand and create knowledge representation formalisms, to analyze, design, and use algorithms for drawing inferences from formal knowledge, and to build and apply knowledge-based systems.

The total number of 14 credit points for this module are attained by lectures with tutorials, and possibly a seminar.

The module can be completed within two successive semesters and it will be offered every year.

Prerequisites: none

Module: Specification and Verification

Contact person: Horst Reichel

Keywords: constructive and declarative specification techniques, algebraic and coalgebraic specifications, inititial and final semantics, process algebras, Petri nets, induction, coinduction, distributed computing.

The module presents formal specification techniques for both the axiomatic and operational specification of software (and hardware) systems. The students learn to specify generic data types and functional enrichments of generic data types by means of initial semantics, to prove properties by induction, and to reason about the correctness of refinements. They can learn to specify the dynamic behavior of a system by means of Petri nets, to use algorithms on Petri nets to reason about the behavior, and to model concurrent systems by means of process algebras and to express dynamic properties using modal and temporal properties, and to apply coinduction as a fundamental definition and proof technique.

The total number of 14 credit points for this module are attained by lectures with tutorials, and possibly a seminar.

The module can be completed within two successive semesters and will be offered every year.

Prerequisites: basic knowledge in predicate logic and modal logics as presented in the module "Advanced logics"

Module: Theoretical Computer Science and Logic

Contact person: Franz Baader

Keywords: complexity and computability theory, automata theory, algorithms, algebra, model theory

This module is concerned with the application of advanced techniques and results from theoretical computer science (like automata on infinite objects, complexity results, term rewriting techniques, etc.) to the analysis of formal properties of different logics (like axiomatizations, proof-theoretic properties, design of algorithms and analysis of the complexity for logical inference problems, etc.). Building on the basic knowledge about automata, formal languages, and computability from the Vordiplom or the Bachelor studies and the introductory courses in CL, this module will introduce different such advanced techniques and then show how they can be applied in Computational Logic. After a successful completion of the module the students should have both, a working familiarity with different methods of theoretical computer science, and a good knowledge of formal properties of various logics.

The total number of 14 credit points for this module are attained by lectures with tutorials, and possibly a seminar.

The module can be completed within two successive semesters and it will be offered every year.

Prerequisites: basic knowledge in theoretical computer science and logics

Module: Syntax-Directed Semantics

Contact person: Heiko Vogler

Keywords: denotational semantics, implementation of imperative, functional, and logic-programming program schemes of functional programming and tree transducers weighted automata

The contents of this module is the investigation of translations of syntactic structures into semantic structures. Here syntactic structures have the form of trees like derivation trees of programs; semantic structures can be complexes like 1. state space, environment, and continuations or 2. dependency graph between semantic values or 3. abstract machines. By abstracting from the operations in the semantic domain, we obtain trees as another, very general type of semantic structure. Whereas the more particular semantic structures lead to the areas of denotational semantics and implementation of imperative, functional, and logic-programming, the more abstract point of view can be called the theory of program schemes of functional programming and the theory of tree transducers (with weights).

The total number of 14 credit points for this module are attained by lectures with tutorials, and possibly a seminar.

The module can be completed within two semesters and it will be offered permanently.

Prerequisites: basic knowledge about automata, formal languages, and computability as it gained in the Vordiplom or the Bachelor level.

Module: Inference Techniques

Contact person: Steffen Hölldobler

Keywords: resolution; term rewriting; answer set programming; inductive theorem proving

The module is concerned with the in-depth study of inference techniques. After recalling basic notions and techniques from automated reasoning some of the following methods and techniques will be presented in detail: Resolution, tableaux, connection or related methods for automated theorem proving; Term rewriting, superposition or related methods for equational reasoning; Answer set programming or related methods for non-monotic rasoning; Inductive theorem proving.

The total number of 14 credit points for this module are attained by lectures with tutorials, and possibly a seminar.

The module can be completed within two successive semesters and will be offered every second year.

Prerequisites: Basic knowledge in logic and reasoning as presented in the module "Introduction to Computational Logic".